AP2PS 2010

The Second International Conference on Advances in P2P Systems

October 25-30, 2010 - Florence, Italy

Editors

Nick Antonopoulos

Antonio Liotta

Yasushi Kambayashi

AP2PS 2010

Foreword

The Second International Conference on Advances in Peer-to-Peer Systems (AP2PS 2010), held from October 25 to October 30, 2010 in Florence, Italy, aimed at capturing the latest developments, findings and proposals in the general area of P2P computing, networking, services, and applications. The areas of interest included topics such as advances in theoretical foundations of P2P, performance analysis of P2P frameworks and applications, security, trust and reputation in P2P, time-constrained P2P systems, and quality of experience in P2P systems.

Peer-to-peer systems have considerably evolved since their original conception, in the 90's. The idea of distributing files using the user's terminal as a relay has now been widely extended to embrace virtually any form of resource (e.g., computational and storage resources), data (e.g. files and real-time streams) and service (e.g., IP telephony, IP TV, collaboration). More complex systems, however, require more sophisticated management solutions, and in this context P2P can become an interesting issue, playing the hole of both the target and the enabler of new management systems.

We take here the opportunity to warmly thank all the members of the AP2PS 2010 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to the AP2PS 2010. We truly believe that thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the AP2PS 2010 organizing committee for their help in handling the logistics and for their work that is making this professional meeting a success. We gratefully appreciate to the technical program committee co-chairs that contributed to identify the appropriate groups to submit contributions.

We hope the AP2PS 2010 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in peer-to-peer systems.

We hope Florence provided a pleasant environment during the conference and everyone saved some time for exploring this historic city.

AP2PS 2010 Chairs:

Marco Aiello, University of Groningen, The Netherlands
Nick Antonopoulos, University of Derby, UK
Petre Dini, IARIA / Concordia University, Canada
Takahiro Hara, University of Osaka, Japan
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Roman Y. Shtykh, Rakuten, Inc., Japan

Yasushi Kambayashi, Nippon Institute of Technology, Japan
Antonio Cuadra-Sanchez, Telefónica Investigación y Desarrollo - Madrid, Spain
Anders Fongen, Norwegian Defense Research Establishment, Norway

AP2PS 2010

Committee

AP2PS Advisory Chairs
Marco Aiello, University of Groningen, The Netherlands
Nick Antonopoulos, University of Derby, UK
Petre Dini, IARIA / Concordia University, Canada
Takahiro Hara, University of Osaka, Japan
Antonio Liotta, Eindhoven University of Technology, The Netherlands

AP2PS 2010 General Chairs
Nick Antonopoulos, University of Derby, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands

AP2PS 2010 Industry Liaison Chair
Roman Y. Shtykh, Rakuten, Inc., Japan

AP2PS 2010 Research/Industry Chairs
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Antonio Cuadra-Sanchez, Telefónica Investigación y Desarrollo - Madrid, Spain
Anders Fongen, Norwegian Defense Research Establishment, Norway

AP2PS 2010 Technical Program Committee
Jemal H. Abawajy, Deakin University - Geelong, Australia
Paulina Adamska, Polish-Japanese Institute of Information Technology - Warsaw, Poland
Marco Aiello, University of Groningen, The Netherlands
Antonio Alfredo F. Loureiro, Federal University of Minas Gerais, Brazil
Nikos Antonopoulos, University of Derby, UK
Farnoush Banaei-Kashani, University of Southern California, USA
Ataul Bari, University of Windsor, Canada
Paolo Bellavista, Università degli Studi di Bologna, Italy
Andreas Berl, University of Passau, Germany
Frances Brazier, TU Delft, The Netherlands
Julian Buhagiar, University of Malta, Malta
Dumitru Dan Burdescu, University of Craiova, Romania
Aldo Campi, University of Bologna, Italy
Juan Carlos Cano, Universidad Politécnica de Valencia, Spain
Charalampos Chelmis, University of Southern California, USA
Xinuo Chen, University of Warwick, UK
Chou Cheng-Fu, National Taiwan University, Taiwan
Giovanni Chiola, University of Genoa, Italy
Yeh-Ching Chung, National Tsing Hua University, Taiwan-China
Carmela Comito, University of Calabria - Rende, Italy
Fernando Cores Prado, University of Lleida, Spain
Noël Crespi, IT-ParisSud, France
Antonio Cuadra-Sanchez, Telefónica Investigación y Desarrollo - Madrid, Spain

Quang Hieu Vu, Institute for Infocomm Research, Singapore
Frank E. Walter, ETH Zurich, Switzerland
Wenjing Wang, University of Central Florida, USA
Ouri Wolfson, University of Illinois at Chicago, USA
Yun Yang, Swinburne University of Technology - Melbourne, Australia

Table of Contents

SPP: A Secure Protocol for Peer-to-Peer Systems

Quang Hieu Vu [1,2]
[1] *Cryptography and Security Department, Institute for Infocomm Research, Singapore*
[2]*ETISALAT BT Innovation Center, Khalifa University, UAE*
qhvu@i2r.a-star.edu.sg

***Abstract*—The main challenge of reputation-based methods that are used to evaluate trust of peers in Peer-to-Peer (P2P) systems is how to collect and distribute reputation scores of peers efficiently. While several protocols have been proposed to address this challenge, most of them rely on a gossiping algorithm, which is expensive and communication-intensive. In this paper, we propose SPP, a Secure Protocol for P2P trust management using trees in which we present a trust model between nodes in a tree, and explain how trust is established and maintained between pairs of nodes. We show that, compared to existing methods, our design allows for scalability and efficient algorithms with low overhead. We present a possible implementation of our basic tree design, and explain how it could be made stable and robust to network dynamism, thus addressing the greatest weakness of a tree structure. We also analyze the implementation for its security against various adversarial scenarios, and suggest further improvements that are possible for general tree-based systems.**

Keywords - Trust management; Security; Protocol; P2P.

I. Introduction

While Peer-to-Peer (P2P) systems have become very popular, security is still a problem of greatest concern among people using these systems. It is because in P2P systems, peers are usually anonymous. A popular method for evaluating trust in P2P systems is to use reputation, where the reputation of peer is determined based on its prior transactions with other peers. The main challenge of this method is how collect opinions of all peers in the system about a particular peer, and to provide access to the reputation score to all who request it. In existing reputation-based systems like eBay [1] and Amazon Auctions [2], the solution to both challenges is to use servers. However, this solution suffers from problems of server-based systems such as network bottlenecks, and having a single point of failure.

An alternative solution is to employ a gossiping algorithm [3], [4], [5], [6] for exchanging knowledge among peers in the system. In this way, after a sufficient number of knowledge exchange steps, every peer should have a global knowledge about reputations of peers in the system. The gossiping algorithm can be implemented in two ways. In the first way, each peer itself has to maintain global state and knowledge of the whole system. After each transaction or after some interval time, peers report the score of their partners in new transactions to all other peers in the system. Based on this report, peers update their global state. This method requires that peers keep and maintain reputation scores for all peers, which is inefficient. The second way avoids this problem by letting each peer keep track of the reputation of peers that it has been in transactions with previously. Whenever a peer wants to retrieve the reputation of another peer, it can apply the gossiping algorithm to ask for that peer's reputation from its neighbors, the neighbors of its neighbors, and so on. Combining the feedback with its local knowledge, it can determine a trust value of that peer. Even though these two ways are different, they share the same drawback of the gossiping algorithm: both are expensive in terms of computation and communication costs.

Instead of using gossiping, in this paper, we present SPP, a Secure Protocol for trust management in P2P systems based on a tree structure. Our method organizes nodes at different positions in a tree based on their reputation, with peers of higher reputation at higher levels. In this tree structure, reputation of a peer is maintained at its parent. A peer always trusts its ancestors while it is answerable for its descendants. When two peers execute a transaction, a trust route is formed between them. If the transaction succeeds, a reward is given to all nodes in the route. On the other hand, if the transaction fails, all nodes in the route are penalized. The main advantage of SPP is that it does not incur a high cost in reputation management compared to methods that use the gossiping algorithm for reputation distribution. Furthermore, the flexible design of SPP allows us to develop a complete system for trust management for use in any existing decentralized P2P system. To sum up, our paper makes the following contributions in the area of P2P security:

- We formulate a general-purpose solution to trust management in P2P systems based on a tree structure and show how to augment a tree with extra links to create robustness and to allow nodes to exchange queries without overwhelming the root. This eliminates the problems of bottlenecks and single points of failures.
- We extend BATON [7], an existing tree structure, to support our proposed protocol, implement the protocol, and conduct an experimental study to evaluate the effectiveness and efficiency of our protocol.

The rest of the report is organized as follows. In Section II, we introduce related work in the area of trust management in

P2P systems. In Section III, we explain our proposed basic design in terms of the trust and security models. In Section IV, we discuss some issues of our basic design, and suggest possible solutions to improve it. In Section V, we present a way to use our design to extend an existing tree structure (BATON [7]) to support reputation management. Section VI describes our experimental study and its results. Finally, in Section VII, we summarize the important contributions of our design and its potential.

II. Related Work

Trust management in P2P systems can be classified into two main categories: credential-based and reputation-based management. Credential-based management systems employ the classical method where a peer trusts another peer after examining the other peer's credentials. If the credentials satisfy the peer's policy, that peer can be trusted in a transaction. Otherwise, the peer would refuse to be in a transaction with the other peer. The weakness of this method is that it has to rely on servers for keeping every peer's credentials, which is not entirely a scalable method. Moreover, since credentials are usually generated once and stored, past transactions of peers, both good and bad, are not considered. As a result, this method is only suitable for specific kinds of systems with fixed credentials, like access control systems. Examples of systems that apply this trust model include X.509 [8], PGP [9], PolicyMaker [10] and its successors, REFEREE [11] and KeyNote [12].

On the other hand, reputation-based management systems rely on reputation to evaluate the trustworthiness of a node. In general, the reputation of a node is computed based on its previous transactions with other nodes in the system and how they rated these transactions. Reputation-based management systems can be further classified into two sub-categories. One type of system considers only the reputation of an individual, like those in [3], [4], [5], [6], [13], [14], [15], while the other takes into account social relationships between nodes in addition to individual reputation, such as [16], [17]. Since no nodes know of all nodes in the system, reputation of nodes have to either be collected and stored on servers for reference or distributed to all nodes in the network by the gossiping algorithm. Both of these methods are not viable for large networks because the first method is not scalable while the second method is expensive.

In the field of data structures, the structure of a tree has a very important role. NICE[1] can be used to do scalable application layer multicast [18] by using the idea of overlay trees for efficient content distribution. However, very few networks proposed so far uses the topology of a tree. In this kind of structure, if the standard query processing algorithm is used, nodes near the root will be accessed many times more compared to nodes near the leaves, and hence congestion at the root or nodes near the root may happen. This is not acceptable in P2P systems. To avoid this problem, P-Tree [19] suggests a use of partial tree structure. In this method, each leaf node in the tree is represented by a P2P node while internal nodes are all virtual. Each P2P node maintains a path from the index root to the leaf node. As a result, queries can be processed at any node without pushing all queries to a special node. Note that, however, if a node has to maintain the whole tree structure, the maintenance cost is very expensive and not suitable for P2P systems. Alternatively, BATON [7] creates links between nodes at the same level in the form of routing tables. Consequently, queries can be processed at any node in the tree without going through the root. Nevertheless, these systems focus only on range query processing, and not trust management.

[1]NICE is a recursive acronym for "NICE is the Internet Cooperative Environment". See http://www.cs.umd.edu/projects/nice/.

III. Basic Design

A. Trust Model

Our tree consists of peers arranged by their reputation. Peers of higher reputation occupy positions at higher levels in the tree, with each parent having a higher reputation score than their children, and so the root node is the peer with the highest reputation. Peers of higher reputation are accorded higher privileges of some kind, to provide incentive for nodes to increase their own reputation. We develop the following terminology and use it to present the model of trust relationships between nodes in the tree.

- *Trust link*. A link exists between a peer and its child, and this denotes a link of trust. We say that (1) the child peer in this link trusts its parent because the parent has a higher reputation than itself, and (2) the parent is answerable for the child. The latter point means that any misbehaved action on the part of the child reflects poorly on the parent as well, and the parent is also held accountable for any misbehavior of the child. This is desirable because it is every peer's responsibility to minimize the presence of malicious peers entering the network as children. Trust links are inherently transitive, because a child that trusts its parent would also trust its parent's parent of higher reputation, while a parent is accountable for its children and thus its children's children as well.
- *Trust chain*. A chain of trust is formed by consecutive trust links. In such a chain, we say that the lowest peer trusts the highest peer, based on transitivity of trust in our model.
- *Trust route*. A trust route is the path between any two peers in the tree. It is composed of one or two trust chains that meet at a common ancestor of the two nodes. We call that ancestor the *connector* of the route. The trust route also includes the connector's parent, which we label as the *arbiter* of the route. A trust

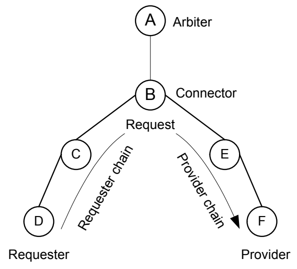


Figure 1. Trust relationships in a trust route

route is formed when a peer requests content from another peer in the system. The former peer is known as the *requester*, and the latter is the *provider*. The trust chain from the requester to the connector is called the requester chain, and that from the provider is called the provider chain. Figure 1 illustrates the relationships mentioned here.

- *Transaction*. A transaction is initiated by a requester, by sending a request through the tree to a chosen provider. The provider responds with the appropriate content to the requester. Transactions occur over a trust route in our tree, and they have a *transaction outcome* in the form of a report sent out by the requester. A positive outcome indicates a successful transaction when the requester is satisfied with the received information. Conversely, a negative outcome indicates a failed transaction when the requester is not satisfied with some part of the received information.
- *Rewards and punishments*. To give a reward means a peer increases the reputation score of a child peer, and a punishment is the converse, a decrease in the reputation score of the child peer. Rewards and punishments are managed based on the transaction outcomes reported by requesters.

B. Trust Management

This subsection describes how trust in our model can be managed. There are two possible outcomes of transactions each of which is dealt with in a separate way.

Successful transactions: if a successful transaction occurs between two nodes via a trust route, parent nodes would reward the child nodes. The rationale is that rewarding a child would allow it to be trusted by more nodes, and hence to increase its potential for bringing in more transactions for itself. This would lead to more opportunities for the parent node to earn its own rewards. In general, after a successful transaction, the arbiter rewards the connector, the connector rewards both the children in the requester and the provider chains, and so on, downward both trust chains. The only exception is the requester, which does not get any reward for initiating a request, since it adds no value to the network.

Failed transactions: for a failed transaction, the converse happens. The arbiter punishes the connector, which in turn pushes the blame downward the tree from parents to children in both chains. The requester again is unaffected by the punishments because it has nothing to gain or lose for accurately reporting the outcome of the transaction. A truthful report would, however, increase the effectiveness of the whole network. To prevent the malicious scenario of a node deliberately reporting multiple failed transactions, a parent might keep track of node failure reports, and identify any nodes that are misbehaving in this way. The parent could then terminate trust links with any evil node, deeming it to be deliberately causing trouble by either requesting content from reputably bad nodes, or inaccurately reporting many failed transactions.

This protocol leads to several implications: Nodes will try to maximize the number of successful transactions and minimize the number of failed ones, in order to optimally increase their reputation. This selfish and self-centered behavior, however, allows for optimal gains for the system as a whole, because each node selfishly seeks to maximize its own rewards and to do so, it has to shrewdly monitor its children and their behavior in transactions. A node would quickly break off links with children that result in many failed transactions and refuse to forward transactions from such nodes, because it is being held accountable for the behavior of its children. At the same time, a node would be willing to forward requests and content from reputable nodes or new nodes because doing so would give it the potential to increase its reputation.

C. Node Ranking Management

If we want to know reputation score of a node, we have to ask its parent, since the parent in our tree is of higher reputation and is thus more trustworthy. If an internal node cannot accomplish its task or turns malicious, we should replace it with a better node. Since a node may never want to step down from its position, we have to exert control over that node through its parent.

Additionally, reputation scores of a node is not only stored at the parent but also at the grandparent. Consider the situation where a node now has a reputation lower than that of its child, implying that the tree is currently not well-formed. The solution to this situation is to swap the positions of these two nodes through a swap operation, and that can only be done from the position of the parent of the ill-placed node. By changing positions, these nodes also exchange knowledge information of their children and reputation of these children they are keeping. An example of node swapping is shown in Figure 2 in which node B has to swap its position with its child E because E has a higher reputation. Actually, since A knows reputation of all B, C, D, E, F, G, it can also swap positions between B and G if G has a better reputation than both B and E. This sort of swapping can be done if A wants to assign a node that is known to be trustworthy from another subtree to be

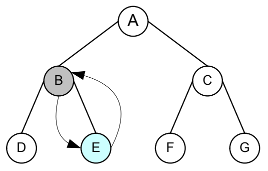


Figure 2. Swapping positions between nodes

the parent of a subtree that could possibly contain colluding malicious nodes.

IV. An Improved Model

The above basic model works well under an assumption that the information given by a node to another node about its children is always correct. In other words, all internal nodes can be trusted in giving information. This is because if an internal node is bad, it can return wrong reputation results about its children to other nodes. For example, a malicious node could return a good reputation score about a bad node or a bad reputation score for a good node. To avoid the problem of the basic model, we introduce a new type of score called a *reference score* for internal nodes. The reference score is used to reflect exactness of information a node gives to others. Now, the trust value of a node is based on not only its reputation score but also the reference score of its parent. In other words, if a node always gives correct information about its children to others, we should trust its information. However, if a node often makes mistakes or gives incorrect information deliberately, our trust in information provided by that node is reduced. Similar to reputation score, a reference score of a node is stored at its parent. So now, as illustrated in Figure 3, before each transaction, a node y should find not only x's reputation score, which is stored at z, the parent of x, but also z's reference score, which is stored at t, the parent of z and after each transaction, y updates scores for both x and z.

The problem now is how to evaluate correctness of information received from z to give feedback of a score after a transaction. Here, we propose a simple solution as follows. When a node is asked about reputation of its children, in addition to giving the total reputation score, it also gives the standard variation of the scores calculated from previous transactions. As a result, the correctness of received information is evaluated by both the reputation score and the standard variation. For example, if the result of the transaction falls far away outside the standard variation, the node giving information should be rated with a bad reference score.

That is not all. Assume that in the worst case, x, z, and t are all malicious peers and they cooperate with each other. If t gives a wrong reference score for z while z gives a wrong reputation score for x, y would still be cheated. To further enhance security, y can also ask reference score of t from its parent. In general y asks for reference scores of a chain of k ancestors of x in which k is a configurable parameter

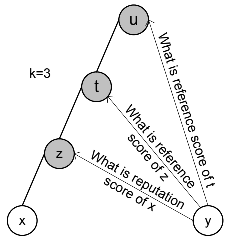

u
k=3
t
z
x
y
What is reference score of t
What is reference score of z
What is reputation score of x

Figure 3. A k=3 reference chain

of the system. Note that since these k nodes form a chain, the cost of lookup algorithm and update algorithm is just $logN + k$. By setting k with a large number, the system becomes strong against collaborative malicious peers. A worry is that k may have to be large, and hence it may be costly. However, since nodes in the system cannot determine the location of them in the tree structure, they have to follow the join algorithm, which scatters nodes along the system to make the tree balanced. As a result, forming a long chain of malicious peers connected by parent-child links is not easy. An example of a $k = 3$ reference chain is shown in Figure 3 in which y asks z for reputation score of x, t for reference score of z and u for reference score of t.

Another technique which can be used by a group of malicious nodes to trick other nodes is to create fake transactions and report good results to their parent to increase their reputation score. To avoid this problem, we just use a simple technique in score calculation as follows. First, we do not simply consider the number of successful transactions as the score. Instead, we limit the score at a maximum value, and the score of a node can only reach that maximum score. Second, we calculate not only the number of successful transactions but also the number of *different* successful transactions of nodes. By "*different*", we mean that transactions of the node that are done with different nodes. As a result, even though a node may have many good transactions with a specific node, it still has a low score if it has many other bad transactions with other nodes.

V. System Design

At this point, we are able to describe in greater detail how to extend SPP to use in BATON, an existing tree based framework. In essence, we try to place our proposed trust management layer on top of an existing networking framework that provides the topology of a tree. The challenge here is to ensure that we can effectively and efficiently implement SPP. In this section, we will first describe the structure of BATON. After that, we introduce the way to deploy SPP on it.

A. BATON

In BATON, each peer participating in the network is responsible for a node in the tree structure. The position of a node in the tree is determined by a pair of a *level* and a *number*. The level specifies the distance from the node

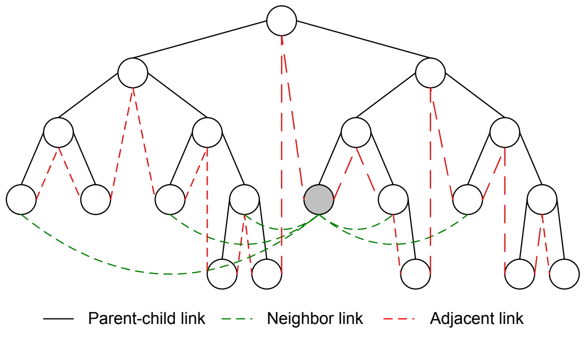


Figure 4. BATON structure

to the root while the number specifies the position of the node within the level. BATON uses three kinds of links to make connections between nodes: parent-child links are used to connect children and parents; adjacent links are used to connect adjacent nodes; and neighbor links are used to connect neighbor nodes at the same level having a distance 2^i from each other. Neighbor links are kept in two special sideways routing tables: left routing table and right routing table. The main purpose of neighbor links is to allow a flexible way to forward queries between nodes in the tree structure without going to the root, and hence BATON can avoid the bottleneck problem as well as single point of failure at the root node. An example of a BATON tree is shown in Figure 4. Note that in this figure, only neighbor links of the grey node are shown.

B. SPP Deployment

Since the most important issues in deploying SPP are how reputation of a node is looked up and how transaction results are reported to responsible nodes, we focus our discussion of these issues.

- Reputation lookup: before each transaction, nodes exchange information about their location in the tree to each other. Knowing the location of a node x, its partner y can infer the location of x's parent, which is z as below:

$$z_{level} = x_{level} - 1$$

$$z_{num} = \begin{cases} x_{num}/2 & \text{if } x_{num} \text{ is even} \\ (x_{num}+1)/2 & \text{if } x_{num} \text{ is odd} \end{cases}$$

  Note that in the tree structure, the level is setup increasingly from the root to the leaf starting at 0 while the number is assigned from the left to the right of each level starting at 1. Now, knowing the location of z, y can issue a query to lookup x's reputation towards z. The algorithm of sending a query towards a node knowing its location is represented as in Algorithm 1. Since at each step, this algorithm makes the search space reduce by half, it is guaranteed that after maximum $O(logN)$ steps, the query should reach the destination node z. When z receives the query, it returns the reputation score of x to y. Note that if x does not tell a truth about its location, and hence when y issues the query either z can not be found or z is not the parent of x. As a result, x can be considered as a bad node.

Algorithm 1 :Query (level l, number n, node z)

```
$l_{node}$ = level of the current node
$n_{node}$ = number of the current node
if $l_{node} = l$ then
  t = the nearest node to z
  t.Query(l, n, z)
else
  if $l_{node} > l$ then
    t = a child of the current node
    t.Query(l, n, z)
  else $\{l_{node} < l\}$
    t = parent of the current node
    t.Query(l, n, z)
```

- Transaction result report: after each transaction, a similar process is done to report the result of the transaction between partners to their parent. In particular, each peer rates the transaction by giving its partner a score in a range of [-1.0, 1.0]. Depending on the level of satisfaction or dissatisfaction, a value is given in which a positive score is used to indicate a good transaction while a negative score indicates a bad transaction.

VI. Experimental Study

To evaluate the performance of our proposal, we have implemented an extension of BATON [7] to support our security protocol. We tested our system in a network of 1,000 nodes, where exists two kinds of nodes: good nodes and malicious nodes. We just make a simple assumption that that good nodes always do good transactions and give correct answers if they are asked for reputation of their children. On the other hand, malicious nodes always do bad transactions and give incorrect answers about reputation of their children.

A. Effect of Varying Number of Malicious Nodes

We first evaluate the effect of varying number of malicious nodes on the strength of the system. The result is displayed in Figure 5 in which the x-axis presents the percentage of bad nodes in the system while the y-axis presents the percentage of correct answers about reputation of nodes. The length of reference chain in this experiment is fixed at 3. The result shows that our system can suffer up to 20% of malicious nodes while still provide good answers for a reputation of nodes: more than 80% of answers is correct. It is because in order to fully cheat other nodes, malicious nodes have to form a subtree height greater than 3. However, it is difficult to do that since nodes are distributed equally in the leaf level to keep the tree balance.

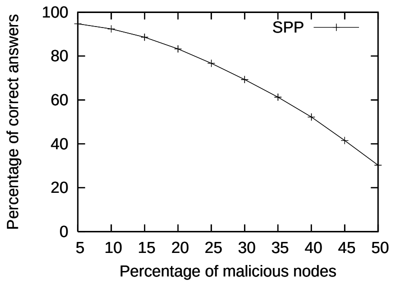


Figure 5. Effect of varying number of malicious nodes

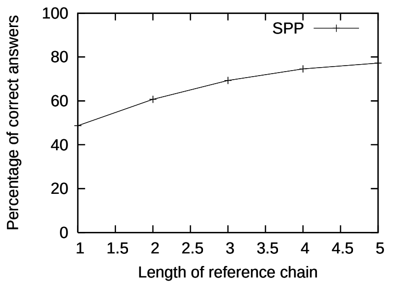


Figure 6. Effect of varying length of reference chain

B. Effect of Varying Length of Reference Chain

In this section, we vary length of reference chain from 1 to 5 while keeping the percentage of malicious nodes at 30%. The result is displayed in Figure 6. The result confirms that the system increases its strength with the increasing length of reference chain.

VII. Conclusion

In conclusion, in this paper, we proposed SPP, a general secure protocol for reputation management in peer-to-peer systems based on a tree structure. By using a tree structure, SPP can avoid the high cost of broadcasting messages that is seen in gossiping-based solutions. At the same time, SPP does not suffer the problem of bottlenecks and single points of failure as seen in server-based solutions through the employment of extra links in the tree structure. We came up with a specific tree structure extended from BATON [7] to implement SPP. Finally, we conducted experiments to evaluate the effectiveness and efficiency of SPP, and presented the above positive results.

References

[1] eBay, "http://www.ebay.com."

[2] Amazon Auctions, "http://auctions.amazon.com."

[3] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "Eigenrep: Reputation management in p2p networks," in *Proceedings of the 12th WWW Conference*, 2003.

[4] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in nice," in *Proceedings of the 2003 Infocom Conference*, 2003.

[5] B. Dragovic, B. Kotsovinos, S. Hand, and P. R. Pietzuch, "Xenotrust: Event-based distributed trust management," in *Proceedings of the 2nd International Workshop on Trust and Privacy in Digital Business*, 2003.

[6] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, no. 7, pp. 843–857, 2004.

[7] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "Baton: A balanced tree structure for peer-to-peer networks," in *Proceedings of the 31st VLDB Conference*, 2005, pp. 661–672.

[8] International Telegraph and Telephone Consultative Committee (CCITT), *The Directory - Authentication Framework, Recommendation X. 509*, 1993 update.

[9] P. Zimmermann, *PGP Users Guide*. MIT Press, 1994.

[10] M. Blaze and J.Feigenbaum, "Decentralized trust management," in *IEEE Symposium on Security and Privacy*, 1996.

[11] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "REFEREE: Trust management for Web applications," *Computer Networks and ISDN Systems*, vol. 29, no. 8–13, pp. 953–964, 1997.

[12] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, *The KeyNote Trust Management System, Version 2*. RFC-2704. IETF, 1999.

[13] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2001.

[14] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a p2p network," in *Proceedings of the 11th WWW Conference*, 2002.

[15] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 2002 ACM Conference on Computer and Communication Security*, 2002.

[16] J. Pujol and R. Sanguesa, "Extracting reputation in multi agent systems by means of social network topology," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[17] J. Sabater and C. Sierra, "Regret: A reputation model for gregarious societies," in *Proceedings of the 4th Workshop on Deception, Fraud and Trust in Agetn Societies*, 2001.

[18] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 205–217, 2002.

[19] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram, "Querying peer-to-peer networks using P-Trees," in *Proceedings of the 7th WebDB*, 2004, pp. 25–30.

Fast and Automatic Verification of Authentication and Key Exchange Protocols

Haruki Ota, Shinsaku Kiyomoto and Toshiaki Tanaka
Information Security Laboratory
KDDI R&D Laboratories, Inc.
Saitama, Japan
{*haruki, kiyomoto, toshi*}*@kddilabs.jp*

***Abstract*—Security in a peer-to-peer (P2P) system is not considered, although it has many potential threats. In order to make the whole P2P system secure, various security functions require to be taken into consideration for these threats, respectively. We take up authentication and key exchange protocols as a target of the security functions in the P2P system. These protocols can bear one duty in order to realize the secure P2P system. It is preferable for authentication and key exchange protocols to be verified automatically and rapidly in accordance with security requirements. In order to meet this requirement, we proposed the security verification method for the aforementioned protocols based on Bellare et al.'s model and showed the verification points of security properties to verify their security efficiently. However, there are three weaknesses in the aforementioned paper. In this paper, (1) we describe the relations of the six verification points, (2) explain how the proposed method verifies the aforementioned protocols by providing one example and (3) show the validity of the proposed method by verifying the security of 87 authentication and key exchange protocols that were generated automatically.**

***Keywords*-security verification method; authentication and key exchange protocols; verification points;**

I. Introduction

A. Motivation

Recently, a peer-to-peer (P2P) system, which is one of the distributed network architectures, has been developed. However, the security in the P2P system is not considered, although it has potential threats, such as viruses and worms, illegal uses of data with copyrights, impersonation, privacy issues and unauthorized access. For these threats, the security functions, such as virus detection techniques, digital rights management, cryptographic protocols, privacy protection mechanisms and intrusion detection techniques, require to be taken into consideration, respectively, in order to make the whole P2P system secure. Then, we take up the cryptographic protocols, in particular, authentication and key exchange protocols as a target of the security functions in the P2P system. In the authentication protocol, pair of users can communicate with each other, while each user knows who his/her communication partner is. In the key exchange protocol, they can send and receive secret data over an unreliable channel. Therefore, these protocols can bear one duty in order to realize the secure P2P system.

For a considerable period, existing authentication and key exchange protocols were designed by trial and error, based on the designer's understanding of security and cryptographic techniques. Therefore, it is vital to be able to deal with compromised protocols quickly. However, the process of specialists designing authentication and key exchange protocols is a time-consuming one and designing a new protocol or modifying an existing protocol and verifying its security are a lengthy process. As a result, there were neither the methods to evaluate the authentication and key exchange protocols formally nor the mechanisms to deal with compromised protocols quickly.

B. Related Work

Two different types of methods have been proposed as ways of verifying the security of authentication and key exchange protocols: those based on a computational complexity approach and those based on formal verification. As methods based on the computational complexity approach, Bellare, Pointcheval and Rogaway introduced the first indistinguishability-based formal model of security for authentication and key exchange protocols [1], [2], [3]. Specifically, Bellare and Rogaway first proposed 2-party mutual authentication and authenticated key exchange protocols in 1993 [1], and subsequently extended this to a 3-party setting via the key distribution center with respect to key exchange protocols in 1995 [2]. In 2000, Bellare, Pointcheval and Rogaway proposed provably secure password-based key exchange and authenticated key exchange protocols, based on the Bellare-Rogaway model [3]. Bellare et al. formulated models that were secure against an off-line dictionary attack and forward secrecy. Hereinafter, we call the model proposed in [1], [2], [3] the "BPR model". The BPR model became the basis of a considerable number of subsequent research studies in this area, such as those that investigated a simulation paradigm [4] and a universally composable framework [5]. However, the problem remained that the security of the protocols still needed to be proved. That is, there was not the automatic verification method based on the BPR model since it is very difficult to implement the notations of the provable security in the BPR model.

On the other hand, methods based on formal verification are classified into the following: methods based on state machine approaches (e.g., the Dolev-Yao model [6]), methods using model checkers (e.g., FDR (Failures Divergences Refinement)/CSP (Communicating Sequential Processes) [7]), methods using algebraic systems (e.g., spi calculus [8]), methods based on modal logic (e.g., BAN (Burrows-Abadi-Needham) logic [9]) and methods based on inductive approaches (e.g., Isabelle/HOL (Higher Order Logic) [10]). However, these methods are less than optimal as it takes a considerable amount of time to verify the security of protocols and/or they cannot always verify the security of protocols automatically.

In order to resolve the aforementioned problems, we proposed a security verification method for authentication and key exchange protocols based on the BPR model [11], [12], [13]. We generalized the process of the security proofs based on the BPR model to implement it as a tool. In particular, we showed the complete verification points of security properties for authentication and key exchange protocols so that the security of each protocol could be verified rapidly and automatically [13]. The verification points have the characteristic that the authentication and key exchange protocols are determined to be secure if they are satisfied with at least one verification point item of the security property. However, there are the following three weaknesses in [13].

1) The relations of the verification points are not clear.
2) It is not clear how the proposed method verifies the authentication and key exchange protocols.
3) The verification results for concrete protocols are not shown using the proposed method.

C. Contributions

In this paper, we provide the following contributions in order to improve the aforementioned weaknesses.

1) We describe the relations of the six verification points by considering the attack models and the security targets.
2) We explain how the proposed method verifies the aforementioned protocols by providing one verification example, which is satisfied with the six security properties.
3) We show the validity of the proposed method by verifying the security of the concrete authentication and key exchange protocols and confirming the verification results and verification time.

D. Organization

The rest of this paper is organized as follows. We introduce the BPR model in Section II. We review the proposed security verification method for authentication and key exchange protocols and describe the relations of the verification points in Section III. We explain the verification example and the verification results using the proposed method in Section IV. Our conclusions are presented in Section V and we present detailed tables of the verification points for the aforementioned protocols in Appendix.

II. BPR Model

This section introduces the security properties of the authentication and key exchange protocols in the BPR model.

In the BPR model, Bellare et al. introduced new notions of security: “matching conversation” of the authentication protocol and “semantic security” of the key exchange protocol [1]. They formulated the following security properties from real attacks, which are shown in brackets, for each notion in accordance with the security requirements.

- Matching conversation (MC) [1]
  In an authentication protocol, an adversary cannot alter messages, send other messages, intercept messages or deliver messages out of order.
  - Security against an impersonation attack (MC-SIA) [1]
    An adversary cannot break an authentication protocol even when he/she controls all communications between parties. [Impersonation attack]
- Semantic security (SS) [1]
  In a key exchange protocol, an adversary cannot distinguish between the session key and random session key.
  - Security against a passive attack (SS-SPA) [1], [2]
    An adversary cannot break a key exchange protocol even when he/she eavesdrops on all communications between parties. [Eavesdropping attack]
  - Security against an active attack (SS-SAA) [1], [2]
    An adversary cannot break a key exchange protocol even when he/she controls all communications between parties. [Active attack (e.g., replay attack, man-in-the-middle attack and so on)]
  - Known key security (SS-KKS) [1], [2]
    An adversary cannot obtain a target session key even when he/she obtains session keys in other sessions. [Known key attack]
  - Weak forward secrecy (SS-WFS) [2], [3]
    An adversary cannot obtain the past session key even when he/she obtains long-lived keys such as the secret keys used in secret key encryption, passwords or private keys used in public key encryption. [Corruption attack]
- Common item
  - Resistance to an off-line dictionary attack (RODA) [3]
    An adversary cannot search for a password of a party that corresponds to the recorded communication off-line from the dictionary. [Off-line dictionary attack]

III. Security Verification Method

This section reviews the proposed security verification method for authentication and key exchange protocols based on the BPR model.

A. Procedure

This subsection describes the procedure of the proposed method.

The verification program (VP) verifies the security of the authentication and key exchange protocols in the following manner.

1) The VP enumerates all cryptographic primitives and data used in the authentication and key exchange protocols. Principal cryptographic primitives are classified as functions that are equivalent to the following definitions.
   - Secret key encryption (SKE)
     Function for the purpose of encryption using the pre-shared key.
   - Encryption using password (EPW)
     Function for the purpose of encryption using the pre-shared password.
   - Public key encryption (PKE)
     Function for the purpose of encryption using the public key.
   - Diffie-Hellman family (DH)
     Function for the purpose of key exchange using the Diffie-Hellman method.
   - Digital signature scheme (SIG)
     Function for the purpose of generating the signature using the signing key.
   - Hash function (HF)
     Function for the purpose of generating the digest without using the pre-shared key.
   - Message authentication code scheme (MAC)
     Function for the purpose of generating the digest using the pre-shared key.
2) The VP sets up the following roles among the cryptographic primitives enumerated in step 1 in the authentication and key exchange protocols.
   - Cryptographic primitives required for authenticator generation in the authentication protocol (PAG).
   - Cryptographic primitives required for key generation in the key exchange protocol (PKG).
   - Cryptographic primitives that appear in flows and include the password (PAF).
   - Cryptographic primitives included in the arguments of other cryptographic primitives (PAO).
   - Cryptographic primitives that are not PAG, PKG, PAF or PAO (PNA).

   Here, we define a framework as $g(f(A, B), C)$ with respect to the aforementioned roles without loss of generality. f and g denote the aforementioned roles and A, B and C denote the values of the cryptographic primitives or data enumerated in step 1, where other arguments of f and g that are not related to the verification are ignored. In this case, the combinations of f and g are as follows.
   - g is the PNA and f is the PAG, PKG or PAF, namely, $f(A, B)$.
   - g is the PAG, PKG or PAF and f is the PAO, namely, $g(f(A, B), C)$.

   There are no other variants, since the VP sets up not only the data but also the values of the cryptographic primitives, as described in step 3.
3) The VP sets up the following elements in respect of the values of the cryptographic primitives and data enumerated in step 1 in accordance with the protocol specifications.
   - Data types
     – General data (GD)
     – Identity data (ID)
     – Temporary data (TD)
     – Long-lived key (LLK)
     – Password (PW)
   - Values types
     – Fixed value (FV)
     – Temporary value (TV)
   - Values and data states
     – Public state (PS)
     – Secret state (SS)
4) The VP sets up the security properties defined in Section II according to the user's requirement for the authentication and key exchange protocols.
5) The VP checks the verification points shown in Appendix, using the elements of step 3 for the security properties of step 4 in the authentication and key exchange protocols. If the authentication and key exchange protocols are satisfied with at least one verification point item of the security property, then the VP determines that these protocols are secure against this security property. Then, the VP sets up these elements and security properties in accordance with the order of the protocol flows for the values of the cryptographic primitives and data that are related to each attack. Here, the values and data states are renewed, where public states are given priority over secret states.

We provide the verification example of the proposed method in Section IV-A in order to show how to verify a protocol.

B. Relations of Verification Points

This subsection describes the relations of the six verification points.

We explain the relations of the six verification points. The VP sets up the data that are related to each attack in the proposed method. Table I denotes the corresponding data and the combinations of f and g. The security properties are roughly classified into three as can be seen from the combinations of f and g in Table I: MC-SIA, SS group (SS-SPA, SS-SAA, SS-KKS and SS-WFS) and RODA. MC-SIA and RODA are independent of the other security properties since the former's target is the authenticator and the latter's target is the flows that include the password, respectively.

On the other hand, there are some relations in the SS group since it has the same target as the key generation function. First, SS-SPA is the weakest security level in the SS group, that is, SS-SPA has the most verification point items. The verification points of the remaining SS-SAA, SS-KKS and SS-WFS are derived from that of SS-SPA. Second, SS-SAA implies SS-SPA from the security properties, that is, the verification point of SS-SPA completely includes that of SS-SAA. Third, the known key attack is equivalent to the active attack, except that the adversary can obtain session keys in other sessions. The verification point of SS-KKS is the same as that of SS-SAA since the data and values with respect to the session keys in other sessions are only set up in accordance with the known key attack. Finally, the adversary can obtain the long-lived keys in the corruption attack, which is different from the eavesdropping attack. That is, the long-lived keys in the verification point of SS-SPA are modified into the public state from the secret state in the verification point of SS-WFS. Then, the inapplicable items in the verification point of SS-WFS need to be deleted.

We show the verification points of MC-SIA, SS-SPA, SS-SAA, SS-KKS, SS-WFS and RODA in Tables III – VII of Appendix. See [13] with respect to how to derive the verification point of each security property.

IV. Evaluation

This section shows the evaluation of the method proposed in Section III.

A. Verification Example

This subsection shows the verification example of the proposed method.

Table I
Setup data and combinations of f and g for each security property.

	MC-SIA		SS-SPA		SS-SAA	
Data	All flows		All flows		All flows	
g	PNA	PAG	PNA	PKG	PNA	PKG
f	PAG	PAO	PKG	PAO	PKG	PAO
	SS-KKS		SS-WFS		RODA	
Data	All flows Other session keys		All flows Long-lived keys		All flows	
g	PNA	PKG	PNA	PKG	PNA	PAF
f	PKG	PAO	PKG	PAO	PAF	PAO

We verify the security of the authenticated key exchange protocol using the proposed method as the example. This protocol, which is satisfied with the six security properties: MC-SIA, SS-SPA, SS-SAA, SS-KKS, SS-WFS and RODA, is one of the authenticated key exchange protocols that were automatically generated using an automatic generation technique [14], as described in Section IV-B. Figure 1 shows the protocol flow. Parties P_1 and P_2 share a password pw beforehand. The party P_1 generates a random number x and sends $\mathcal{E}_{pw}(g^x)$ to the party P_2, where $\mathcal{E}_{pw}$ is the encryption using the password pw and g^x is the Diffie-Hellman-based public value. The party P_2 generates a random number y and sends $\mathcal{E}_{pw}(H(g^x \parallel g^y) \parallel g^y) \parallel H(g^x)$ to the party P_1, where H is the hash function and g^y is the Diffie-Hellman-based public value. The party P_1 sends $H(g^y)$ to the party P_2. Finally, the parties P_1 and P_2 share a session key $sk = H(g^{xy})$.

Then, the roles of cryptographic primitives and types and states of data and values are set up for this protocol as items 1 and 2, respectively. Note that the states of data and values are different for the case of SS-WFS and cases other than SS-WFS in item 2. Also, the VP determines that this protocol is secure against each security property, since f and g take the corresponding cryptographic primitives and A, B and C take the corresponding values of the cryptographic primitives and data for the framework $g(f(A, B), C)$ in item 3, where "null" denotes empty.

1) Roles of cryptographic primitives:
   - Cryptographic primitives $= \{g_1, g_2, g_3, g_4, H_1, H_2, H_3, H_4, H_5, E_1, E_2\}$
     - $g_1 = g^x$ [DH], $g_2 = g^y$ [DH]
     - $g_3 = (g^y)^x$ [DH], $g_4 = (g^x)^y$ [DH]
     - $H_1 = H(g_1 \parallel g_2)$ [HF]
     - $H_2 = H(g_3)$ [HF], $H_3 = H(g_4)$ [HF]
     - $H_4 = H(g_1)$ [HF], $H_5 = H(g_2)$ [HF]
     - $E_1 = \mathcal{E}_{pw}(g_1)$ [EPW]
     - $E_2 = \mathcal{E}_{pw}(H_1 \parallel g_2)$ [EPW]

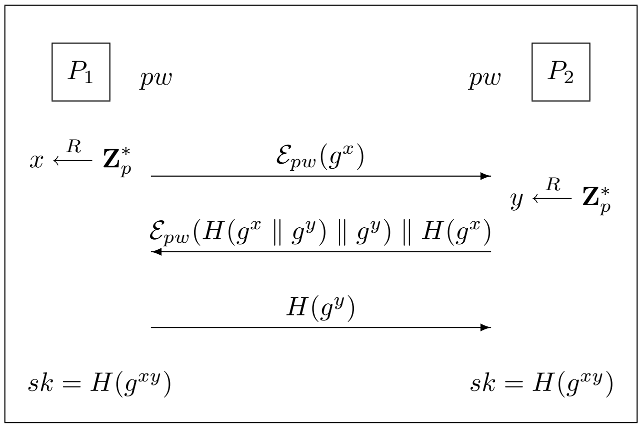

P_1
pw
pw
P_2
$x \xleftarrow{R} \mathbf{Z}_p^*$
$\mathcal{E}_{pw}(g^x)$
$y \xleftarrow{R} \mathbf{Z}_p^*$
$\mathcal{E}_{pw}(H(g^x \parallel g^y) \parallel g^y) \parallel H(g^x)$
$H(g^y)$
$sk = H(g^{xy})$
$sk = H(g^{xy})$

Figure 1. Protocol example, which is satisfied with the six security properties: MC-SIA, SS-SPA, SS-SAA, SS-KKS, SS-WFS and RODA.

- PAG $= \{H_5\,(\text{of}\,P_1), H_4\,(\text{of}\,P_2)\}$
- PKG $= \{H_2\,(\text{of}\,P_1), H_3\,(\text{of}\,P_2)\}$
- PAO for PKG $= \{E_2\,(\text{for}\,H_2), E_1\,(\text{for}\,H_3)\}$
- PAF $= \{E_1, E_2\}$

2) Types and states of data and values:
   - Types and states $= \{pw, x, y, g_1, g_2, g_3, g_4, H_1, H_2, H_3, H_4, H_5, E_1, E_2\}$
     - $pw =$ PW-PS (when SS-WFS)
     - $pw =$ PW-SS (except for SS-WFS)
     - $x =$ TD-SS, $y =$ TD-SS
     - $g_1 =$ TV-PS (when SS-WFS)
     - $g_1 =$ TV-SS (except for SS-WFS)
     - $g_2 =$ TV-PS (when SS-WFS)
     - $g_2 =$ TV-SS (except for SS-WFS)
     - $g_3 =$ TV-SS, $g_4 =$ TV-SS
     - $H_1 =$ TV-PS (when SS-WFS)
     - $H_1 =$ TV-SS (except for SS-WFS)
     - $H_2 =$ TV-SS, $H_3 =$ TV-SS
     - $H_4 =$ TV-PS, $H_5 =$ TV-PS
     - $E_1 =$ TV-PS, $E_2 =$ TV-PS

3) Reasons that meet each security property:
   - MC-SIA
     - P_1: $\text{null}(H_5\,[\text{HF}](g_2\,[\text{TV-SS}], \text{null}), \text{null})$
     - P_2: $\text{null}(H_4\,[\text{HF}](g_1\,[\text{TV-SS}], \text{null}), \text{null})$
   - SS-SPA
     - P_1: $\text{null}(H_2\,[\text{HF}](g_3\,[\text{TV-SS}], \text{null}), \text{null})$
     - P_2: $\text{null}(H_3\,[\text{HF}](g_4\,[\text{TV-SS}], \text{null}), \text{null})$
   - SS-SAA and SS-KKS
     - P_1: $H_2\,[\text{HF}](E_2\,[\text{EPW}](pw\,[\text{PW-SS}], \text{null}), g_3\,[\text{TV-SS}])$
     - P_2: $H_3\,[\text{HF}](E_1\,[\text{EPW}](pw\,[\text{PW-SS}], \text{null}), g_4\,[\text{TV-SS}])$
   - SS-WFS
     - P_1: $H_2\,[\text{HF}](E_2\,[\text{EPW}](pw\,[\text{PW-PS}], \text{null}), g_3\,[\text{TV-SS}])$
     - P_2: $H_3\,[\text{HF}](E_1\,[\text{EPW}](pw\,[\text{PW-PS}], \text{null}), g_4\,[\text{TV-SS}])$
   - RODA
     - 1st flow: $\text{null}(E_1\,[\text{EPW}](pw\,[\text{PW-SS}], g_1\,[\text{TV-SS}]), \text{null})$
     - 2nd flow: $\text{null}(E_2\,[\text{EPW}](pw\,[\text{PW-SS}], H_1\,[\text{TV-SS}]), \text{null})$

We explain the verification process of P_1 in MC-SIA as an example. The VP sets up the items 1 and 2 by steps 1 $\sim$ 3 of the proposed method. Here, PAG of P_1 is $H_5 = H(g_2)$ and its PAO is null. Thus, the authenticator of P_1 has the form of "$\text{null}(H_5\,[\text{HF}](g_2\,[\text{TV-SS}], \text{null}), \text{null})$" for the framework $g(f(A, B), C)$, as described in item 3. In this case, the aforementioned form is satisfied with the item of the third row in Table III. That is, f is HF of PDH, A is TV-SS of T*-SS and g, B and C are null.

B. *Verification Results*

This subsection describes the verification results using the method proposed in Section III.

An automatic generation technique of the authentication and key exchange protocols was proposed in [14], in relation to this paper. In [14], eighty-seven types of authentication and key exchange protocols, which are composed of 15 authentication (Auth), 22 key exchange (KE) and 50 authenticated key exchange (AKE) protocols, were automatically generated using this automatic generation technique. In the automatic generation technique, the optimal protocol is generated automatically when the following items are set up.

- Types: Auth, KE and AKE
- Cryptographic algorithms: algorithms that correspond to SKE, EPW, PKE, DH, SIG, HF and MAC
- Security properties: MC-SIA, SS-SPA, SS-SAA, SS-KKS, SS-WFS and RODA
- The numbers of flows: 1, 2 and 3

Then, we verified the security of the aforementioned authentication, key exchange and authenticated key exchange protocols, using the proposed method. Table II shows the verification results, best, worst and average verification time, minimal, maximal and average protocol definition file size for the authentication, key exchange and authenticated key exchange protocols, respectively, where the unit of the verification time is the millisecond and the unit of the protocol definition file size is the kilobyte. Symbols "Y", "N" and "—" denote that the protocol "meets", "does not meet" and "does not require" the corresponding security property, respectively.

These results completely coincide with the security requirements for the automatically generated protocols. The verification time is within 110 [ms] in the 87 authentication and key exchange protocols, using a PC with an Intel Pentium 4 2.6-GHz processor and 2.0-Gbyte RAM. On the other hand, TRUST [15] takes 40 [ms] $\sim$ 1.8 [s] at the fastest among the methods based on formal verification [16]. We cannot make a precise comparison between the proposed method and the existing methods, since the performance of the PC and the verified protocols are different from ours. However, the proposed method can verify the security of each protocol automatically and more quickly than most existing methods, since our method takes 4.6 [ms] $\sim$ 110 [ms] from Table II. Furthermore, the size of the protocol definition file is within 14.2 [KB] in the aforementioned protocols and the program size is 1.25 [MB].

V. Conclusion

Various security functions require to be taken into consideration for many potential threats, respectively, in order to realize the secure P2P system. Then, we took up the authentication and key exchange protocols as a target of the security functions in the P2P system. These protocols can

Table II
VERIFICATION RESULTS IN AUTHENTICATION AND KEY EXCHANGE PROTOCOLS.

Types	MC	SS				RODA	Numbers	Verification Time [ms]			Protocol Definition File Size [KB]		
	SIA	SPA	SAA	KKS	WFS			Best	Worst	Average	Minimum	Maximum	Average
Auth	Y	—	—	—	—	—	13	4.648	9.256	6.597	6.34	8.42	7.18
	Y	—	—	—	—	Y	2	8.235	11.988	10.112	7.11	7.93	7.52
KE	—	Y	Y	N	N	—	3	8.948	16.451	11.590	4.79	5.66	5.09
	—	Y	Y	Y	N	—	5	12.352	15.091	13.071	5.51	6.28	5.82
	—	Y	Y	Y	Y	—	12	20.035	32.445	27.058	6.27	8.10	7.20
	—	Y	Y	Y	N	Y	1	23.424	23.424	23.424	6.36	6.36	6.36
	—	Y	Y	Y	Y	Y	1	39.138	39.138	39.138	7.69	7.69	7.69
AKE	Y	Y	Y	Y	N	—	20	30.215	67.539	40.519	7.27	9.68	8.39
	Y	Y	Y	Y	Y	—	28	41.864	109.054	73.821	8.23	14.20	10.72
	Y	Y	Y	Y	N	Y	1	64.928	64.928	64.928	8.77	8.77	8.77
	Y	Y	Y	Y	Y	Y	1	88.700	88.700	88.700	9.90	9.90	9.90

bear one duty in order to realize the secure P2P system. So far, we proposed the security verification method for the aforementioned protocols based on the BPR model and showed the verification points of security properties to verify their security efficiently.

In this paper, we described the relations of the six verification points and explained the verification example using the proposed method. We also verified the security of 87 authentication and key exchange protocols, which were generated automatically. Then, we confirmed that the verification time was within 110 [ms] and that the security properties of the verification results completely coincided with the security requirements for the aforementioned protocols.

REFERENCES

[1] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptology — CRYPTO'93*, vol. 773. Santa Barbara, CA: Springer-Verlag, Aug. 1993, pp. 232–249.

[2] M. Bellare and P. Rogaway, "Provably secure session key distribution — The three party case," in *Proc. 27th Annual ACM Symposium on Theory of Computing*. Philadelphia, PA: ACM Press, May/Jun. 1995, pp. 57–66.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology — EUROCRYPT 2000*, vol. 1807. Bruges, Belgium: Springer-Verlag, May 2000, pp. 139–155.

[4] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," in *Proc. 30th Annual ACM Symposium on the Theory of Computing*. Dallas, TX: ACM Press, May 1998, pp. 419–428.

[5] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*. Las Vegas, NV: IEEE Computer Society Press, Oct. 2001, pp. 136–145.

[6] D. Dolev and A. Yao, "On the security of public key protocols," in *Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1981)*. Nashville, TN: IEEE Computer Society Press, Oct. 1981, pp. 350–357.

[7] A. Roscoe, "Modelling and verifying key-exchange protocols using CSP and FDR," in *Proc. Eighth IEEE Computer Security Foundations Workshop*. County Kerry, Ireland: IEEE Computer Society Press, Mar. 1995, pp. 98–107.

[8] M. Abadi and A. Gordon, "A calculus for cryptographic protocols: The spi calculus," *Information and Computation*, vol. 148, no. 1, pp. 1–70, 1999.

[9] M. Burrows, A. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[10] L. Paulson, "Proving properties of security protocols by induction," Univ. of Cambridge, Cambridge, UK, Computer Laboratory Tech. Rep. 409, 1996.

[11] H. Ota, S. Kiyomoto, and T. Tanaka, "Security Verification for Authentication and Key Exchange Protocols," in *Proc. 2008 International Symposium on Information Theory and its Applications (ISITA 2008)*. Auckland, New Zealand: Computer Society Press, Dec. 2008, pp. 507–512.

[12] H. Ota, S. Kiyomoto, and T. Tanaka, "Security Verification for Authentication and Key Exchange Protocols," *International Journal of Computer Science and Network Security*, vol. 9, no. 3, pp. 1–11, 2009.

[13] H. Ota, S. Kiyomoto, and T. Tanaka, "Security verification for authentication and key exchange protocols, revisited," in *Proc. 2010 IEEE 24th IEEE International Conference on Advanced Information Networking and Applications Workshops*. Perth, Australia: IEEE Computer Society Press, Apr. 2010, pp. 226–233.

[14] S. Kiyomoto, H. Ota, and T. Tanaka, "Security protocol dynamic generation and modification mechanisms for ubiquitous services," in *Proc. 11th International Conference on Wireless Personal Multimedia Communications (WPMC'08)*, Lapland, Finland, 2008.

[15] R. Amadio, D. Lugiez, and V. Vancackere, “On the symbolic reduction of processes with cryptographic functions,” *Theoretical Computer Science*, vol. 290, no. 1, pp. 695–740, 2003.

[16] A. Bracciali, G. Baldi, G. Ferrari, and E. Tuosto, “A coordination-based methodology for security protocol verification,” in *Proc. 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP2004)*, vol. 121. Bologna, Italy: Elsevier Science, 2004, pp. 23–46.

Appendix

This appendix presents detailed tables of the verification points referred to in Section III-B.

Tables III – VII show the verification points of MC-SIA, SS-SPA, SS-SAA, SS-KKS, SS-WFS and RODA. Table IV shows the common verification point of SS-SPA, SS-SAA and SS-KKS and Table V shows the remaining verification point of SS-SPA, where the verification points of SS-SAA and SS-KKS coincide with Table IV. In addition, each abbreviation symbol denotes the following.

- ALL denotes SKE, EPW, PKE, DH, SIG, HF or MAC.
- 6-SIG denotes SKE, EPW, PKE, DH, HF or MAC.
- SSM denotes SKE, SIG or MAC.
- S/S denotes SKE or SIG.
- SM denotes SKE or MAC.
- EPDH denotes EPW, PKE, DH or HF.
- PDHM denotes PKE, DH, HF or MAC.
- PDH denotes PKE, DH or HF.
- T*-*S denotes TD-PS, TD-SS, TV-PS or TV-SS.
- T*-SS denotes TD-SS or TV-SS.
- T*-SS+ denotes TD-SS, TV-SS or FV with LLK-SS.
- EXC denotes elements except for PW-PS and PW-SS.

Table III
VERIFICATION POINTS OF MC-SIA.

g	f	A	B	C
—	SSM	LLK-SS	T*-*S	—
—	EPW	PW-SS	T*-*S	—
—	PDH	T*-SS	—	—
SSM	SSM	LLK-SS	T*-*S	LLK-SS
ALL	SSM	LLK-SS	—	T*-*S
SSM	EPW	PW-SS	T*-*S	LLK-SS
ALL	EPW	PW-SS	—	T*-*S
SSM	PDH	T*-*S	—	LLK-SS
EPW	SSM	LLK-SS	T*-*S	PW-SS
EPW	EPW	PW-SS	T*-*S	PW-SS
EPW	PDH	T*-*S	—	PW-SS
PDH	SSM	LLK-SS	T*-*S	—
PDH	EPW	PW-SS	T*-*S	—
PDH	PDH	T*-SS	—	—
PDH	PDH	—	—	T*-SS

Table IV
COMMON VERIFICATION POINTS OF SS-SPA, SS-SAA AND SS-KKS.

g	f	A	B	C
SM	SSM	LLK-SS	T*-*S	LLK-SS
6-SIG	SSM	LLK-SS	—	T*-*S
SSM	EPW	PW-SS	T*-*S	LLK-SS
6-SIG	EPW	PW-SS	—	T*-*S
SM	PDH	T*-*S	—	LLK-SS
EPW	SSM	LLK-SS	T*-*S	PW-SS
EPW	EPW	PW-SS	T*-*S	PW-SS
EPW	PDH	T*-*S	—	PW-SS
PDH	SSM	LLK-SS	T*-*S	—
PDH	EPW	PW-SS	T*-*S	—
SIG	SM	LLK-SS	T*-*S	LLK-SS

Table V
REMAINING VERIFICATION POINTS OF SS-SPA.

g	f	A	B	C
—	SM	LLK-SS	T*-*S	—
—	EPW	PW-SS	T*-*S	—
—	PDH	T*-SS	—	—
EPW	PDH	—	—	T*-*S
PDH	PDH	T*-SS	—	—
PDH	PDH	—	—	T*-SS
SIG	PDH	T*-SS	—	LLK-SS

Table VI
VERIFICATION POINTS OF SS-WFS.

g	f	A	B	C
SSM	PDH	T*-SS	—	LLK-PS
S/S	MAC	LLK-PS	T*-SS	LLK-PS
EPW	PDH	T*-SS	—	PW-PS
EPW	MAC	LLK-PS	T*-SS	PW-PS
PDHM	SSM	LLK-PS	—	T*-SS
PDHM	EPW	PW-PS	—	T*-SS
PDH	PDH	T*-SS	—	—
PDHM	PDH	—	—	T*-SS
PDH	MAC	LLK-PS	T*-SS	—
MAC	SSM	LLK-PS	T*-SS	LLK-PS
MAC	EPW	PW-PS	T*-SS	LLK-PS

Table VII
VERIFICATION POINTS OF RODA.

g	f	A	B	C
—	SM	PW-SS	LLK-SS	—
—	EPDH	PW-SS	T*-SS+	—
SM	SSM	PW-SS	LLK-SS	LLK-SS
SM	EPDH	PW-SS	—	LLK-SS
EPW	SSM	PW-SS	LLK-SS	PW-SS
EPW	EPDH	PW-SS	T*-SS+	PW-SS
EPDH	EPDH	PW-SS	—	T*-SS+
PDH	SSM	PW-SS	LLK-SS	—
PDH	EPDH	PW-SS	T*-SS+	—
SIG	SM	PW-SS	LLK-SS	EXC
SIG	EPDH	PW-SS	T*-SS+	EXC

Anonymous Access to Trust Information Using k-anonymity Chord

Ahmet Burak Can
Department of Computer Engineering
Hacettepe University
06800 Ankara/Turkey
abc@hacettepe.edu.tr

Bharat Bhargava
Department of Computer Science
Purdue University
West Lafayette, IN 47907 USA
bb@cs.purdue.edu

***Abstract*—In a reputation based trust network, each peer stores trust information of others and answers the trust queries, in addition to providing services to others. We present a cryptographic protocol on Chord, which provides anonymous access to trust information. Peers form anonymity groups and generate responses inside the group. Responder of a trust query has k-anonymity protection against an adversary who can sniff all communication on the network. Moreover, our encryption scheme ensures that the initiator of a trust query can check the validity of an anonymous reply.**

***Keywords*-anonymity, trust management, peer-to-peer**

I. Introduction

Trust is fundamental to achieve collaborative tasks in a peer-to-peer system. When organizing, sharing, and searching resources, trustworthy peers are identified based on the trust information. Due to the large scale and distributed nature of peer-to-peer systems, a centralized entity cannot track a massive number of peer interactions and manage all trust information. Consequently, the burden should be shared among the peers, whereby each peer can become a *trust holder* [1], [2]. A trust holder needs anonymity to protect itself against malicious peers. Such protection motivates a peer to perform the trust holding duty and may prevent denial of service attacks, making information more available.

In peer-to-peer systems, several methods are studied to protect anonymity: limitations on routing information exchange [3], probabilistic random path building [4], [5], and flooding [6], [7], [8]. These methods are vulnerable to global passive adversaries who can sniff all the communication on the network. Mix networks [9] or onion routers [10] might be adapted for peer-to-peer systems. Trusted mix nodes encrypt and shuffle the network traffic so a global passive adversary can not determine who is communicating with whom. In an ideal solution, peers should organize themselves to protect anonymity and should not depend on trusted nodes. This is more adequate for the decentralized nature of peer-to-peer systems.

We propose k-anonymity Chord [11] to protect the anonymity of a trust holder when responding to trust queries. As in most peer-to-peer systems [12], we assume the existence of a bootstrap peer, which is a connection point to the network. Peers register their pseudonyms and encryption keys to the bootstrap peer when joining the network for the first time. A new peer obtains some certificates during the registration and then, joins two overlay networks: *service* and *trust* networks. The service network can be any network substrate, e.g., Gnutella[12], Freenet [6]. In a service request, a peer queries the service network to find a particular service such as a file. Several service providers respond to the query and send back their certificates to the requester. For each service provider, the requester sends trust queries to the trust network. This network must overlay on k-anonymity Chord, which runs the oblivious reply protocol to protect anonymity of trust holders. Peers form anonymity groups of size k. Each peer in an anonymity group sends back a trust reply after receiving a trust query. A peer's reply can not be distinguished from the replies of others. Thus, the real responder has k-anonymity protection against global passive adversaries. The requester can check the authenticity of trust replies to identify fake replies of malicious peers.

Section II explains the related research. Section III presents the encryption architecture, peer registration, and communication during service and trust queries. Section IV introduces k-anonymity Chord and the oblivious reply protocol. Section V gives a discussion about performance considerations and other issues. Section VI outlines the conclusions and results of our work.

II. Related Work

Various methods have been studied to protect anonymity in computer networks. We outline some of the prominent methods as follows.

Mix networks and onion routers. Mix networks are first proposed by Chaum [9] to protect anonymity of communicating parties for delay tolerant applications. Trusted mix nodes use cover traffic to shuffle messages so an adversary can not determine who is communicating with whom. Onion routers [10] form an overlay network to build anonymous, bi-directional virtual circuits for real-time communication. While mix networks are designed for delay tolerant applications, e.g., e-mail systems, onion routing is more feasible for real-time applications such as HTTP. Tor [13] extends onion routing with forward secrecy, congestion control, integrity checking and configurable exit policies. Our approach aims to protect anonymity without relying on a trusted mix nodes or onion routers so it should be considered in a different category of anonymity systems. In our approach, peers register themselves to a trusted peer but this peer does not participate in the anonymity protocol.

DC-Nets. Chaums dining cryptographer networks (DC-net) [14], provide unconditional sender anonymity in a group of participants. If the group size is N, this approach requires $O(N^2)$ message exchange for each message sending operation. Furthermore, before sending a message, $O(N^2)$ encryption keys should be distributed among N participants using a secure external method. This makes Chaums DC-net impractical for real life scenarios.

Buses. In the bus [15] approach, synchronous message tokens traverse in the network forever. When a peer receives a bus, it fills some seats with encrypted messages or dummy messages if it does not have any real message. When a bus arrives to a receiver, all or some related seats are decrypted to get the message. Ren et. al [16] applied this approach on peer-to-peer networks and circulated bus tokens on overlay rings. Although the bus approach can protect sender/receiver anonymity, the bus must traverse the network forever even the nodes do not have any real message.

Flooding. Freenet [6] and Freehaven [7] flood the storage requests in peer-to-peer storage systems to protect the requester and responder anonymity. Since no peer on the flooding path knows the whole path, it is hard to determine the requester and responder. Trustme [8], floods encrypted trust queries to the network and trust holders send back authenticated replies. Han and Liu [17] split a query into n shares and send the shares to neighbors in a peer-to-peer network. The peers who take t shares can decrypt and flood the query. The responder builds an onion path to the requester and sends the response on this path. MuON [18] uses a gossip protocol to reduce the traffic caused by query flooding. In all of these approaches, flooding protects anonymity if the adversary can not sniff the whole network. Additionally, excessive network traffic caused by flooding reduces the scalability of these systems.

Random path building. In Crowds [4], nodes form anonymity groups (crowds) and randomly forwards the requests in the crowd to protect the requester anonymity. Tarzan [5] establishes a random tunnel between a peer and an Internet server to protect the peers anonymity. Since none of the peers on a tunnel know the whole path, the initiator of a request can not be determined. MorhpMix [19] defines a peer-to-peer mix network where random mix nodes are selected during an anonymous communication. These approaches protect anonymity if the adversary can not sniff the whole communication path.

Changing the routing method. Anonymity has been studied on Chord by using recursive, randomized, indirect, split, bidirectional routing [20]. Achord [3] defines routing limitations on Chord to provide censorship resistance. These schemes offer anonymity protection in a local adversary model and can not protect anonymity against a global passive adversary.

III. Architecture

We assume the existence of a bootstrap peer (bp), which provides a connection point to the network for new peers. There might be multiple bootstrap peers to provide tolerance to failures and attacks. For simplicity of the notation, the rest of the paper considers one bootstrap peer, which is a basic certification authority for pseudonyms and encryption keys. It has a public/private key pair $\{U_{bp}, R_{bp}\}$. We assume all peers learn U_{bp} in a secure way, e.g., through a secure web site. A peer registers itself to the bootstrap peer when joining the network for the first time. During the registration, the bootstrap peer issues some certificates for the new peer.

P_i denotes the i^{th} peer. ID_i and TID_i are the pseudonyms of P_i in the service and trust networks respectively. While ID_i is selected by P_i before registration, TID_i is assigned by the bootstrap peer during the registration. ID_i and TID_i have no relation with each other. $\{U_i, R_i\}$ is P_i's public/private key pair for the service network operations. For the trust network, it has $\{TU_i, TR_i\}$ and $\{OU_i, OR_i\}$ key pairs. All key pairs are randomly selected by P_i and have no relation with each other. We assume that peers have good random number generators to prevent brute force guessing attacks.

If K is a public key or a symmetric encryption key, $K(M)$ stands for the encryption of M with key K for message confidentiality. When K is a private key, the operation is considered as signing of M. $H[M]$ is the hash digest of M. $X|Y$ denotes the concatenation of X and Y.

A. Adversary Model

An adversary tries to learn the pseudonym (TID) or IP number of a trust holder. It (we assume that an adversary is a peer so we will use ”it”) might have passive attack capability, e.g., sniffing the network communication. It may collaborate with some peers and launch attacks by coordinating with them. A local passive adversary can perform passive attacks only in a limited number of networks links. A global passive adversary can perform passive attacks on all links of a network. It has polynomial time computational capabilities and can not break cryptographic algorithms in polynomial time. Semi-honest adversary model [21] means that an adversary stays complaint with the protocols but may observe the network communication to obtain information.

B. Peer Registration

To demonstrate registration operation, we assume that P_i is joining the network for the first time and registering itself to the bootstrap peer as follows:

1) P_i sends $U_{bp}(ID_i|U_i|TU_i|OU_i|r_1)$ to the bootstrap peer as a registration request. r_1 is a random value

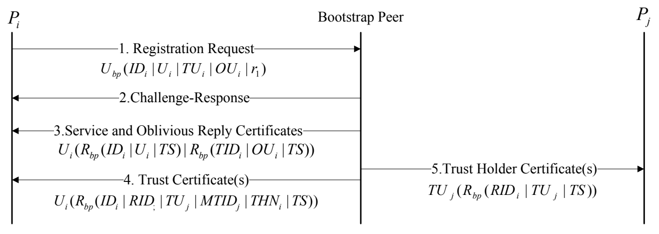

P_i
Bootstrap Peer
P_j
1. Registration Request
$U_{bp}(ID_i|U_i|TU_i|OU_i|r_1)$
2.Challenge-Response
3.Service and Oblivious Reply Certificates
$U_i(R_{bp}(ID_i|U_i|TS)|R_{bp}(TID_i|OU_i|TS))$
4. Trust Certificate(s)
$U_i(R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS))$
5.Trust Holder Certificate(s)
$TU_j(R_{bp}(RID_i|TU_j|TS))$

Figure 1. Registration of P_i to the bootstrap peer

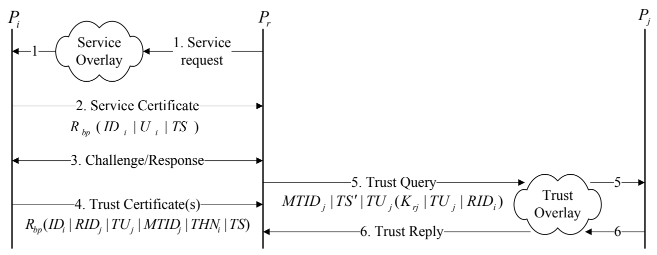


Figure 2. P_r is searching for a service provider (P_i) and then, querying its trust information

selected by P_i. Only the bootstrap peer can read the contents due to the encryption. It decrypts the request and stores ID_i, U_i, TU_i, OU_i for future accountability.

2) The bootstrap peer runs a challenge-response protocol to verify that P_i has U_i, TU_i, OU_i keys.
3) Assuming P_i passed step 2, the bootstrap peer selects TID_i value and sends back $U_i(R_{bp}(ID_i|U_i|TS)|R_{bp}(TID_i|OU_i|TS))$ to P_i. Only P_i can decrypt the message and check if TS value is same. $R_{bp}(ID_i|U_i|TS)$ is a *service certificate*. P_i sends this certificate to peers who request P_i's services. It proves P_i's registration to the service requester. $R_{bp}(TID_i|OU_i|TS)$ is an *oblivious reply certificate*, which is used during the oblivious reply operations explained in Section IV-C. It also informs P_i about its TID_i.
4) The bootstrap peer randomly selects P_i's trust holders. Let P_j be such a trust holder. The bootstrap peer sends $U_i(R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS))$ to P_i. The inner part, $R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS)$, is the *trust certificate*, which means that a peer associated with $MTID_j$ value and TU_j key will store P_i's trust information. $MTID_j$, explained in Section IV, is an anonymized value of TID_j and represents a range of peers instead of a particular peer. Using this certificate, P_i or another peer can send trust queries destined to P_j, but can not learn P_j's identity. RID_i is a random value to hide P_i's real identity from its trust holder, P_j. THN_i is the number of P_i's trust holders determined by the bootstrap peer.
5) The bootstrap peer sends a *trust holder certificate* to each trust holder. For example, P_j's trust holder certificate is $R_{bp}(RID_i|TU_j|TS)$. This certificate informs P_j about its trust holding duty on P_i's trust information. To protect P_i's anonymity, ID_i is not added to the certificate. P_j does not know P_i's identity but it can answer trust queries by using RID_i value.

A service certificate and related trust certificates expire according to TS field. The owner of an expired service certificate requests a new one from the bootstrap peer. Figure 1 briefly explains the peer registration operation.

C. Searching a Service Provider and Sending a Trust Query

Assume that P_i is a service provider, P_j is a trust holder of P_i, and P_r requests a service from P_i. Figure 2 shows the message exchanges during a service request and a trust query. In Step 1, P_r sends a query to the service network to find a service, e.g., a particular file. Assuming P_i has the service, it sends back a reply message containing its service certificate, $R_{bp}(ID_i|U_i|TS)$ (Step 2). P_r verifies the certificate using U_{bp} and runs a challenge/response protocol to authenticate P_i (Step 3). Then, P_i sends its trust certificates to P_r (Step 4). In our case, P_r receives only $R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS)$, which is the trust certificate for trust holder P_j. If ID_i, TS values match with the values from the service certificate, P_r ensures that P_j is a legitimate trust holder. However, P_r can not learn P_j's identity. If there are other trust holders, THN_i value informs P_r about the existence of other trust holders and forces P_i to send all certificates.

After verifying service and trust certificates, P_r sends a *trust query*, $MTID_j|TS'|TU_j(K_{rj}|TU_j|RID_i)$, to the trust network (Step 5). TS' is a time-stamp and unique among P_r's queries. K_{rj} is a random session key, which can only be learned by P_j due to encryption with TU_j. The encrypted part, $TU_j(K_{rj}|TU_j|RID_i)$, includes TU_j and RID_i fields to prevent forgery of the content. P_j checks these values and understands if the query is destined to itself. In Step 6, P_j sends back a *trust reply* message. The details of Step 5 and 6 will be explained in the next section.

IV. k-ANONYMITY CHORD

Chord [11] is a distributed hash table (DHT) designed for peer-to-peer networks. Chord's algorithm assigns each resource to a particular peer. We use Chord to access trust information efficiently. However, anonymity of a trust holder can not be protected on Chord when responding to a trust query. A responder can be identified since peers partially learn the network structure using Chord's finger tables. Additionally, a peer may learn more about an arbitrary part of the address space by sending excessive finger requests [3]. This makes guessing a responder easier without having global sniffing ability.

We propose the oblivious reply protocol on Chord to provide k-anonymity protection for trust holders. This means that a trust holder's identity can not be distinguished from k other peers when responding to trust queries. We call this DHT structure k-anonymity Chord, which performs peer join, leave, and finger table maintenance operations like a normal Chord ring. However, the search operation is modified to protect anonymity of the responder. In our case, the trust network overlays on a k-anonymity Chord ring. A peer joins the trust network with its TID value, e.g., P_j joins with TID_j. For the rest of the section, we assume that P_i is a service provider, P_j is a trust holder of P_i and P_r wants to get a service from P_i.

A. Formation of Anonymity Groups

As explained in Section III-C, P_r obtains P_i's service and trust certificates during its service request and prepares a trust query destined to P_j. However, P_r can not send the query directly to P_j since it does not know TID_j. It sends the query on the k-anonymity Chord by putting $MTID_j$ value to the query message. $MTID_j$ is an anonymized version of TID_j where the last m bits are set to zero. In a trust query, $MTID_j$ represents the range of pseudonyms between $MTID_j$ and $MTID_j + 2^m$. We call this range *search range* and the peers in the search range *target peers*. The bootstrap peer decides the value of m so that the expected number of target peers in a search range is equal to k. Since the bootstrap peer registers all peers, it can compute a precise m value. To explain $MTID_j$ selection, we give a numerical example.

Chord peers are located on a 2^n circular address space where n is the length of a TID. We assume that the bootstrap peer uniformly distributes peers on this address space. Suppose that $n = 32, k = 64$, $TID_j = 12345678H$ and there are 2^{16} peers in the network. Let X be an indicator random variable that represents if there is a peer on a particular location (When $X = 1$, there is a peer on that location). The probability of $X = 1$ is

$$P(x = 1) = \frac{2^{16}}{2^{32}} = \frac{1}{2^{16}}$$

and the expected number of nodes on a particular location is

$$E[X] = \sum_{x} x \cdot P(x) = 1 \cdot P(x = 1) + 0 \cdot P(x = 0) = \frac{1}{2^{16}}$$

Let Y be a random variable representing the number of peers that fall into a search range. The bootstrap peer selects a search range that has $Y \geq k = 64$ expected number of peers. Let S be the number of locations in a search range. Due to the uniformity of peer distribution, the expected number of peers in the search range is

$$E[Y] = E[X] \cdot S = \frac{1}{2^{16}} \cdot S \geq 64$$

The bootstrap peer finds that $S \geq 2^{24}$. This inequality suggests us that $m \geq \log_2 S = \log_2 2^{24} = 24$.

Then, the bootstrap peer computes $MTID_j$ as $MTID_j = 12345678H \wedge FF000000H = 12000000H$. This means that P_j has a TID_j value between $12000000H$ and $12FFFFFFH$. The expected number of peers in this range is 64 due to our selection.

B. Routing a Trust Query

Let $P_0, P_1 \ldots P_{k-1}$ be k target peers in $MTID_j$ and $MTID_j + 2^m$ range and P_j be one of the target peers. By this definition, P_0 is the owner of $MTID_j$ value. We define a two-phase routing method for trust queries. The first phase is a recursive Chord search to find P_0, the owner of $MTID_j$ value. P_r starts the first phase by preparing a trust query, $MTID_j|TS'|TU_j(K_{rj}|TU_j|RID_i)$, for P_j. It looks up its finger table, sends the query to the closest peer preceding P_0. The receiving peer forwards the query to another one by looking up $MTID_j$ value in its finger table. Forwarding operation continues until P_0 receives the query. TS' value gives a hint for the expiration time. Each forwarding peer caches the query to send the trust reply back to P_r.

After the query reaches to P_0, the second phase starts and the oblivious reply protocol runs to send the query to P_j and get its reply anonymously. The following section explains this protocol. In the attack scenarios, P_r tries to identify P_j by sniffing the network or obtaining collaborators. Note that, P_i may pretend to be P_r to learn P_j's identity.

C. Oblivious Reply

Oblivious reply is a cryptographic protocol to protect anonymity of a trust holder against a global passive adversary. This protocol is secure against collaborating passive adversaries in semi-honest adversary model [21]. The basic idea is that each target peer generates a separate trust reply. These replies can not be linked with the senders and P_j's reply can not be tracked during the operation of protocol. The protocol has several assumptions:

- Each target peer knows its search range and the other target peers in the search range. Additionally, each target peer knows its exact location in the range, i.e, the number of hops from P_0 and P_{k-1}.
- All target peers exchange their $R_{bp}(TID_i|OU_i|TS)$ certificates. Once the certificates are exchanged, they can be used in many trust queries.
- The public key encryption scheme ensures semantic security [22]. This implies that the result of an encryption depends on the message, key, and a sequence of coin tosses. Thus, encryption of a message with the same public key results in a different cipher text in each trial. However, the decryptions of these cipher texts give the same plain text.
- The public key encryption scheme is not commutative, which means that $A(B(M)) \neq B(A(M))$.

After P_0 receives P_r's query, each target peer forwards the query until P_{k-1} receives it. P_{k-1} tries to decrypt the contents of the query. If the decryption is successful, it prepares O_{k-2}^{k-1} as follows:

$$\begin{aligned}
O_{k-2}^{k-1} &= OU_{k-2}(O_{k-3}^{k-1}) \\
O_{k-3}^{k-1} &= OU_{k-3}(O_{k-4}^{k-1}) \\
&\dots \\
O_{1}^{k-1} &= OU_{1}(O_{0}^{k-1}) \\
O_{0}^{k-1} &= OU_{0}(K_{rj}(TV_i|RID_i|TS')|AB)
\end{aligned}$$

O_{k-2}^{k-1} denotes P_{k-1}'s *oblivious reply*, which is destined to P_{k-2}. The last field, AB, is the authenticity bit. It is set to 1 if the reply is authentic.

If the decryption fails, P_{k-1} generates a false oblivious reply. The innermost layer of O_{k-2}^{k-1} contains $K_{random}(RTV|RHID|TS')|AB$ as the content. K_{random} is a randomly generated key. RTV and $RHID$ are random trust and hash values respectively. These random values should have the same amount of bits as authentic values. AB is set 0 to indicate that the reply is inauthentic. Due to the layered encryption, only P_0 can read AB field. Therefore, P_{k-1}'s oblivious reply will look same for other peers. For the rest of the paper, we will use "reply" and "oblivious reply" terms interchangeably. The protocol runs as follows:

1) P_{k-1} sends $MTID_j|TS'|O_{k-2}^{k-1}$ to its predecessor, P_{k-2}.
2) P_{k-2} decrypts the top layer of O_{k-2}^{k-1}, which becomes O_{k-3}^{k-1}. Then, P_{k-2} prepares O_{k-3}^{k-2} and sends $MTID_j|TS'|(O_{k-3}^{k-1} \cup O_{k-3}^{k-2})$ to P_{k-3}. The operation $\cup$ denotes the concatenation in random order. Since O_{k-3}^{k-1} and O_{k-3}^{k-2} are encrypted and contain the same number of bits, P_{k-3} can not distinguish these replies after the randomization of their order.
3) P_{k-3} decrypts the top layers of O_{k-3}^{k-1} and O_{k-3}^{k-2}. It creates O_{k-4}^{k-3} and sends $MTID_j|TS'|(O_{k-4}^{k-1} \cup O_{k-4}^{k-2} \cup O_{k-4}^{k-3})$ to P_{k-4}.
4) This operation is repeated by all target peers until P_0 receives $MTID_j|TS'|(O_0^{k-1} \cup \ldots \cup O_0^2 \cup O_0^1)$. After decrypting the last layers, it checks AB fields and determines the authentic reply. P_0 sends this reply to the previous peer on P_r's query path. All peers on the path repeat the same operation until P_r receives the reply. If there are multiple replies with $AB = 1$, all of them are sent to P_r since only P_r can determine the authentic one. If $P_0 = P_j$, it ignores all replies and generates its own reply and sends it to P_r.
5) P_r decrypts the reply using K_{rj}. If the reply is containing the correct RID_i and TS' values, it is authentic. A malicious peer can not forge an authentic reply since it can not obtain K_{rj}.

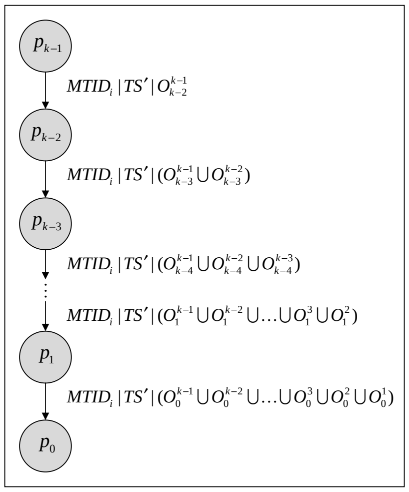

p_{k-1}
$MTID_i\,|\,TS'\,|\,O_{k-2}^{k-1}$
p_{k-2}
$MTID_i\,|\,TS'\,|\,(O_{k-3}^{k-1} \cup O_{k-3}^{k-2})$
p_{k-3}
$MTID_i\,|\,TS'\,|\,(O_{k-4}^{k-1} \cup O_{k-4}^{k-2} \cup O_{k-4}^{k-3})$
$MTID_i\,|\,TS'\,|\,(O_{1}^{k-1} \cup O_{1}^{k-2} \cup \ldots \cup O_{1}^{3} \cup O_{1}^{2})$
p_1
$MTID_i\,|\,TS'\,|\,(O_{0}^{k-1} \cup O_{0}^{k-2} \cup \ldots \cup O_{0}^{3} \cup O_{0}^{2} \cup O_{0}^{1})$
p_0

Figure 3. Message communication among target peers in the oblivious reply protocol

Figure 3 shows the flow of oblivious replies among target peers. If P_r is a global passive adversary, it can observe all the communication among target peers but it can not identify the sender of any reply. Identical reply sizes, semantic security assumption, layered encryption of replies, and randomization of their order on each target peer do not allow P_r to trace the replies. The oblivious reply protocol provides k-anonymity protection for trust holders as long as adversaries perform passive attacks. Due to space limitations, we can not give the proofs of our claim in this paper. Interested readers may refer to [23]. For a better understanding of our encryption scheme, similar ideas in [9], [14], [10] can be referred.

The oblivious reply protocol can not protect anonymity if adversaries perform active attacks, e.g. forging replies, dropping selected replies, skipping a target peer. If a target peer can be forced to stay complaint with the rules of oblivious reply protocol, these attacks can be prevented. Goldreich [21] shows that semi-honest behavior can be forced by compiling each instruction (message).

V. Discussion

Performance Considerations. We consider the message complexity to evaluate the performance of oblivious reply protocol. A reply is forwarded up to $O(k)$ times. For k replies, $O(k^2)$ network packets are forwarded in phase 2. More than one reply can be sent in the same network packet for efficiency. Assuming η is the number of replies in a network packet, phase 2 can be performed with up to $O(k^2/\eta)$ network packets. Note that the size of a reply decreases and η increases as replies are getting closer to P_0.

Sending trust holder certificates. In Section III-B, the bootstrap peer sends a separate certificate to each trust holder in step 6. If a global passive adversary observes the bootstrap peer during this step, it can learn P_j's identity. Therefore, the bootstrap peer sends a special message containing P_j's certificate, $MTID_j|TS'|TU_j(R_{BS}(H[ID_i]|TU_j|TS))|"Cert"$ to the trust network like a normal trust query. The last field in the message indicates that this message is a certificate, not a trust query. All target peers forward the message till the last peer in the search range receives it. Due to the encryption with TU_j, only P_j can read the content of the message. No peer can understand who is the receiver of the certificate.

VI. Conclusion

In a peer-to-peer system, defending anonymity against only local attacks results in a weak anonymity protection. An adversary with global passive attack capabilities or with some collaborators may learn about the anonymous peer by launching collaborative attacks. The oblivious reply protocol provides k-anonymity protection for a trust holder against global passive adversaries. The protocol requires $O(k/\eta)$ message exchanges where k is the group size and η is the number of reply messages that can fit into a network packet.

The oblivious reply protocol can be adapted to other DHT structures or applications that need responder anonymity. Moreover, our ideas can be used to support requester anonymity. A group of peers may generate an anonymous request so the identity of the requester is protected.

References

[1] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the 10th International Conference on Information and knowledge management (CIKM)*, 2001.

[2] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The (eigentrust) algorithm for reputation management in P2P networks," in *Proceedings of the 12th World Wide Web Conference (WWW)*, 2003.

[3] S. Hazel and B. Wiley, "Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems," in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[4] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[5] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, 2002.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*, ser. LNCS, vol. 2009, 2001.

[7] R. Dingledine, M. Freedman, and D. Molnar, "The Free Haven project: Distributed anonymous storage service," in *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*, ser. LNCS, vol. 2009, 2001.

[8] A. Singh and L. Liu, "Trustme: Anonymous management of trust relationships in decentralized P2P system," in *Proceedings of the 3rd IEEE Conference on Peer-to-Peer Computing (P2P)*, 2003.

[9] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, 1981.

[10] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1997.

[11] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM*, 2001.

[12] Gnutella. http://en.wikipedia.org/wiki/Gnutella.

[13] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[14] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[15] A. Beimel and S. Dolev, "Buses for anonymous message delivery," *Journal of Cryptology*, vol. 16, no. 1, pp. 25–39, 2003.

[16] J. Ren, T. Li, and Y. Li, "Anonymous communications in overlay networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2008.

[17] J. Han and Y. Liu, "Mutual anonymity for mobile p2p systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 8, pp. 1009–1019, 2008.

[18] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo, "Muon: Epidemic based mutual anonymity in unstructured p2p networks," *Computer Networks*, vol. 52, no. 5, pp. 915 – 934, 2008.

[19] M. Rennhard and B. Plattner, "Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, 2002.

[20] N. Borisov and J. Waddle, "Anonymity in structured peer-to-peer networks," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-05-1390, 2005.

[21] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2001, vol. 1.

[22] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the 14th annual ACM symposium on Theory of Computing*, 1982.

[23] A. B. Can, "Trust and anonymity in peer-to-peer systems," Ph.D. dissertation, Department of Computer Science, Purdue University, 2007.

Fast Packet Recovery for PULL-Based P2P Live Streaming Systems

Houssein Wehbe, and Gérard Babonneau
Orange Labs
4, rue du Clos Courtel
Cesson Sévigné, France, 35512
Email: {houssein.wehbe, gerard.babonneau}@orange-ftgroup.com

Bernard Cousin
IRISA / Université de Rennes 1
Campus de Beaulieu
Rennes, France, 35042
Email: Bernard.Cousin@irisa.fr

Abstract— **Nowadays, Peer-to-Peer (P2P) networks play an essential role in large scale live video transmission. Though many algorithms have been proposed to deal with packet loss in P2P networks, there is still a lack of mechanisms dealing with the delay and loss constraints of live video streaming. In this paper, we propose a new loss recovery mechanism allowing the quality optimization of live video transmitted on P2P networks. Its principal feature consists in request retransmission of lost packets from a peer different of the original packet sender. This mechanism increases the probability of choosing the best available peer to make the retransmission and hence, improves the received video quality before its display time. We show by simulations that the proposed solution is efficient in comparison with the current retransmission mechanisms. This solution is independent from the sender peer selection used algorithm.**

Keywords: Peer-to-peer network; video streaming; packet loss; packet recovery; efficient; retransmission mechanism.

I. INTRODUCTION

In the last few years, multimedia data transmission over IP networks has spread enormously. High quality video transmission is becoming more and more important. However, video transmission generates constraints on the network in terms of bandwidth, latency, error rate and jitter.

The emergence of peer-to-peer (P2P) systems in the live video transmission context, also called P2P live streaming systems, enables a large performance improvement when it is compared to a centralized system, mainly in terms of scalability. The P2P principle is based on the equivalence between the roles of all the system entities called "peers". In most of these systems, the video is split into chunks, i.e., fragments, which could be either pushed by the issuer peers or pulled by the receiver ones. In PULL-based P2P systems, the video applicant initiates itself the video distribution from its owners by deciding the chunk and the peer to use. In PUSH-based P2P approach, the video owners manage the system. They decide the chunk to be sent and the destination peer. In these systems, video clients have a more passive role. In both approaches, the peers build a P2P overlay used for chunks transmission. This overlay is a P2P network built on the basic of another network, Internet network for instance. The P2P overlay construction for each peer is achieved mainly by the selection algorithms of its peer neighbors. The peers in the overlay are connected via logical links, each of which composed of a path in the underlying network. In general, peers are the end hosts in the underlying network.

To get a good video quality, a client must receive its chunks before their display times. Generally, a chunk is transmitted over the internet in several IP packets. Without forward-error-correction (FEC) technique, the loss of one single packet makes this chunk unusable. The chunk receiver cannot use it nor send it to other peers. This may degrade the video quality in a large number of peers. To guarantee the quality of service, a packet recovery technique must be applied. This technique must ensure fast packet recovery. In other words, the packets should be received before their chunk display time.

In literature, P2P live streaming systems focus on the algorithm design of overlay construction [1] or chunk exchange policy between peers [2], i.e., the choice of chunks to send (respectively receive) and its destination receiver (resp. sender) in PUSH-based (resp. PULL-based) P2P system. However, they do not propose a specific recovery technique of lost packets. Moreover, usual recovery techniques do not take into account the video temporal constraints since they have been proposed initially for file transmission. In practice, these techniques propose to retransmit the lost packets from their original sender. However, if this peer is not available during the retransmission request, the probability for not receiving retransmitted packets in time will remains very high. Which can affect the video quality.

In this paper, we propose to request retransmission of lost packets from a randomly selected peer different from the original sender. Indeed, we assume that the original sender in not always the most appropriate to achieve the retransmission. The mechanism proposed in this work aims to choose an available sender for retransmission. This choice increases the probability for receiving the lost packets before their display time and enhances then the received video quality. Our retransmission mechanism is mainly applied in PULL-based P2P live streaming systems [3]. Indeed, these systems, it is the client that manages the video streaming and then the video quality improvement. It has information on the chunks it requires and on the system peers. In loss case, it may then easily apply our retransmission mechanism. The advantage of this mechanism is that it does not need adding new signaling messages, or the modification of the overlay construction used algorithms. The performance carried on the proposed mechanism shows its efficiency in comparison with current retransmission mechanisms.

This paper is organized as follows. Section 2 presents the state of the art of loss recovery mechanisms used in current P2P streaming systems. Section 3 presents our proposed solution. Section 4 introduces its performance evaluations. Finally, Section 5 concludes our works.

II. State of the art

The loss recovery schemes used in P2P streaming systems are based on packet retransmission or FEC techniques.

Packet retransmission is a very simple and well known method in the state of the art. In transport control protocol (TCP) the video receiver must send to the sender an acknowledgement for each received packet. The sender uses the lack of acknowledgement (or several acknowledgements with the same acknowledgement number) to detect lost packets and retransmit them. But given that TCP considers the packets loss as network congestion signs, it reduces its transmission rate systematically, at loss detection. This may additionally degrade the video quality. Moreover, in asymmetric networks (like P2P networks using, for instance, ADSL links as access network) when the receiver upload channel is blocked, the acknowledgements can be lost or arrive late to the sender. In this case as well, TCP sender needlessly reduces its transmission rate. For these reasons, TCP is still not suitable for real-time video transmission.

User Datagram Protocol (UDP) does not have these drawbacks because it maintains its transmission rate, but it does not have a lost packet recovery mechanism. To apply retransmission with UDP protocol, it is necessary to define a technique dealing with packet loss detection at receiver. That allows it to request their retransmission. We can, for instance, use the Real-Time Transport Protocol (RTP) [4]. This protocol allows the sender to number the packets before sending them. The receiver can thus send a retransmission request when a packet loss is detected. However, with this simple retransmission mechanism, video receivers do not have guarantee on packet recovery time and thus on the video quality.

To solve the retransmission issue in UDP, the FEC technique is proposed. In this technique, a redundancy data is added to the transmitted stream in such a way the lost packets can be recovered by the receiver without retransmission. The live video case, video source must know the rate of loss and its pattern for adding enough redundancy data to protect the stream. However, in P2P networks, where the links are heterogeneous and nodes behavior is unpredictable, video source cannot know the loss rate throughout the whole paths. Thus, the use of FEC in these networks does not allow correcting all loss types. Besides, generally the redundancy data quantity introduced into the stream is very high, because it is permanently configured to repair the worst loss case. This may increase network congestions and therefore contribute to the packets loss.

In reality, most of the current P2P streaming systems use TCP protocol for data transport [1][6][7]. Systems using UDP protocol (for instance [8]) do not propose a specific loss recovery mechanism. An enhancement for video quality transmitted over P2P network was proposed in [9]. It consists in protecting, via FEC and/or retransmission, the most important video packets. The authors of [9] suppose that these packets may contain an image of type I and P of a group of pictures (GOP) of a MPEG video stream. Indeed, images of these two types are more important than the images of the third type (B image) because the decoding of the third type is dependent on both first ones. Also P images depend on I images. Results presented in [9] show that this protection technique improves the video quality. However, performance analyses have been carried out on a P2P system where the receiver can receive the video packets from only one peer. However, these results are only valid in this particular case. Other works have been concentrated on retransmission of pre-recorded video. They do not treat the retransmission of live video where the video is more sensitive to transmission delay [5]. The authors of [11] propose a model-based packet scheduler for P2P streaming systems with retransmission. Their proposal consists in requesting retransmission of lost packets from their original sender. This sender is chosen initially by a specific technique. Its principle is that the video receiver watches the channel state of all the peers which it knows, and then it chooses for each video packet the peer minimizing the transmission delay. That may, according to the authors, accelerate recovery in the loss case and then improve the video quality. However, this technique is not optimal and it can not be applied in the live video case. Indeed, it requires selecting a sender for each IP packet, which may in P2P live streaming systems, according to [10], generate an important loss rate and a waste of network resources. Generally, it is preferable to apply the sender selecting technique for each chunk composes of several packets. This proposal will be detail in the next sections using live streaming video.

III. The proposed retransmission mechanism

In PULL-based P2P live streaming systems, video receiver decides which chunk should be requested and its sender peer. It exchanges, periodically with its neighbor signaling messages to know their available chunks, and then it chooses the chunk to request. The presence of several peers having the same video chunk raises an issue about the selection of the best peer to ensure a good quality of service for the chunk transmission. Several metrics exist which are used to select this peer, such as the available bandwidth between sender and receiver, available bandwidth of the sender node, the transmission delay between sender and receiver, i.e., one way delay, the round-trip time (RTT), the number of hops, etc. Signaling messages exchanged between peers or monitoring mechanisms allow estimation of these metrics.

Generally, a chunk is bigger than a IP packet [10]. A chunk will be sent in several IP packets. In network disturbance cases, one or some consecutive IP packets are lost. Video receiver considers a chunk unusable if at least one packet is lost. It can not display this chunk neither sends it to other peers. To improve video quality received by all the system peers, we carry out a lost packet retransmission. In fact, the retransmission of some packets allows decreasing network congestion by comparing it with the chunk complete retransmission or redundancy FEC that must be sent permanently even if there is no loss in the network.

In current P2P streaming systems, the receiver requests the retransmission of lost packets from their original sender. This is the case of TCP and most of retransmission techniques proposed with UDP [9][11]. This is the peer that sends their chunks. The receiver chooses the best peer available in the network according to the selection algorithm used. This peer is thus the best which may give the chunk at time. However, after loss detection, it is not necessarily the best sender of retransmission packets. Indeed, firstly, the processing and sending capacity of this sender is, on one hand, reduced by sending others packets belonging to the chunk affected by loss. On the other hand, the capacity can be reduced by new chunk requests received by this peer since it was selected. Secondly, processing and sending capacities of other peers may have significantly increased. Thirdly, loss rate and delay characteristics can vary according to the data volume transmitted. For example, small volumes (only one packet), often undergo shorter delays than large volume. For these reasons we propose to request retransmission of lost packets from the most available peer. It will not necessarily be the original sender. In this way, the probability of receiving the retransmitted packets before their chunk display time is increasing. This is very important in live video streaming because the receiver watches live video and hence is very sensitive to the display time. Figure 1 and Figure 2 show such scenario in which the mechanism proposes in figure 2 is more efficient than current mechanism. In the scheme represented in these figures, the peer P2 needs the chunk C1 existing in peers P1 and P3. Without loss of generality, let us assume that Round Trip Time (RTT) between P2 and P1 is less than RTT of (P2-P3), that the upload bandwidth of P3 is greater than that of P1 and the chunk C1 is composed of packets (p1, p2, p3, p4, p5). Though, P1 is closer to P2, its lower upload bandwidth does not guarantee the complete reception of C1 before its display time, noted TV in Figure 1 and 2. Thus, P2 sends its request to P3. Now, let us assume that the packet p3 is lost during its transmission between P3 and P2. If P2 requests retransmission of p3 from P3, C1 will be received completely at instant T1. However, if T1 is greater than TV display time, C1 will be considered unusable. However, if P2 sends its retransmission request to P1 (Figure 2), the probability of receiving the retransmitted packet at instant T0 smaller than TV, is high. Indeed, issuing a single packet does not request a large bandwidth. Thus, our retransmission mechanism can allow the complete reception of the chunk before its display time, which improve the quality of the video display.

In practice, several metrics may be used to select the best retransmission sender RTT, peers bandwidth, etc. The estimation of these metrics requires the exchange of signaling messages periodically between peers. These messages may increase network congestion, and then may contribute to the packet loss. To limit the control messages overhead, we propose to select the retransmission sender randomly among peers having the chunk affected by loss. This retransmission

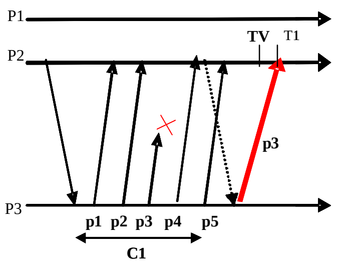


Figure 1: Video chunk transmission on P2P networks. Retransmission of lost packets is requested to the original sender.

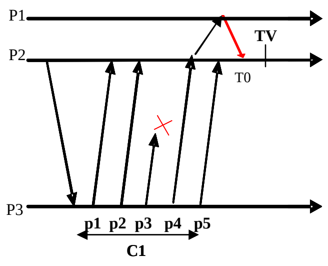


Figure 2: Retransmission of lost packets is requested to peer different from the original sender.

mechanism presents advantages in terms of flexibility and robustness. It doesn't imply any constraint on the data coding neither on network architectures. It does not need new control messages. The only condition is that the receiver has a list of peers having the chunk affected by the loss. This is always true in PULL-based P2P live streaming systems, because the receiver has this list before requesting initially the chunk. In consequence, this mechanism works properly with any PULL-based P2P system. To verify our mechanism effectiveness, we carried out a set of simulations whose results are presented in the next section.

IV. PERFORMANCE EVALUATION

Our mechanism is considered efficient if it can improve video quality in comparison with mechanisms where the receiver sends its request retransmission of the lost packets to their original sender, called hereafter "classic mechanism". A fairly high error rate will be applied. It puts the mechanisms under conditions that do not allow them to correct all the losses. Indeed, if we test the two mechanisms in situations where a correction rate of 100% is assured, it would not be possible to distinguish them.

To show our mechanism effectiveness, we carried out, using OPNET Modeler, a simulator modeling PULL-based P2P live streaming service. We explain in this following the chosen transmission algorithms.

To implement classic retransmission mechanism, we have made the following choices: (1) using peers with highest upload bandwidth, (2) request the chunks from the closest peer in term of RTT. The first choice makes, in loss case, the original sender of the lost packets more available. It has a high bandwidth, and can therefore answer to many requests (chunk transmissions or packet retransmissions) without having congestion problems. In the second choice the use of RTT aims to reduce as much as possible the packet transmission time. Applying these two choices, the probability of receiving the retransmitted packets before their chunk display time, will be increased. This improves the video quality.

Then, we need to define a chunk location technique. Its aim is to give to each peer in the system, a set of other peers and the chunks they have. In classic P2P system, this set is used to select a sender for a given chunk. This set is also used by our mechanism to randomly select a retransmission sender. But, our mechanism does not impose any constraint on the technique used to build this set of peers and their chunks. Moreover in any P2P system it always exists. Any chunk locating technique may be used. We propose to use the following chunk location technique. When a new peer connects to the system, it will receive a list of peers watching the video. These peers are called "Neighbor" and the number of neighbor is identified by the parameter "Neighbor_number". Periodically, a peer exchanges with its neighbor peer signaling messages to get their available chunks. The exchange period is noted "Exchange_period". This periodic exchange exists in many P2P systems such as [6][7]. The "Neighbor_number" must be limited to reduce signaling message overhead and thus the network congestion. In our simulator, we assign random neighbors to a new peer among all the peers watching the video. They will not thus necessarily be next to the first peer in term of geographic distance or RTT. This choice ensures a load balance in the network.

"Neighbor_number" and "Exchange_period" are two important simulation parameters. They may affect the video quality even if there is no loss in the network. If "Neighbor_number" is small, the receiver will not have much choice to select the best sender for a given chunk. The probability of choosing an unavailable peer will thus be high, with the risk to hinder the chunk reception and affect the video quality. Similarly, if "Neighbor_number" is large, the number of signaling messages exchanged between peers every "Exchange_period" may generate traffic which may increase network congestion and hence packets loss. Optimal values of these two parameters are thus needed to ensure the video quality. We carried out a series of simulations to find these values and verify the validity of our algorithms before testing our retransmission mechanism.

Our simulator consists of 500 peers and a central server generating a live video. Without lack of generality, the simulations were performed with a single video since the videos are independent. The same assumptions are considered in literatures [9][12]. We used a video of 300 kbit/s, as in [12]. Peers are homogeneous and have no constraint on their download bandwidth (it is often the case in P2P system, such as [13]). To respect the choices we discussed above, we attribute to peers a large upload bandwidth. We chose a value of 2 Mbit/s. It is very large compared to the video rate. It allows each peer to serve many requests simultaneously and ensure proper dissemination of content among peers. This value is possible on FTTH network, but also on xDSL network.

A. Performance evaluation of the model without packet loss

Performance evaluation without loss of packet allows us to find the good values for "Neighbor_number" and "Exchange_period". These values will be used later to test our retransmission mechanism with packet loss.

The most important metrics to measure are:

- Chunk loss rate: A chunk is considered lost if the receiver does not receive its all packets before its display time. To measure this metric, during the simulation, each peer computes the number of video chunks to be received and the number of chunks considered as lost. Using the values computed by all the system peers, we compute at the end of the simulation, the lost chunks percentage for all video chunks during the simulation. This percentage allows us to know the system loss rate. If it is at 0%, it means that all peers have received a perfect video.
- Control traffic percentage: This is the percentage of bytes of the signaling messages in relation to the total number of data bytes sent by the system peers (signaling + data). This metric allows us to see if the control traffic wastes the peer upload bandwidth.

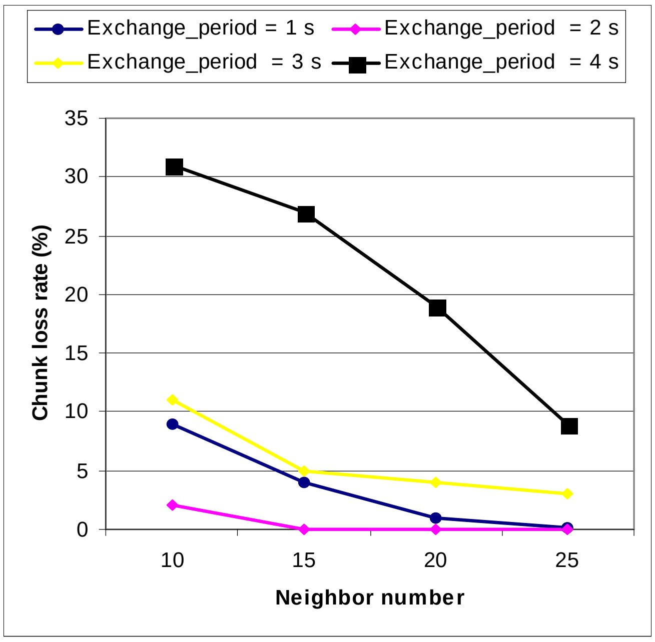


Figure 3: Chunk loss rate evaluation relative to neighbor number and signaling messages exchange period.

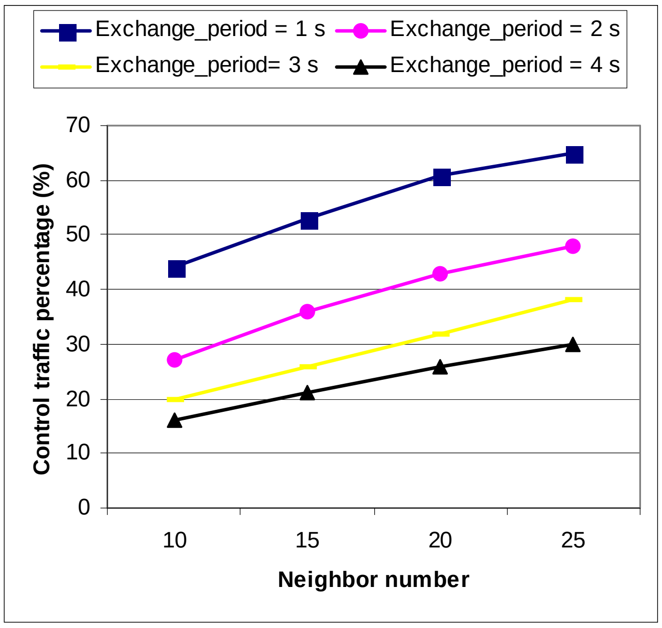


Figure 4: Control traffic percentage in all the system relative to neighbor number and signaling message exchange period.

Figure 3 shows the chunk loss rate relative to "Exchange_period" and "Neighbor_number". Since that there is no loss on the network, the reason of chunks loss is due their late reception. Figure 3 shows that the chunk loss rate decreases with the number of Neighbor increasing. Indeed, if the latter increases, the receiver has more choices to select a best sender for a given chunk. This increases the probability of finding an available sender, and reduces the probability of receiving the chunk out of delay.

However, Figure 3 shows that chunk loss rate also varies according to the "Exchange_period". When the period is larger or equal to 2 s, we remark that chunk loss rate increases with the increase of exchange period. The reason is that for a long exchange period, peers must wait some time before locating the new chunks in the system. This can delay the chunk requests and consequently the chunk reception time. The probability of receiving chunks with a delay will be large. This explains the existence of chunk loss with a large exchange period. In the case where exchange period is equal to 1 s, we can remark that the loss rate is not always 0%. For instance, this is different from the case where period is 2 s. With this short period, chunks requests will not be delayed. The reason of the loss in this case is, therefore, the non-availability of sender peers. Figure 4 gives us a verification of this fact. It assesses the control traffic percentage relative to "Neighbor_number" and "Exchange_period". Remember that each peer must exchange signaling messages with all its Neighbor at each "Exchange_period". Thus the control traffic quantity increases, thus, with the increase of "Neighbor_number". It also increases if the exchange period is decreased. Figure 4 shows this percentage variation. If the exchange period is 1 s, control traffic percentage is high; it is grater then 45%. In this case, peers send so much traffic control, which makes them less available to send the chunks. These chunks may be received out of delay. This explains the presence of loss when the period is 1 s.

It may be noted that "Neighbor_number" best value is 15 peers, when the "Exchange_period" is 2 s. These two values allow us to get the best video quality since the chunks loss rate is 0%. Thus minimizing the control traffic quantity exchanged between peers. By conducting simulations with these values, we can ensure that there is no chunk loss due to the algorithms selected such as the chunk location algorithm.

B.Performance evaluation with packet loss

To show the effectiveness of our mechanism, it is compared with classic retransmission mechanism.

We have shown in the previous section that our simulation model can guarantee a perfect video quality, if there is no loss on the network. To compare the two retransmission mechanisms, we have introduced on the links a uniform loss which rate is equal to 10% of transmitted packets. This is a very high rate compared to real network ones, but we have selected this value to show our mechanism effectiveness. Without lack of generality, we assume that signaling messages are not affected by loss, which could correspond to transport them by TCP.

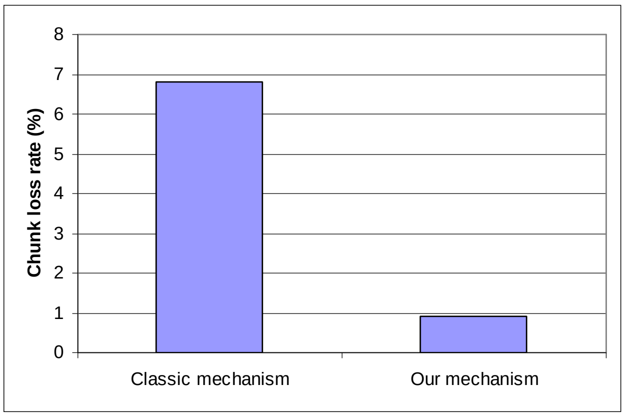


Figure 5: Comparison of the retransmission mechanism efficiency. Chunk sender is chosen relative to RTT.

Using parameter values deduced from previous paragraph, we carried out two simulations. In the first, we applied classic retransmission mechanism and in the second we applied our mechanism. A retransmission mechanism is considered efficient, if it ensures retransmitted packet reception before their chunk display time. In other words, the mechanism is considered efficient if it minimizes chunk loss rate. Remember that a chunk is considered lost if one of its packets is affected by a loss or if one of its packets arrives out of delay in relation to the display time.

Using the RTT selection parameter, we first measured the chunks loss rate with the two retransmission mechanisms. The results are presented in Figure 5. This figure shows that the chunk loss rate is 6.8% using the classic retransmission mechanism where our proposed mechanism has minimized this rate to 0.9%. These results show the effectiveness of our mechanism since almost 99% of chunks are arrived on time.

According to these results, we can notice that the chunk original sender is not necessarily the most adapted peer to make the packet retransmission in loss case. Retransmission from the original sender has not avoided the late arrival of chunks. Thus this retransmission mechanism does not guarantee the video quality. The results show also that the retransmission from a peer selected randomly among the neighbors, may improve the video quality since the chunks loss rate is very low. The proposed retransmission mechanism increases the probability of selecting an available sender peer to make retransmission. This increases the probability of receiving on time the retransmitted packets and hence improves the video quality.

In Figure 5, the transmission algorithms and simulation parameters were chosen to model the case where classic retransmission can work as well as possible. Thus we can assume that our mechanism will also be effective in any other case. To verify this assumption, we carried out simulations comparing the two retransmission mechanisms in another case. We kept the same simulation parameters and we changed the selection algorithm of sender peer. Indeed, if this algorithm ensures the choice of the best peer for a given chunk, then the retransmission from this peer can increase the probability of receiving the retransmitted packets before their chunk display time. Our mechanism shows its efficiency independently of the

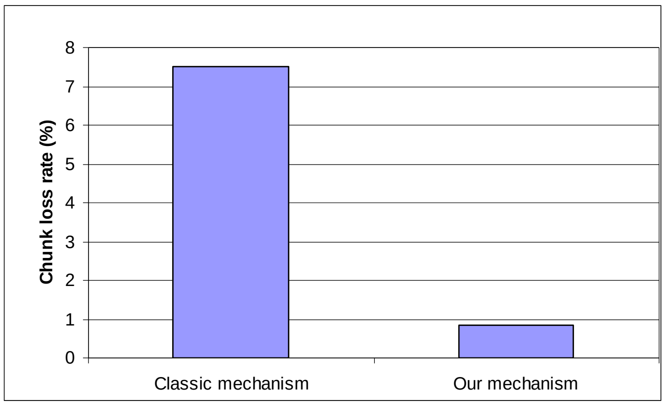


Figure 6: Comparison of the retransmission mechanism efficiency. Chunk sender is chosen randomly among the neighbors.

used algorithms. In the previous simulations, we have used an algorithm selecting the sender for a given chunk according to RTT. In the simulations presented in Figure 6, we used an algorithm based on randomly selection of this peer among peers having the chunk. We can remark that applying our retransmission mechanism, the chunk loss rate is always reduced in comparison with the classic retransmission. Thus, we assume that there is no impact of the sender peer selection algorithm on our retransmission mechanism effectiveness.

V. CONCLUSION

Today, video distribution towards a large number of receivers is a fundamental need. Appearance of P2P systems has allowed this need to be answered. Current P2P live streaming systems have low video quality. The main reason is the packet loss. This loss is due to the heterogeneous and dynamic characteristics of peers involved in the system, as well as the lack of performance guarantees in IP network.

Current P2P live streaming systems do not offer a specific mechanism to solve packet loss problem. Usual mechanisms do not take into account the packet recovery time since they are proposed initially for file transmission. In this paper, we have proposed a packet retransmission mechanism for PULL-based P2P live streaming systems. It consists in requesting lost packets retransmission from a peer randomly selected among the peers having the chunk, in general, a different peer of the original sender. We have shown that this increases the probability of receiving retransmitted packets before their chunk display time. With our mechanism, we have shown that the chunk loss rate is reduced to 0.9% improving then the video quality. The advantage of this retransmission mechanism is that it does not impose constraints nor on P2P architectures, neither on data coding. Moreover, this mechanism is independent from the sender peer selection used algorithm.

REFERENCES

[1] R. Lobb, C da Silva, A. Leonardi, E. Mellia, and M. Meo, "Adaptive overlay topology for mesh-based P2P-TV systems", 18th international Workshop on Network and Operating Systems Support For Digital Audio and Video, June 2009, pp. 31-36.

[2] P. Hoong and H. Matsuo, "Push-pull incentive-based P2P live media streaming system", Wseas Transactions on Communications, Feb. 2008, pp. 33-42.

[3] N. Magharei and R. Rejaie, "PRIME: peer-to-peer receiver-driven mesh-based streaming", IEEE/ACM Transactions on Networking, Aug. 2009, pp. 1052-1065.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 3550 - RTP: A Transport Protocol for Real-Time Applications", IETF standard, July 2003.

[5] F. Pianese., J. Keller, and E. Biersack, "PULSE, a flexible P2P live streaming system", 9th IEEE Global Internet Symposium, 2006, pp. 1-6.

[6] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays", Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2006-275, 2006.

[7] T. Do, K. Hua, and M. Tantaoui, "P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment", IEEE International Conference on Communications, 2004, pp. 1467-1472.

[8] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B.Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast", Eleventh ACM international Conference on Multimedia, 2003, pp. 45-54.

[9] B. Akabri.; H. Rabiee, and M. Ghanbari, "Packet Loss Recovery Schemes for Peer-to-Peer Video Streaming", Third International Conference on Networking and Services (ICNS), IEEE Computer Society, 2007, pp. 94.

[10] N. Hegde, F. Mathieu, and D. Perino, "Size Does Matter in Epidemic Live Streaming", Technical report 7032, INRIA, 2009.

[11] Y. Jung and Y. Choe, “Channel-adaptive packet scheduler for retransmission-based peer-to-peer stored-video streaming”, Tenth IEEE International Symposium on Multimedia, 2008, pp. 390-395.

[12] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?", IEEE Journal on Selected Areas in Communications, 2007, pp. 1678-1694.

[13] F. Picconi and L. Massoulie, "Is there a future for mesh-based live video streaming?", Eighth International Conference on Peer-to-Peer Computing, 2008, pp. 289-298.

[14] Z. Xinyan, L Jiangchuan, L. Bo, T. Shing, and Y. Peter, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", In IEEE Infocom, 2005, pp.13-17.

[15] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments", Nineteenth ACM symposium on Operating systems principles, 2003, pp. 298-313.

[16] C. Zhang, H. Jin, D. Deng, S. Yan, Q. Yuan, and Z. Yin, "Anysee: Multicast-based Peer-to-Peer Media Streaming Service System", Asia-Pacific Conference on Communications, 2005, pp. 274-278.

[17] H. Luo, D. Wu, S. Ci, A. Argyriou, and H. Wang, "Quality-Driven TCP Friendly Rate Control for Real-Time Video Streaming", Global Telecommunications Conference, 2008, pp. 1-5.

Using Groups to Reduce Communication Overhead in VANETs

C. Caballero-Gil, P. Caballero-Gil, J. Molina-Gil
Department of Statistics, Operations Research and Computing,
University of La Laguna,
Spain
Email: {ccabgil, pcaballe, jmmolina}@ull.es

***Abstract*—A Vehicular Ad hoc NETwork (VANET) is a type of mobile Peer-To-Peer wireless network that allows providing communication among nearby vehicles and between vehicles and nearby fixed roadside equipment. The lack of centralized infrastructure, high node mobility and increasing number of vehicles in VANETs result in several problems discussed in this paper, such as interrupting connections, difficult routing, security of communications and scalability. Groups are proposed as a solution to decrease the number and size of packets exchanged among vehicles because by using groups, VANETs can be split in small sub-VANETs that allow to avoid sending the same information through different paths. In this way, the proposal improves the efficiency and safety of communications through a hybrid model that combines symmetric and asymmetric cryptography. To reach this goal, nodes must know how to behave depending on their state, so this paper provides a full description of each group management process and of how to deal with the information within a group.**

***Keywords*-VANET; Groups; P2P; wireless networks;**

I. Introduction

A VANET is a spontaneous Peer-To-Peer (P2P) network formed by moving vehicles. As any other MANET (Mobile Ad-hoc NETwork), a VANET has no central infrastructure, which implies the need of self-management in a distributed environment where nodes have to adapt to unpredictable changes. Such autonomic networks present unique challenges such as high mobility, real-time constraints, scalability, gradual deployment and privacy.

Intelligent VANETs hybridly integrate multiple ad-hoc networking technologies such as WiFi IEEE 802.11 b/g, WiMAX IEEE 802.16, Bluetooth, etc. to achieve effective wireless communication. Such networks constitute a fundamental part of the Intelligent Transportation System (ITS). Many research projects on ITS are being financed by the European Commission because road safety is classified as a priority objective. Different possible situations exist where communications between vehicles would help to prevent accidents and to avoid traffic jams, which would save time and money, reduce contamination of the environment and consumption of fuel reserves.

Several general characteristics can be considered in wireless networks: authenticity, privacy, anonymity, cooperation, low delay, stability of communications, scalability, etc. [2], [8]. However, when dealing with VANETs, the protection of those properties is an even greater problem due to specific characteristics of these networks, such as very changing scenarios, from local roads with very few vehicles to cities or highways full of vehicles. In this work we propose the use of groups in VANETs, which will allow to optimize communication in dense traffic situations, and to define group secret keys for the use of symmetric cryptography to ensure information exchanges efficiently.

Section II gives a definition of group. In Section III, the different group stages included in our proposal are explained in full detail: Detection, Election, Creation, Membership and Life of a group. Section IV describes how communications are conducted within the group. Sections V and VI analyze simulation results. Finally, conclusions close the paper.

II. Groups

A group in a VANET is defined as a set of vehicles that are located in a close geographic area whose formation is determined by the mobility pattern of vehicles. The group needs a minimum of vehicles and is managed by a given node called ”leader of the group”. All vehicles forming part of a group have a direct wireless connection with the leader of such a group and share a secret key.

There are several bibliographic references that propose the use of groups or clusters, which are the same in VANETs. [4] presents a theoretical analysis of a directional stability-based clustering algorithm. [5] describes clusters where the leader is the node in the middle with the lowest identifier. [7] proposes clusters to maximize the advance of the relayed information and to avoid interferences, but there the head cluster must know the exact positions of nodes in the cluster. None of these works define in detail the processes that nodes have to complete for group management and do not show any implemented scheme to demonstrate the reliability of obtained data [1], which is the main objectives of this work.

Groups will be used only when the conditions of the routes require it. Examples are dense traffic, traffic jams or congested highways, where the density of vehicles in a geographic zone causes that the number of communications is huge. But groups are formed before the number of nodes begins to degrade the network. Without any mechanism to minimize the number of communications, a simple broadcast will be launched from every vehicle generating a lot of unnecessary redundancy. The number of packets generated depends on the number of nodes in the network and interconnection among them. Therefore, it will be generated n packets for each data communications where n is the number of vehicles with On Board Unit (OBU) in the network (in the scope of interest). This number of connections is not extremely large, and perhaps would not need to taking steps to reduce that number, but some studies like [3] showing that many vehicles duplicate data packets causing collisions in the information that is sent, which degrades communication quality.

On the other hand, where the number of vehicles is low and there is no saturation of communications, the groups are not used. With a group scheme would be generated 3 connections per group for every data. The first one goes from the vehicle which produces the information to the leader, then, the leader launches a multicast to all vehicles of the group. Finally, another connection between the leader and another vehicle (in the best position) continues multicasting the information. Therefore it will be generated $(n/numberofgroups)*3$ for each data packet. Vehicles will form groups according to dynamic cells where the leader is the vehicle with VANET technology that has initiated the group or that has the greatest number of neighbors when the previous leader falls below an established threshold for group formation. The definition of these groups will be based on the average speed of the route and the direction in which vehicles circulate, so that vehicles that circulate at a speed near that one will not change group during their journey on that route. The group leader will be the one in charge of managing the information and connections.

III. Group Stages

We distinguish among several stages in group management, corresponding to different situations of vehicles, depending on the route and on their status in each moment. The stages are: Detection, Election, Creation, Membership and Life of a group.

VANETs are wireless networks where there are a large number of highly volatile connections between vehicles. For this reason it is necessary to define in detail the way in which vehicles must act according to their situation.

The global network life scheme proposed in this paper is as follows. Initially all nodes start in the Group Detection stage. After this, they can enter the Creation or the Election stage, depending on the circumstances. After Group Creation, the node would be the group leader, while after Group Election, the node would proceed to Group Membership.

A. Group Detection

This is the first stage, where vehicles are in normal conditions without dense traffic. This stage is described in Algorithm 1, where $neighbor(i)$ denotes the i-th neighbor of the node that initiates the stage. From time to time the vehicle checks the number of neighbors and the number of leaders among them. If there is at least one neighbor who is leader of a group, the node proceeds to the Election stage, and otherwise to the Creation stage. This stage does not generate any traffic of control due to the fact that all the necessary information is contained in the beacons that nodes generate.

Algorithm 1 GroupDetection

```
01:function GroupDetection (...)
02:   numberOfNeighbors = 0;
03:   numberOfLeaders = 0;
04:   while (neighbor(i) exists) do
05:      if (isLeader( neighbor(i) )) then
06:         numberOfLeaders = 0;
07:      end
08:      numberOfNeighbors++;
09:      i++;
10:   end
11:   if (numberOfLeaders == 0) then
12:      GroupCreation();
13:   else
14:      GroupElection();
15:   end
16: end function
```

B. Group Election

This stage starts when the vehicle has found among its neighbors at least one node that is leader of some group. If there is only one neighbor who is a group leader, the choice is automatic. Otherwise, if there are several leaders, the vehicle has to choose one of them to join it. Algorithm 2 shows this stage, where $groupValue$ denotes a quantity used for the choice and $groupLeader(j)$ represents the j-th neighbor of the node that is leader of a group.

If there are several leaders among its neighbors, the vehicle chooses one according to the $groupValue$ that depends on the following values for each group j:

- Density $A(j)$ of vehicles.
- Average quality of signal $B(j)$ within the vehicles.
- Time $C(j)$ during which it has been connected to the leader.

Algorithm 2 GroupElection

```
01:function GroupElection (...)
02:   if (numberOfLeaders $\geq 1$)then
03:      j = 1;
04:      e = 0;
05:      groupValue[e] = 0;
06:      while (groupLeader[j] exists) do
07:         groupValue[j] = A(j)+B(j)+C(j);
08:         if (groupValue[j] $\geq groupValue[e]$)then
09:            groupValue[e] = groupValue[j];
```

```
10:         end
11:         j++;
12:       end
13:   else
14:       e = 1;
15:   endif
16:   sendRequest (groupLeader[e]);
17:   receiveGroupKey(groupLeader[e]);
18:   GroupMembership();
19:end function
```

Once the group has been chosen, the vehicle sends a login request encrypted with its public key to the group leader. After authenticating it, the leader sends the group secret key encrypted with such a public key and from then, the vehicle becomes part of the group.

C. Group Creation

In the Group Creation stage (Algorithm 3), the vehicle is not close to any leader of a group. It should check whether within their neighbors there are at least X nodes that do not belong to any group, plus a variable Y that indicates the number of vehicles that can either turn off, separate or not join the new group that is being created. If the number of neighbors without group is lower than the minimum threshold required for group creation, the vehicle waits a period $time1$ and starts again the Group Detection stage. Otherwise, if the number of neighbors is greater than the threshold $X + Y$, the vehicle begins a new Group Creation process. In order to do it, it multicasts a group creation request towards all neighbors with distance equal to 1. Nodes that receive this request respond accepting or rejecting the invitation. If the number of neighbors that accept the invitation is greater than the minimum threshold X, the new group leader sends to each node the secret key of the group encrypted with the public keys of each node. In this moment the new group is formed. Otherwise, the number Y of estimated vehicles is increased by adding the number of vehicles that did not accept the invitation.

Algorithm 3 GroupCreation

```
01:function GroupCreation (...)
02:   if (numberOfNeighbors $\geq X + Y$)then
03:       AcceptedNeighbors = 0;
04:       n = 1;
05:       l = 0;
06:       MulticastNeighbors (NeighborsList[]);
07:       for (n=1; n $\leq numberOfNeighbors; n++$)do
08:           ReceiveGroupElection(n);
09:           if (neighbor(n) accept) then
10:               acceptedNeighbors(l) = neighbor(n);
11:               l++;
12:               acceptedNeighbors++;
13:           end
14:       end
15:       if (acceptedNeighbors $\geq X$)then
16:           for (n=1; n$\leq acceptedNeighbors; n++$)do
17:               SendGroupKey(acceptedNeighbors(n),
18:               PuKacceptedNeighbors(n));
19:           end
20:           GroupLife();
21:       else
22:           Y = Y + X - acceptedNeighbors;
23:           Wait(time1);
24:           GroupDetection();
25:       end
26:   else
27:       Wait(time1);
28:       GroupDetection();
29:   end
30:end function
```

In conclusion, this stage requires: a multicast of invitation to join the new group, unicast responses from n users and a multicast to relay a message that enables the members to build the group secret key. This means a total of $2n + 1$ packets in case of positive group creation, and $n + 2$ if the process fails. The Group Creation starts when the appropriate number of neighbors reaches a certain threshold of traffic, but without to be dense traffic. Consequently, management packets generated at this stage not increase communications in dense traffic conditions.

D. Group Membership

Once the group is formed, the leader must periodically validate that the group continues being useful. Otherwise, it would be necessary to change the leader or to end the group.

Algorithm 4 shows the process where a node leaves the group which it belongs. When the node loses any contact with the leader of the group for certain time, the node stops to belong to its group and begins the Group Detection stage if node density exceeds the corresponding threshold.

Algorithm 4 GroupMembership

```
01:function GroupMembership (...)
02:   if (See( groupLeader )) then
03:       Wait(time3);
04:       GroupMembership();
05:   else
06:       Wait(time4);
07:       if (See( groupLeader )) then
08:           Wait(time3);
09:           GroupMembership();
10:       else
11:           finalGroupMembership();
12:           GroupDetection();
13:       end
14:   end
15: end function
```

E. Group Life

Algorithm 5 shows how the leader of a group periodically checks that the group is still useful. If group size falls below a certain threshold, the leader checks whether it has a number of neighbors greater or equal to D (dense traffic threshold) and waits for $time2$ instead of ending the group in order to avoid introducing group management traffic when the vehicle is in a dense traffic situation.

If the leader is not in a dense traffic situation, it begins a leader change or a group ending process. First, the leader asks about the neighborhood density in order to know if neighborhood density (number of neighbors of the same group or without any group near) is bigger than X. It also finds out which of its neighbors has the largest number of neighbors. After this, it sends a multicast signal of leader change to all its neighbors. The new leader will begin a Group Creation stage with those nodes without any group that are in its transmission range. In the absence of any neighbor exceeding the threshold, the leader sends the group ending signal through multicast to all its neighbors.

Algorithm 5 GroupLife

```
01:function GroupLife (...)
02:   for (n=1; n$\leq numberOfNeighbors; n++$)do
03:       if (Belongs( neighbor(n),group(a) )) then
04:          groupSize++;
05:     end
06:   end
07:   if (groupSize $\geq X$)then
08:      Wait(time2);
09:      GroupLife();
10:   else
11:      if (numberOfNeighbors $\leq D$)then
12:         newLeader=0;
13:         for(n=1;n$\leq numberOfNeighbors; n++$)do
14:           //groupSize+withoutGroup
14:            pot = potential(neighbor(n));
15:            if ((pot $\geq X) and (pot \geq groupSize)$)then
16:               groupSize=groupSize(n);
17:               newLeader=n;
18:            end
19:         end
20:         if (newLeader == 0) then
21:            Multicast (End-of-Group-Signal);
22:            GroupDetection();
23:         else
24:            Multicast (Leader-Change-Signal);
25:            //New leader init GroupCreation proccess
26:            GroupDetection();
27:         end
28:      end
29:   end
30:end function
```

IV. Message Management inside Groups

By using groups the number of communications can remarkably decrease without missing any useful information.

Algorithm 6 shows the steps that a vehicle beloging to a group must follow in order to process an input signal.

Algorithm 6 Message Management inside Groups

```
01:function MessageManagement (...)
02:   if(AmIfinalDestination(packet)) then
03:      TreatData(packet);
04:   else
05:      if (AmILeader()) then
06:         if (IsPublicInformation(packet)) then
07:            TreatData(packet);
08:            Multicast(packet, GroupKey);
09:         else
10:         relayer = estimatePosition(DestinationNode);
11:            Unicast (packet, relayer);
12:         end
13:      else
14:         if(IsForwardingSequence(packet)) then
15:         relayer = estimatePosition(DestinationNode);
16:            Unicast (packet, relayer);
17:         end
18:      else
19:         if(IsSentbyLeader(packet)) then
20:         relayer = estimatePosition(DestinationNode);
21:            Unicast (packet, relayer);
22:         end
23:      else
24:         Unicast (packet, GroupLeader);
25:      end
26:   end
27: end function
```

If the node is the final destination, it simply processes the information. Otherwise, it checks whether data were sent by the group leader. In particular, the leader can send two types of packets towards any member of the group that is not the final destination of the data:

- A connection of a vehicle to Internet services, or any other supplied service where it is necessary a relay of an information sequence,
- A packet of other type of information that must be forwarded towards other parts of the network.

With respect to this second type of packets there are two types of communications that must be differentiated:

- safety-related information
- commercial advertising

In both cases the vehicle belonging to the group that receives or produces the communication, sends it to the group leader who will forward it to all connected members of the group and towards the zones where the message has not been yet spread.

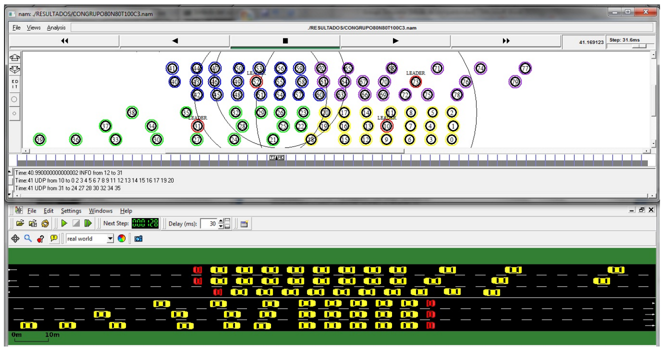
An Internet connection can be passed through to another group through intermediate nodes who forward the information. If one vehicle wants to connect to the Internet, it sends a request towards a vehicle outside its group, which will forward the request towards its leader. The leader will send the request towards other group or the RSU (Road Side Unit), which will answer by giving some details about the transmission such as the number of packets required for the connection. With this information, and knowing both the location and the speed of the vehicles in the group and of the vehicle that wants to connect, the leader calculates how long is the connection between the RSU, the intermediates vehicles and the destination vehicle. Then, it balances the load of connection so that the packets get to the destination as quickly as possible. Once the leader has informed the intermediate vehicles, they connect with the RSU and with the connected vehicles. After this, they relay the Internet connection.

For these types of communications, mechanisms for enforcement cooperation [6] are necessary because without them, intermediate vehicles would not have the necessary incentives to relay others connections, what would disable any type of service that incorporates an indirect connection with the RSU.

V. Simulation

Both the feasibility and effectiveness of our approach are shown through the figures where a simulation exemplifies its performance. In the first part of our demonstration (Figure 1), a NS-2 and SUMO display shows the VANET state in one moment when groups are operating.

The most relevant options selected for the demonstration have been: Total number of vehicles: 80, number of vehicles with OBUs: 80, number of lanes for each direction: 3 and 3, simulation time: 100 seconds, moment when retransmissions begins: 40 seconds, retransmission period: 10 seconds, distance relay nodes: 75 meters, traveled distance before the traffic jam happens: 800 meters.


nam: ./RESULTADOS/CONGRUPO80N80T100C3.nam
File Views Analysis
./RESULTADOS/CONGRUPO80N80T100C3.nam
41.169123
Step: 31.6ms
LEADER
LEADER
LEADER
LEADER
TIME
File Edit Settings Windows Help
Next Step:
Delay (ms): 30
real world
0m 10m

```
Time:40.990000000000002 INFO from 12 to 31
Time:41 UDP from 10 to 0 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 19 20
Time:41 UDP from 31 to 24 27 28 30 32 34 35
```

Figure 1. Simulation

The implemented simulations for groups consider four levels of development: vehicle mobility, node energy, group formation and P2P communications in the network.

- The vehicle mobility layer manages the node movement in the movement pattern, which defines roads, lines, different speed limits for each line, traffic jams, etc.
- The node energy layer is used to distinguish between vehicles with and without OBUs. Vehicles without OBUs are present in the road but do not contribute in the communications.
- The group formation layer defines which vehicles belong to each group, who is the leader of each group, who generates traffic information and who relays information to other groups.
- The P2P communications layer is responsible for the definition of which nodes are in the transmission range of the retransmitting node at any time.

Statistics extraction. Simulations give essential statistics such as number of generated, dropped or lost packets or bytes. These basic statistics data are useful to make efficient simulations for large scale scenarios.

Two implementation mechanisms. Simulations provide two mechanisms to implement VANETs: One with groups and the other without them. The implementation without groups does not involve the group formation layer while the implementation with groups allows comparing the behavior and data of both types of simulations.

VI. Analysis results

The implemented simulations with groups can be compared with results obtained from the simulation without the use of groups with the same topology (see Figure 2). This helps to illustrate the vehicular P2P network evaluation.

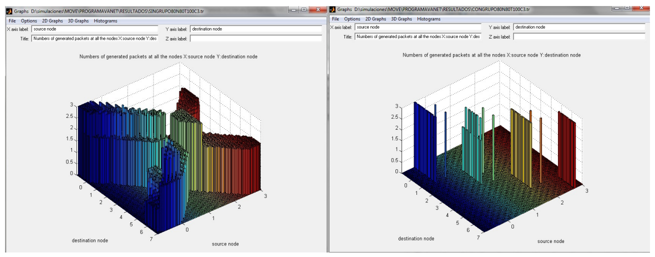


Figure 2. Generated packets without and with Groups

Among the obtained information from the simulations we have the number of packets and bytes generated, sent, broadcast, received, lost, etc. for each node. Also, other general information shown is the number of packets generated or lost in the whole network, the number of formed groups, which nodes are the leaders of the groups, which nodes generate packets and which nodes forward them, etc. In addition to all this information, another interesting aspect is that it provides a detailed simulation of what happens in each moment in the VANET thanks to the use of the NS-2 display. It also shows the traffic model through the SUMO tool while the information is represented using TraceGraph.

Table I
SIMULATION RESULTS

VARIABLES		USING GROUPS		WITHOUT GROUPS		
number of vehicles	vehicles with OBU	generated packets	loss packets	generated packets	loss packets	groups formed
60	10	278	107	167	12	1
31	15	598	402	277	63	2
40	20	825	443	351	0	2
31	31	2343	1804	638	139	2
40	40	2805	2014	932	135	2
50	50	5077	3981	1101	182	2
100	50	3539	2350	1327	68	2
60	60	5732	4415	1314	199	3
80	80	6675	4529	2120	215	4

Table I shown some result of simulations. We have chosen to use the following set of parameters to be varied in order to study the network behavior under different conditions: simulation time: 100 seconds, retransmission period: 15 seconds, distance relay nodes: 75 meters, 3 lines for each direction: 3 and 3, moment when retransmissions begins: 40 seconds, maximum number of hops: 1, routing protocol: DSDV, traveled distance before the traffic jam happens: 800 meters. The remaining variables are indicated in the table. Finally, the values of all parameters which are not explicitly mentioned are set equal to the different simulations.

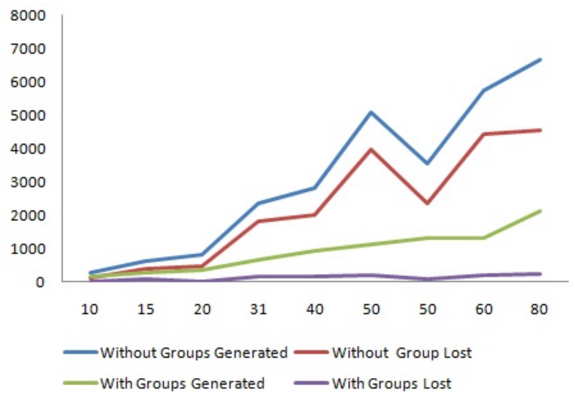


Figure 3. Generated packets

We can observe in Figure 3 the comparison between the average generated and lost packets: it is clear that, without the use of Groups in VANETs, the number of generated packets grow up much faster than with the use of Groups. But also the lost packets grow up much faster. The main reason is likely to be the heaviest traffic load that VANETs generates in traffic jams conditions: indeed, the original protocol makes a massive use of broadcast operations. The use of Groups will help to decrease the percentage of lost packets and to perform the VANETs operation.

VII. CONCLUSION

In this paper, the use of groups has been proposed as a solution to decrease the number of communications in VANETs under dense traffic conditions when the overhead of transmitted data causes a considerable drop in communication quality. In particular, a complete description of the proposed scheme for autonomic group management in VANETs is provided, which includes differentiation among possible vehicle states: from the initial state when it does not belong to any group, to the choice of an existent group to join it, the creation of a new group, and the end of a group. This paper also shows how to proceed with group communications.

A complete analysis has been done through simulations using the open source traffic simulator SUMO and network simulator NS-2. Such simulations allow the analysis of the operations at each stage, and a comparison between communication overhead when using groups and without using them in VANETs.

ACKNOWLEDGMENT

Research supported by the Ministerio de Ciencia e Innovación and the European FEDER Fund under Project TIN2008-02236/TSI, and by the Agencia Canaria de Investigación, Innovación y Sociedad de la Información under PI2007/005 Project.

REFERENCES

[1] P. Caballero-Gil, C. Caballero-Gil, J. Molina-Gil, and C. Hernández-Goya A simulation study of new security schemes in mobile Ad-hoc NETworks, Lecture notes in computer science, EUROCAST 2007:Vol:4739:73-81

[2] P. Caballero-Gil, C. Caballero-Gil, J. Molina-Gil, and C. Hernández-Goya, Flexible Authentication in Vehicular Ad hoc Networks, Proceedings of APCC IEEE Asia Pacific Conference on Communications. 2009.

[3] O. Dousse, F. Baccelli, and P. Thiran, Impact of Interferences on Connectivity in Ad Hoc Networks. INFOCOM 2003

[4] P. Fan, P. Sistla, and P. C. Nelson, Theoretical analysis of a directional stability-based clustering algorithm for vanets. Vehicular Ad Hoc Networks 2008:80-81

[5] Y. Gunter, B. Wiegel, and H. P. Gromann, Medium Access Concept for VANETs Based on Clustering. VTC Fall 2007:2189-2193

[6] J. Molina-Gil , P. Caballero-Gil, and C. Caballero-Gil, A Vision of Cooperation Tools for VANETs, Proceedings of the First International Workshop on Data Security and PrivAcy in wireless Networks, WoWMoM, June 2010.

[7] Z. Y. Rawashdeh and S. M. Mahmud, Media Access Technique for Cluster-Based Vehicular Ad Hoc Networks. VTC Fall 2008

[8] M. Raya, P. Papadimitratos, and J.P. Hubaux, Securing Vehicular Communications - Assumptions, Requirements and Principles, Proceedings of Fourth Workshop on Embedded Security in Cars (ESCAR), 2006.

Wireless P2P: Problem or Opportunity?

Rossana Motta and Joseph Pasquale
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, USA
Email: rmotta, pasquale@cs.ucsd.edu

***Abstract*—P2P is currently considered a problem by many wired and wireless providers, especially because of the large amount of traffic it generates. However, given new technology developments such as Wi-Fi Direct, we see an opportunity for P2P in mobile settings that, rather than treating mobiles as second-class citizens, seeks to take advantage of their capabilities over stationary devices. In this paper, we outline a new approach for fully decentralized mobile P2P that allows mobiles to run a variety of services and at the same time alleviates data overload in mobile networks. We consider mobiles integral and a richer class of nodes, and assume that they are capable of forming P2P networks without necessarily relying on the aid of stationary hosts or servers. We present a new architecture based on JXTA middleware, which is meant to not only optimize mobile resources, but also aims to take full advantage of the features that mobiles offer, such as mobility, ubiquity, location-awareness and sensors.**

***Keywords*-Wireless; Peer-To-Peer; Mobiles; Broadbands.**

I. Introduction

Mobile broadband networks currently serve not only users of mobile devices but also subscribers that, for reasons such as lack of coverage of wired networks, convenience of mobility, or price, choose mobile broadband as their primary connection. Thus, the scenarios where P2P is currently used are wide-ranging, with mobile devices connected either to a mobile network or to a wi-fi access point, and with stationary hosts connected either to a wired network or to a mobile network. While wired networks may also have some problems with traffic overloads due to P2P, these problems are especially exacerbated in mobile networks.

The introduction of the Wi-Fi Direct protocol [1] represents an opportunity to support P2P in a way that can effectively relieve the overload on some mobile networks. Both Internet Service Providers (ISPs) and consumers could benefit from this. Wi-Fi Direct will allow a mobile to connect directly with another mobile in its range that is also running the protocol, with no hubs or routers are involved. Data rates are expected to be over 250 Mbps with a coverage range of about 100 meters [1].

In this paper, we present some of the main issues that motivate the need for an alternative strategy for mobile P2P. We then present a new middleware framework based on JXTA to support mobile P2P systems that, amongst other capabilities, takes advantage of technologies such as Wi-Fi Direct.

II. P2P, Mobile Networks, and Emerging Technologies

A. P2P and file demand

As reported by Gigacom in 2009, the ISPs of wired networks have long considered P2P as a "voracious, bandwidth-eating monster" and have adopted aggressive traffic shaping policies and bandwidth caps in order to stop P2P traffic from overrunning their networks [2] [3]. Operators of cellular networks are paying closer attention to P2P traffic, which has become a problem in mobile networks too [2]. Many mobile providers have adopted traffic shaping policies to throttle P2P traffic at least during certain parts of the day (e.g., [16] [4]) or even blocking it completely [5].

According to a study conducted by Allot Communications and published in 2010 [6], P2P is the single largest factor leading to congestion. In mobile broadbands it accounts for 34% of bandwidth utilization in the 5% of users generating the largest amount of overall traffic. However, the study also reveals a rapid growth in HTTP traffic during 2009. Allot reports that in the second half of 2009 "HTTP downloads grew by 73%, and have become a feasible alternative for massive file sharing" [6]. In fact, a number of HTTP services are becoming extremely popular to share files, for example one-click hosters (e.g., RapidShare, MediaFire, MegaUpload, etc.) and other file hosting services, such as DropBox and LiveMesh. These services typically offer free or paid plans for users to upload and download files via HTTP. RapidShare reports to be currently hosting over 10 petabytes of data.

These facts suggest that the traffic problems in many current networks are not solely caused by P2P technologies, but rather by the rapidly growing demand for file sharing and other services. Should P2P technologies disappear, the current problems will likely remain, but shifted towards other existing or yet to emerge technologies. Additionally, an increasing number of paid services currently offer virtual private networks that can be used to tunnel and encrypt any type of traffic, including P2P, which is thus disguised and can circumvent ISP restricting policies.

B. Mobile Broadbands

Mobile broadbands, which allow wireless access to various Internet services through 3G, GPRS or other cellular networks, are becoming very popular worldwide. According to the statistics collected by Allot Communications, mobile broadband usage has increased of 72% in the second half of 2009 [6]. Mobile broadbands are being more and more widely used as a replacement for wired connections to exchange data. Verizon reports a wireless data revenue rise of 41% from 2007 to 2008 [7].

It is predicted that the number of mobile devices may pass the number of PCs and laptops by 2013 [31], which directly translates in a further increase in the number of users of mobile networks. Currently the 3G network counts about 1 billion subscribers, which are expected to reach 2.8 billion by 2014 [31]. A few providers have been trying to relieve their cellular networks by spreading wi-fi hotspots in some areas with particularly high traffic so that customers can use them in place of the 3G network for data transfers. This approach is clearly expensive and requires adding hardware and infrastructure.

However, 3G is mainly designed for voice traffic, which is much less bandwidth intensive than data traffic. Although the network has a data overlay capability, the latter is designed for bursts of data rather than continuous streams. For this reason, a continuous stream of data, which is often seen in P2P transfers, represents a problem for a mobile network. If many users are transferring large amounts of data, this will clog the network and leave other subscribers unable to access their own services [8].

Additionally, 3G networks have intrinsic limitations in the number of connections that a sector can accept, regardless of the amount of data transferred. Thus, whenever many nearby users try to access the network simultaneously, as often happens during social or emergency events, some users may be denied a connection because that sector of the network has already reached the maximum number of connections. Little can be done to prevent this problem, and even setting up the cellular network to reset connections more often provides limited results. It is estimated that 40% of the connections transfer less than 100 bytes, however many mobile applications automatically attempt to establish very frequent connections [31] and this greatly contributes to quickly clogging the mobile network.

One of the possible problems with restricting P2P usage in mobile networks is that the experience of their subscribers may significantly degrade, as users are unable to reach acceptable speeds in P2P communications. The possible dissatisfaction in their service may have a negative impact on mobile broadband ISPs, as customers may feel like the broadband subscription is not worth paying for. In fact, subscribers are often expecting their mobile networks to perform in much the same way as fixed networks and to be able to do the same things they would do on a normal wired network [2]. On the other hand, as other non-P2P services that generate high amounts of data gain popularity, restricting P2P becomes less and less effective to relieve overload problems.

C. Emerging Technologies

The upcoming release of the Wi-Fi Direct protocol [9] holds promise for promoting the feasibility and convenience of fully distributed wireless technologies. The Wi-Fi Direct protocol uses physical P2P communication in that it allows wi-fi devices in range to talk to each other without the need of intermediate wireless access points or routers. While few devices already support the protocol [10], the Wi-Fi alliance has announced that the wi-fi cards of many existing wireless devices can be made compatible with Wi-Fi Direct through upcoming software upgrades [1].

Since Wi-Fi Direct does not require any special hardware or infrastructure and can be installed on virtually any device with a wireless card, it is not unreasonable to expect that the concentration of devices running the protocol will become high in populated areas. This implies that in general each device could easily find a number of other devices in its range, which provides an opportunity to build chains of local connections. Additionally, even if two devices are not in each other's range, a connection could be established using multiple hops through other intermediate devices.

While P2P systems in theory can always be built on top of centralized network technologies, and the advent of 4G [1] will make data transmission over mobile networks more efficient, relying on fully distributed technologies can be beneficial for several reasons. Certain areas, for instance, may always lack coverage. The cellular network may be overloaded, or too expensive for some users, or mobile providers may decide to block P2P. Also, in certain countries where heavy censorships are in use to limit the freedom of communication, fully decentralized technologies may be a viable alternative to exchange information circumventing censorships.

III. Driving applications

A wide range of services will benefit from mobile P2P and could be run in a Wi-Fi Direct supported P2P system. The heterogeneity of mobile devices and the desirable compatibility of the P2P system with stationary hosts could create a pool of different features, with peers complementing each other. Consider the following possible areas of application:

- *Voice communication.* The success of voice-over-IP (Voip) P2P applications such as Skype shows that P2P networks are good enough to provide time-sensitive services

[1]4G network technology for mobile networks is just now starting to be tested, with the two main competitors being WiMax and LTE [11]; it is not clear when it or the other technologies will actually be more generally deployed.

such as real-time audio and video communications. Voice communication does not necessarily need a dedicated infrastructure, nor a lot of computational power or bandwidth, as shown by Skype measurements [17].

- *Text messaging.* Being a time-insensitive service, text messaging can easily be implemented using mobile P2P, through multiple hops if necessary. Each peer would buffer a message only until it can be forwarded to another peer closer to the destination. By using epidemic routing [34], a time insensitive message can travel long distances under the assumption that at least some of the peers move in space.

- *Dissemination of traffic information.* It is becoming more and more common for smartphones and other mobile devices to have GPS capabilities. This allows for dissemination of information about local traffic in a P2P fashion. For instance, the GPS function of a mobile could automatically compute the mobile's speed and infer the degree of traffic congestion. Such data could be transparently transmitted in a P2P fashion, to help other drivers choosing less congested routes [26].

- *Dissemination of emergency data.* During emergencies, many nearby people typically try to send or retrieve information through their phones and it may happen that the number of attempted connections exceeds the number of connections that a specific cellular network sector can accept. The information could be shared in real-time and received by anyone that has a mobile connected to the P2P network. The devices do not need to be connected to an access point, nor to have access to the cellular network. This works well also because many emergencies are local, and the data can be spread in a precise area.

- *Photo/video sharing during an event.* Events such as sports or meetings of any kind encourage people that share a common interest to gather together. Their proximity allows a local mobile P2P network to be formed, and thus peers can share photos or videos (or any other type of information relevant to the event). It is well known that such events represent a tough challenge for mobile networks because of the high number of connections within a small area. Video transfers, which can be especially bulky, currently represent a big portion of the mobile traffic and are quickly growing [12].

- *Last–mile connectivity.* For people living in areas that are not covered by cellular networks or by Internet connections, such as farms in rural areas, a self-supporting mobile P2P network can be an alternative, assuming there are enough nodes that act as bridges between the uncovered area and an area where an Internet connection is available.

- *Local service networks for hospitals and other organizations.* Some corporations, especially in poor countries, cannot afford expensive LANs due to the costs of setting up and managing an infrastructure. In addition, the personnel may be often in movement and their having access to standard wi-fi connections may be sporadic. A corporate, protected mobile P2P network can allow personnel to be securely connected to each other at any time and with relatively low costs.

- *Local social networking.* Social networks are becoming very popular, and the integration of location information, while still immature, will provide powerful capabilities. In addition to promoting these capabilities, the proximity of mobiles allows the dynamic creation of location-aware social networks.

- *Multi–player gaming.* Even without an Internet connection, a mobile peer could find players for a multi-player game in the local P2P network. This allows peers to play games requiring more than one player. that otherwise could not be played alone.

IV. Current state of P2P support for mobiles

Currently, P2P exists in mobile networks in three different forms.

- *P2P protocols designed for stationary hosts but used in mobile broadbands.* Many subscribers use their mobile broadbands through USB cards connected to laptops and PCs. These users often run P2P applications designed for wired networks, often expecting comparable performance with the same application running on a computer connected to a wired network [2]. This can be highly inefficient because traditional P2P protocols are not optimized for mobile networks, which clearly have different parameters and capabilities. Additionally, this approach does not take advantage in any way of the added features and capabilities that mobile devices can have compared to stationary hosts, such as mobility, ubiquity, and sensing.

- *P2P protocols where mobile devices are considered weaker nodes and thus have to rely on stationary hosts.* In several P2P systems (e.g., [18] and projects based on JXME, JXTA in Java Micro Edition), it is assumed that mobiles need stationary hosts that relieve them from the computational load derived from being part of the P2P network. Mobiles typically have their queries to and from other peers mediated by a stationary node, which acts like a proxy. While in the past mobiles had indeed very reduced computational and storage capabilities, this is changing. At present, many mobile devices have better computational capabilities than PCs had ten years ago and are suitable for properly designed P2P technologies. Additionally, the proxy approach greatly reduces the applicability of mobile P2P systems, which cannot take advantage of the proximity of other mobile devices unless they can connect to a stationary host running the P2P protocol.

- *Ad-hoc and application-specific P2P systems.* A large number of different P2P systems for ad-hoc networks have been proposed, either relying on existing protocols, such as Bittorrent ([33], [13]) or adopting new ones (e.g. [26], [30]). The fragmentation in P2P protocols for ad-hoc networks, along with the incompatibility of the different protocols, does not maximize the opportunities for cooperation of the devices. In fact, a device cannot easily find other nodes running the same P2P protocol and, since mobile P2P heavily relies on the high number and proximity of peers for efficiency, this represents a significant disadvantage. Additionally, since there is no widely established middleware framework, each new P2P application needs to be developed from scratch.

A mobile P2P middleware framework may represent a significant improvement towards an efficient mobile P2P system. A middleware framework can support basic building blocks for protocols, and thus allow for the intercommunication of services and applications with different purposes built on the same compatible protocols. This implies that mobiles running different applications, but using the same protocol, could still cooperate on various tasks that are vital to the P2P network (e.g., query routing and peer/resource discovery). Additionally, new applications do not need to be developed from scratch, but instead can rely on the existing building blocks, and only the application-level development is necessary for a new service. Tailoring a P2P system on a mobile environment also means that it can be designed to take full advantage of the capabilities that mobiles have, such as mobility, ubiquity and sensing. At the same time, it can be optimized to take into account the different features that mobile networks exhibit.

V. Mobile-optimized mobility-optimized JXTA

Mobiles have several advantages over stationary hosts, mostly deriving from their capability for mobility. Since mobile devices are used in a variety of contexts, they can also carry information to and from each context. Additionally, mobiles can be tailored to the needs of their owner and of the surrounding environment. While a number of mobile P2P applications have been proposed, there are very few existing middleware frameworks, and they typically do not take the physical location into account.

JXTA, an open-source middleware system developed by Sun Microsystems in 2002 [14], is a good basis for versatile mobile P2P middleware that can be adapted to emerging technologies such as Wi-Fi Direct. The additions and extensions that we are developing take advantage of the added functionalities of mobiles, and at the same time, seek to improve the performance of JXTA-based networks in highly dynamic environments.

Traditionally, JXTA is organized in three basic layers. The core layer includes mandatory JXTA functionalities that are strictly necessary for the JXTA network to function properly. The service level includes fundamental higher-layer services, designed for flexibility and extensibility. The application layer includes a variety of applications that can be developed on top of the other two layers.

Figure 1 shows the structure of our mobile-optimized, mobility-optimized JXTA. The orange blocks represent parts of our system that differ from traditional JXTA. At the core layer, our focus is on JXTA's grouping abstraction. In JXTA, groups of peers provide a minimum set of given services and/or share specific interests. The group structure improves the performance and limits the load on each node by restricting the scope and the number of peers to query. We extend the peer group structure so that, in a true P2P communication scenario, that grouping criteria can be based on the physical location of nodes. This grouping extension promotes good performance by minimizing the number of hops that messages travel. Note that scoping is not restricted since a peer may be a member of both "local" and "remote" groups, and thus can also act as an intermediary for nearby nodes that only have a direct (as in Wi-Fi Direct) connection. Other core layer features, such as peers publishing XML-represented advertisements for resources (peers, groups, pipes and communications services), assigning IDs, and the various levels of support for authentication and privacy, remain the same as in the traditional JXTA.

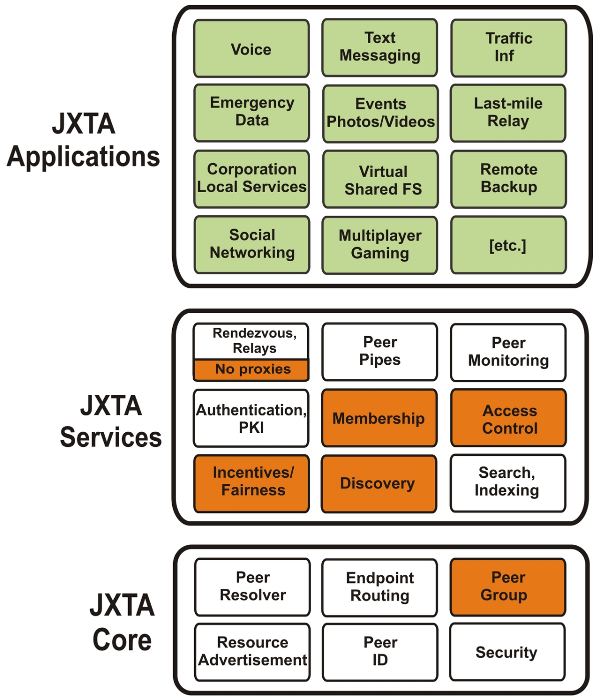


Figure 1. The three layers in our mobile-optimized JXTA. The colored blocks represent components that we either modify from the traditional JXTA (in the core and service layers) or introduce (in the application layer).

In JXTA, peers can run rendezvous and relay services. Rendezvous peers improve the efficiency of the network, as they maintain global advertisement indexes and play an important role in resource discovery. Relay peers are used to communicate with unreachable peers (i.e., those behind firewalls or NATs). The typical JXTA approach towards supporting minimal-edge peers (traditionally identified with mobile resource-limited devices) has been to establish proxy peers that mediate all the communication with the minimal-edge peers to allow them to access JXTA functionalities. Proxy peers are no longer absolutely necessary in our mobile-optimized system. In fact, current devices are powerful enough to be part of a P2P network without the need for stationary hosts to act as a proxies.

Consequently, at the service layer, we focus on the following extensions. In traditional JXTA, the membership service is used to securely establish identities and trust within a peer group, while the access service is used to validate requests of resources, made by one peer to another. In a highly dynamic environment where mobiles are continuously moving and thus causing changes to the structure of groups, it is virtually impossible to enforce strong rules for membership while maintaining efficiency and flexibility. Since having more peers in the network can lead to better performance, resilience and availability of services, it is convenient to have a loose policy for membership. Thus, we allow location/proximity to be a major criteria for access control in our system, as peers that happen to be geographically close represent an opportunity for efficient connections. Blacklists are used for peers that misbehave. Additionally, existing JXTA cryptographic tools can be customized to ensure an adequate level of privacy and authenticity, implemented at the peer level, rather than at the group level. Each peer is thus able to independently decide whether it wants secure connections, possibly only with trusted peers.

Traditional JXTA does not provide any specifications to ensure fairness or provide incentives. However, whether services are provided for free or for payment, incentives represent a fundamental component to encourage P2P resource sharing. Relying on the generosity of each node can lead to widespread free-riding behavior, especially in a mobile context, where nodes typically have to take their battery usage into account. Incentives can be economic rewards, or they may be established through a more general fairness policy in the exchange of services. For instance, a peer can be granted access to a service only if it also provides that same service, or in a more flexible scheme, it can be granted access to some given number of different services if it also provides a given number of similar ones. Due to the variety of services that JXTA can support, and given the dynamicity of P2P interactions, our approach is that each peer be rewarded for any service provided to the P2P community in general, and not to specific peers only. A similar concept has been used by Efstathiou et al. in the design of their P2P Wireless Network Confederation (P2PWNC) [27], and differs from the incentive strategy adopted by many popular P2P systems (e.g., Bittorrent and Edonkey). We take advantage of code mobility feature of JXTA [22], [25] which allows services to be added dynamically, not only by loading a pre-installed module but also by downloading from a remote source, such as another peer, the code needed to run a service. This provides further flexibility for a fairness scheme.

The discovery protocol is used in JXTA to find advertisements published by other peers within a group. To provide better availability of content, it is convenient that the discovery protocol also include the capability of retrieving content from peers outside the "local" group. Peers connected to the Internet through a wi-fi connection could, for example, retrieve content located far away and inject it in the local P2P network. A similar approach is described in [33].

We make use of JXTA support for a fully decentralized search infrastructure, which is based on resource indexing through distributed hash tables (DHT). Despite that this requires more computational resources than other search methods such as flooding, it is much more efficient as to network usage, which is particularly important in mobile broadbands.

VI. Related Work

Mobile P2P applications have typically not considered the geographical location as a prime concern. We are not aware of any P2P system that is specifically designed to optimize Wi-Fi Direct connections.

A number of potential mobile P2P applications have been proposed, for either new (e.g., [26], [30]) or existing (e.g. [33], [13]) protocols. As for middleware solutions, we refer to [28] and [32] for an extensive list. Many of the existing middleware frameworks present similarities, especially in the communication and resource discovery schemes, while the different scopes of each middleware determine some of the intrinsic differences among them, such as optimizations and sets of offered primitives. The great majority of the the middleware frameworks in [28] and [32] are currently discontinued.

Regarding JXTA-related mobile P2P, several works by Bisignano et al. present a JXTA-based middleware for MANETs (Mobile Ad hoc NETworks) [22], [19], [21], [23], [20]. In these works, a software layer is added on top of JXTA, with the main purpose of creating optimizations for advertisements and connections in MANETs. Results shown through simulations appear to be encouraging.

In [35], a set of improvements is presented to specifically optimize file sharing by using JXTA-Overlay. The latter is a project that builds an overlay on top of JXTA to offer a set of commonly needed functionalities and basic primitives [15]. In particular [35] proposes a distinction among different types of advertisements, to be handled differently, and, according to JXTA-Overlay specifications, the use of *broker* peers to govern the JXTA P2P network. ContextTorrent [29] is a semantic context management framework for distributed searches among local and remote context-aware applications. It is implemented in JXME, which has been ported to Android. Finally, JXBT [24] implements a JXME infrastructure using Bluetooth. It enhances basic Bluetooth and overcomes some of its limitations, such as the reduced number of interconnectable devices and limited transmission range.

VII. Conclusion

In this paper, we have highlighted how P2P is currently a widespread reality in both wired and wireless networks, and the reasons why a new form of P2P, one that exploits upcoming technologies for fully distributed connections, such as Wi-Fi Direct, can be beneficial for both consumers and service providers. We also presented some of the applications that become possible with new forms of P2P. We described the main building blocks of a new JXTA-based P2P middleware architecture, optimized not only for mobiles but also for mobility and location awareness. We are in the process of implementing the system in Android and we plan on testing it using Wi-Fi Direct as soon as it becomes available.

References

[1] http://www.wi-fi.org/files/20091019_Wi-Fi_Direct_FAQ.pdf.

[2] http://gigaom.com/2009/07/21/will-p2p-soon-be-the-scourge-of-mobile-networks.

[3] http://arstechnica.com/tech-policy/news/2010/04/just-like-comcast-rcn-accused-of-throttling-p2p.ars.

[4] http://virginmobile.custhelp.com/app/answers/detail/a_id/290/̃/peer-to-peer-%2F-p2p-shaping.

[5] http://www.ovum.com/go/content/c,57072.

[6] http://www.allot.com/Allot_MobileTrends_Report_Shows _Significant_Growth.html.

[7] http://gigaom.com/2009/07/12/what-will-carriers-do-when-the-data-gravy-train-derails.

[8] http://gigaom.com/2008/11/06/att-buys-wayport-to-keep-iphone-users-happy.

[9] http://www.wi-fi.org/news_articles.php?f= media_news&news_id=909.

[10] http://www.sci-tech-today.com/story.xhtml?story_id= 10200AGALOB0.

[11] http://technologizer.com/2009/05/20/lte-vs-wimax-the-4g-wireless-war.

[12] http://www.fiercewireless.com/special-reports/mobile-video-traffic-alleviating-capacity-crunch.

[13] http://amorg.aut.bme.hu/projects/symtorrent.

[14] https://jxta.dev.java.net/.

[15] https://jxta-overlay.dev.java.net.

[16] Traffic inspection for visibility, control and new business opportunities. Technical report, Ericcson, 2008.

[17] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *INFOCOM 2006*.

[18] A. Berl, I. Dedinski, E. Georgiew, and H. D. Meer. Mobile p2p: Turning heterogeneity to an advantage.

[19] M. Bisignano. Jmobipeer: a middleware for mobile peer-to-peer computing in manets. In *Distributed Computing Systems Workshops, 2005*.

[20] M. Bisignano. A jxta compliant framework for mobile handheld devices in ad hoc networks. In *ISCC 2005*.

[21] M. Bisignano. Expeerience: a jxta middleware for mobile ad hoc networks. In *Third International Conference on P2P Computing*, 2003.

[22] M. Bisignano. Design and development of a jxta middleware for mobile ad-hoc networks, 2004.

[23] M. Bisignano. An infrastructure-less peer-to-peer framework for mobile hand- held devices. *European Transactions on Telecommunications*, 2004.

[24] C.Blundo and E. Cristofaro. A bluetooth-based jxme infrastructure. In *Proceedings of the 9th International Symposium on Distributed Objects, Middleware and Applications*, 2007.

[25] R. Y. Chen and B. Yeager. Java mobile agents on project jxta peer-to-peer platform. *Hawaii International Conference on System Sciences*, 9, 2003.

[26] S. Dornbush and A. Joshi. Streetsmart traffic: Discovering and disseminating automobile congestion using vanet's. In *Vehicular Technology Conference, 2007*.

[27] E. C. Efstathiou and G. C. Polyzos. A self-managed scheme for free citywide wi-fi. In *WoWMoM 2005*.

[28] S. Hadim. Trends in middleware for mobile ad hoc networks. 2006.

[29] D. H. Hu. A semantic context management framework on mobile device. *Second International Conference on Embedded Software and Systems*, 2009.

[30] N. B. Niraula, K. Kanchanasut, and A. Laouiti. Peer-to-peer live video streaming over mobile ad hoc networks. In *IWCMC '09*, pages 1045–1050, 2009.

[31] R. Padovani and QUALCOMM. A day in the life of a smartphone, http://www.calit2.net/newsroom/article.php?id=1683, 2010.

[32] G. Paroux. A survey of middleware for mobile ad hoc networks. Technical report, Telecom Paris, 2007.

[33] M. K. Sbai. Bithoc: Bittorrent for wireless ad hoc networks, 2008.

[34] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. *Duke Tech Report*, 2000.

[35] F. Xhafa. Extending jxta protocols for p2p file sharing systems. In *International Conference on Complex Intelligent and Software Intensive Systems, 2008*.

All links were last checked on 7/7/2010.

Resource Assignment in Box-Based P2P Video-on-Demand Systems

Juan Pedro Muñoz-Gea, Josemaria Malgosa-Sanahuja, Pilar Manzanares-Lopez, Pedro Jose Piñero-Escuer
Department of Information Technologies and Communications
Polytechnic University of Cartagena
Campus Muralla del Mar, 30202 Cartagena, Spain
Email: {juanp.gea, josem.malgosa, pilar.manzanares, pedrop.escuer}@upct.es

***Abstract*—Video-on-demand (VoD) systems that make use of the storage capacities at set-top boxes to assist the streaming servers have been introduced recently. In these systems, videos are separated into substreams and randomly stored in boxes, which are organized in a P2P network. By this way, this architecture combines the load balancing and fault tolerating features of P2P systems with the stability of set-top boxes, since they usually have much longer online time than traditional PC based peers. The VoD service provider controls two different resources: the allocation of substreams to the selected boxes, and the parameters associated to the streaming servers (number, bandwidth, storage, placement in the network). In this work, we present the Resource Assignment Problem (RAP) which tries to optimize the previous resources in order to reduce the associated costs. This problem is presented as a linear programming problem and it is solved using the MATLAB optimization toolbox. We have evaluated the influence of the bandwidth and the placement of the server in the Internet.**

***Keywords*-VoD; P2P; set-top box; linear programming.**

I. Introduction

Nowadays, there exist two main categories of video-on-demand (VoD) streaming over IP. The first category is the Internet Protocol Television (IPTV) that assumes video services delivery in a managed network, usually deployed and operated by broadband providers. On the other hand, there is the Internet video streaming which relies on 3rd party servers (usually Content Delivery Networks, CDNs) to stream multimedia content to end-users attached to Internet. This category of video streaming architectures is gaining unprecedented attraction, fuelled by the fact that the service delivery is well balanced and non-biased towards any main stakeholder (such ISP in the IPTV world) [1]. In addition, due to the open nature of Internet, the barrier to enter the Internet streaming market is much lower than in the IPTV case.

A key characteristic of CDNs is that they comprise of a large number of caches distributed throughout the network to facilitate more speedy access to content. Load balancing techniques are achieved by dynamically redirecting VoD request to appropriate caches based on the load and proximity of each cache to the end-user. However, this approach has multiple drawbacks, among which complexity of deploying data centers, power consumption, and lack of scalability are the most critical ones [2].

Peer-to-Peer (P2P) has recently emerged as the main approach to increase the scalability of streaming servers. However, one serious problem with using a conventional P2P solution, at least in the current technological landscape, is the limited upload bandwidth of each peer [3]. In addition, the available bandwidth has to be shared among different applications. The result is that a video request would require finding a great number of peers that have already downloaded that video, which can be highly unlikely, in order to get the uplink bandwidth related to the video bit rate.

Several works [3], [4], [5], [6] have suggested to making use of the storage capacities at set-top boxes (STBs) or residential gateways of clients, combined in a P2P approach, to assist the CDN of the video provider. In these systems, movies are broken into small substreams which are pre-cached through the P2P network during off-peak hours. Even though each peer can only afford to contribute limited upload bandwidth, the aggregate is sufficient to support high definition delivery. For example, ten peers with pre-cached content can serve substreams at a steady state rate of 200 Kbps to satisfy a 2 Mbps video request from a peer.

Therefore, in this scenario, the VoD service provider controls two different resources: First of all, the allocation of substreams to the selected boxes. A good allocation can represent a substantial saving when considering the network cost of the streaming sessions, i. e. the cost to transport the data from the boxes hosting the substreams to the client. And secondly, the parameters associated to the streaming servers, for example: number, bandwidth, storage capacity and placement in the network.

In this paper, we present the Resource Assignment Problem (RAP) which tries to optimize the allocation of substreams and the parameters associated to the streaming servers in order to reduce the associated costs: the number of traversed routers to transport the data, the use of the access networks of network operators and the use of the streaming servers.

The rest of the paper is organized as follows. Section II discusses several related works. Section III outlines the most remarkable issues of the system. Sections IV and V presents the Resource Assignment Problem and the results provided by the optimization tool. Finally, Section VI concludes the paper.

II. Related Works

The systems presented in [3], [4], [5] are designed to work in a managed network, deployed and operated by an IPTV provider. In all of them, the streaming server performs the main intelligence in the system because it allocates resources for each incoming VoD request. In order to perform this, it tracks two main resources: (i) the currently available uplink bandwidth at each peer, and (ii) the content stored in each peer. When a given peer requests a specific content, a VoD request is sent from the peer to the server. The server looks up its database to determine the most appropriate set of contributing STBs. If during the streaming session the available bandwidth of one of the peers changes, and the receiving peer is unable to properly recover the video, it sends a request to the server and it answers the identity of a new candidate.

In the previous proposals it is possible that the servers have a global knowledge about the system because they work in a managed network. However, controlling the state of all the peers on the Internet can be a very costly task. In our proposal, it is assumed that during the substream pre-caching process the peers also receive the identity of the STBs they have to contact with to download every video. By this way, the clients do not have to contact the server to start each streaming session. On the other hand, as some peers may fail, redundancy is necessary to guarantee smooth delivery. Before breaking the movie into substreams, we use an erasure code with a threshold (say 80 %). That is, the movie file is broken into segments, and each segment is encoded as, say, 12 blocks, and any 10 of which are sufficient to reconstruct the segment.

On the other hand, in [6] authors consider a global Internet scenario, and they study the allocation of substreams to the selected boxes in order to reduce the network cost. However, they only consider the network cost as the number of traversed routers. In our work we also want to take into account the cost associated to the use of the access networks and streaming servers. In addition, authors assume that when a new box joins the system, it iteratively explores the nearest boxes in the network until it discovers all the necessary substreams minus one of them, and then it receives the unassigned substream from the server. That is, the new box is the responsible to locate the complementary substreams that it needs using an iterative process. In our system, boxes do not have this extra cost, because they receive the identity of their complementary boxes from the streaming servers.

Finally, in [6] authors do not take into account that streaming servers can also provide substreams as STBs. In our system, we take into account that streaming servers can be considered as usual STBs during the substream pre-caching process. By this way, we can take advantage of the characteristics of streaming servers to reduce the cost associated to the streaming sessions.

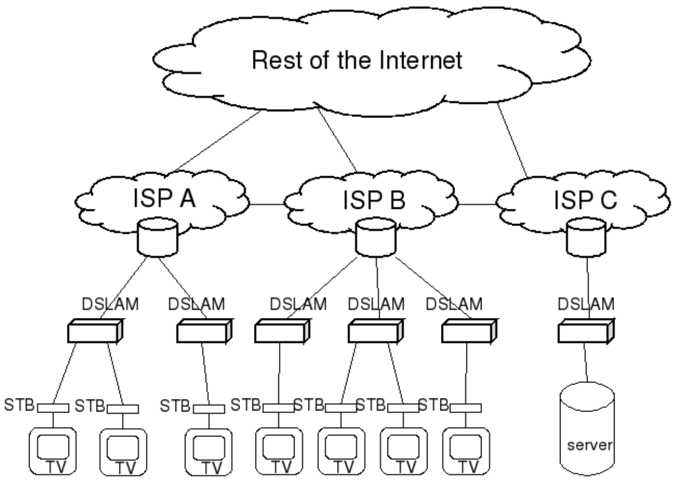


Figure 1. Internet scenario of the system.

III. Scenario and Architecture

A. Scenario

Research in P2P streaming typically considers Internet at a logical level: it represents the Internet as an abstract cloud and only considers the capacity of the content server and the characteristics of the access links to related hosts. This view of the Internet is referred as the ”cloud model”. In contrast to the cloud model, the physical model considers the network architecture and bandwidth constraints of the underlying links and network devices. A key insight of [3] is that using the ”cloud model” for P2P streaming is overly simplistic. More realistic results can be obtained by considering the network at the physical infrastructure level. Authors show that the cloud model of the Internet frequently used in simulation studies of peer-to-peer systems may drastically overstate the benefits of P2P video content delivery. Thus, one must consider physical network infrastructure to obtain more reliable results.

Figure 1 represents the scenario where our system is deployed. The clients are located in the networks managed by several ISPs. In addition, in order to simplify the proposed model, the streaming server of the video provider is located in a different network, where there are not any clients. We assume that the video provider knows the network of the box, i.e. the identifier of the first router that connects this box to the Internet, and the network cost between every two routers (see [7], [8] for some techniques that may be used).

B. Overall Architecture Description

We are going to consider separately one video of the top 5 % popular videos, in the same way that [3]. First of all the video provider breaks the video into segments, and then it applies erasure coding to every segment with a specific rate k/n. It means that each segment is encoded as n blocks $(b_1, b_2, ., b_n)$ and any k of them will be enough to reconstruct the segment. After that, it creates n substreams joining the b_i blocks of each segment and it allocates one substream to each box following a serial dispatching strategy. When the n substreams have been allocated, the process is repeated iteratively till all the boxes have a different substream. In addition, every peer also receives a neighbor table, which represents the identity of the n boxes it has to contact with to get the video (taking into account that it is only necessary to establish k connections). This neighbor table is created by the service provider taking into account the results of the optimization algorithm, which will be presented in next section.

The download of consecutive segments is managed by a sliding window algorithm. Counting from the moment of play-back, a segment may be downloaded from a STB if it is farther than a time parameter. That is, substreams are usually downloaded from other STBs in the network. However, nearer segments which have not been already downloaded from other peers can be downloaded from the streaming servers. Because segment reconstruction can occur only when k blocks are downloaded in their entirely, buffering is essential in our approach: a downloading peer delays movie rendering until the first segment is downloaded and reconstructed, and then downloads the blocks of the future segments while the current segment is being rendered. Because of buffering, when a downloading peer detects failure of a peer used for movie delivery, there is usually significant time left until the interrupted block will be needed for viewing.

IV. Resource Assignment Problem

In this scenario, the VoD service provider controls two different resources: First of all, the allocation of substreams to the selected boxes. And secondly, the parameters associated to the streaming servers, for example: number, bandwidth, storage capacity and placement in the network. In this section, we present the Resource Assignment Problem (RAP) which tries to optimize the allocation of substreams and the parameters associated to the streaming servers in order to reduce the associated costs.

The RAP problem is presented as a linear programming problem. The decision variable $\overline{x}$ represents the number of substreams that the clients in every network have to get from each network. For example, in a scenario with 3 networks

$$\overline{x} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{bmatrix}$$

where x_{ij} represents the number of substreams that the clients in network i have to get from clients located in network j. The streaming server of the video provider is located in network 3, where there are not any clients. That is the reason why elements x_{3j} do not appear in $\overline{x}$.

The optimization problem can be characterized asfollows:

$$\min_{\overline{x}} \quad C_T(N, \overline{x}, \overline{v}) + C_S(N, \overline{x}) \tag{1}$$

$$s.t. \quad \sum_{j=1}^{N} x_{ij} = (k-1) \cdot p \cdot c_i \tag{2}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{N} x_{ij} \cdot r + x_{ii} \cdot 2 \cdot r + \sum_{\substack{j=1 \\ j\neq i}}^{N} x_{ji} \cdot r \leq BW_i \tag{3}$$

The objective function, Equation 1 seeks to minimize the cost of transport and the cost of streaming from the central server:

$$C_T(N, \overline{x}, \overline{v}) = \sum_{i=1}^{N-1} \sum_{j=1}^{N} \theta \cdot v_{ij} \cdot x_{ij} \tag{4}$$

$$C_S(N, \overline{x}) = \sum_{i=1}^{N-1} \sum_{j=1, j\in S}^{N} \beta \cdot x_{ij} \tag{5}$$

where θ is the cost of one hop in the transport network, v_{ij} represents the number of hops between the networks i and j and β is the cost of streaming one substream from the central server. They are called *cost of transport* and *cost of servers* respectively.

Constraint 2 is defined for every network i, and it represents the total number of substreams that are downloaded by the clients of a specific network in the peak hour. Constraint 3 is also defined for every network i, and it represents the total bandwidth consumed in network i. Every substream downloaded from a different network consumes a bandwidth similar to its streaming rate, whereas the substreams downloaded from the same network consume a double bandwidth. Finally, the substreams uploaded from a specific network with destination in other network also consume the corresponding upload bandwidth in the origin network. In the constrainsts k is the number of substreams, p is the percentage of active clients in the peak hour, c_i is the number of clients in the network i, N is the number of networks, r is the streaming rate of substreams and BW_i is the bandwidth constraint in the network i. They are called *number of substreams* constraint and *network bandwidth* constraint, respectively.

To solve the problem we use the MATLAB Optimization Toolbox. Specifically, we use the `linprog` program which solves linear programming problems.

V. Performance Evaluation

A. Introduction

In this section we present the results provided by the optimization tool in three small scenarios. First of all, we try to make readers understand that solving this problem is not as obvious as they could think. We are going to assume that there are three networks, with clients located in networks 1 and 2, and the server located in network 3. The number of hops between the 3 networks is represented in matrix H, where every element (H_{ij}) represents the number of hops between the networks i and j

$$H = \begin{pmatrix} 0 & 8 & 8 \\ 8 & 0 & 2 \\ 8 & 2 & 0 \end{pmatrix}$$

The erasure coding rate is assumed to be 10/12, therefore, the number of substreams that every VoD request needs is $k - 1 = 9$, because every STB has one of the necessary substreams. Every substream has a streaming rate of 200 Kbps and the bandwidth restriction of networks 1 and 2 is set to 18 Mbps. In addition, we assume that during the peak hour only a 75 % of the clients located in a network are using the VoD service. Finally, parameters β and θ are set to 1.

1) Scenario 1: The number of clients of both networks is set to 7. Therefore, the number of active clients during the peak hour is 5, and the total number of substreams that they need is $(10 - 1) \times 5 = 45$. The result provided by the optimization tool is the following

$$\overline{x} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{bmatrix} = \begin{bmatrix} 45 \\ 0 \\ 0 \\ 0 \\ 45 \\ 0 \end{bmatrix}$$

This result tells us that all the substreams that the clients in both networks need must be obtained from their networks. On the other hand, we have to take into account that 45 substreams is the maximum number of substreams that the clients of a network can obtain from their own networks. These substreams consume a double quantity of bandwidth, therefore they correspond with a bandwidth of $45 \times 200 \times 2 = 18$ Mbps which corresponds with the bandwidth restriction.

2) Scenario 2: The number of clients in network 1 is set to 9, and the number of clients in network 2 is set to 5. Therefore, the number of active clients during the peak hour is 7 and 4, and the total number of substreams that they need is 63 and 36, respectively. The result provided by the optimization tool is the following

$$\overline{x} = \begin{bmatrix} 27 \\ 18 \\ 18 \\ 0 \\ 36 \\ 0 \end{bmatrix}$$

This result tells us that the 36 substreams that clients in network 2 need can be obtained from its own network. On the other hand, the 63 subtreams that clients in network 1 need must be distributed in the following way: 27 must be obtained from its own network, 18 from network 2, and 18 from the server. The algorithm tries to get the maximum number of substreams from clients located in network 2. Taking into account that this network is consuming $36 \times 200 \times 2 = 14400$ Kbps, there are 3600 Kbps free. It corresponds to $3600/200 = 18$ substreams. Finally, the algorithms gets other 18 substreams from the server. Explaining this last result is a bit more complicated, but it can be seen that with this configuration the bandwidth consumed in network 1 corresponds to the maximum. Let's see: the 27 substreams obtained from network 1 consume $27 \times 200 \times 2 = 10800$ Kbps. On the other hand, the 36 substreams obtained from network 2 and the server consume $36 \times 200 = 7200$ Kbps. The sum of both of them is 18000 Kbps which is the maximum bandwidth. Therefore, we deduce that the 18 substreams obtained from the server correspond to the minimum number of substreams to fulfill the bandwidth restriction.

3) Scenario 3: The number of clients in network 1 is set to 5, and the number of clients in network 2 is set to 9. Therefore, the number of active clients during the peak hour is 4 and 7, and the total number of substreams that they need is 36 and 63, respectively. The result provided by the optimization tool is the following

$$\overline{x} = \begin{bmatrix} 36 \\ 0 \\ 0 \\ 0 \\ 27 \\ 36 \end{bmatrix}$$

This result tells us that the 36 substreams that clients in network 1 need can be obtained from its own network. On the other hand, the 63 subtreams that clients in network 3 need must be distributed in the following way: 27 must be obtained from its own network and 36 from the server. Let's analyze this result. The first option would be to try to obtain the 63 substreams from network 2, however, it is impossible because these substreams consume a double bandwidth. Therefore, the algorithm tries to get the maximum number of substreams from the server, because the cost is less than obtaining them from the clients in network 1.

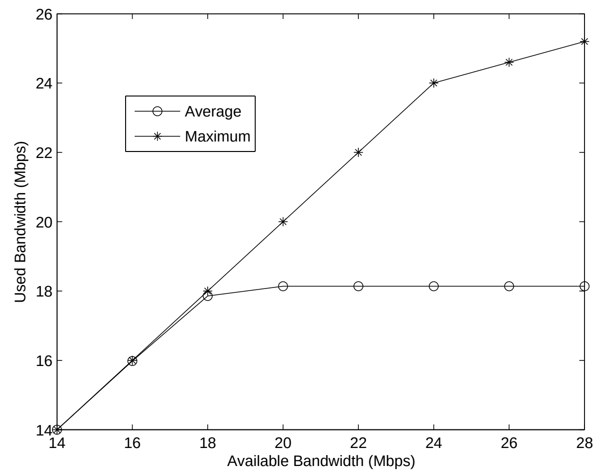


Figure 2. Bandwidth used in the networks.

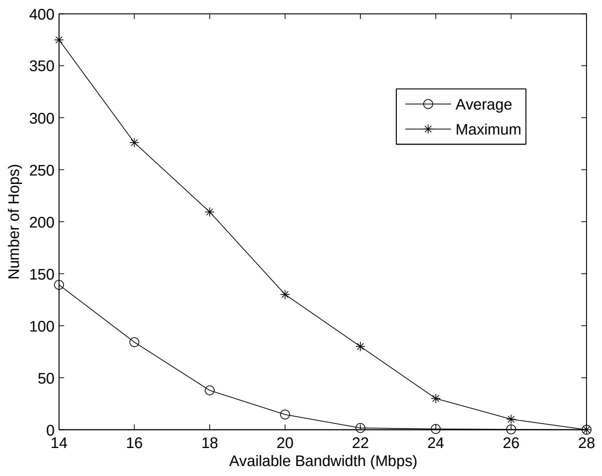


Figure 4. Number of necessary hops.

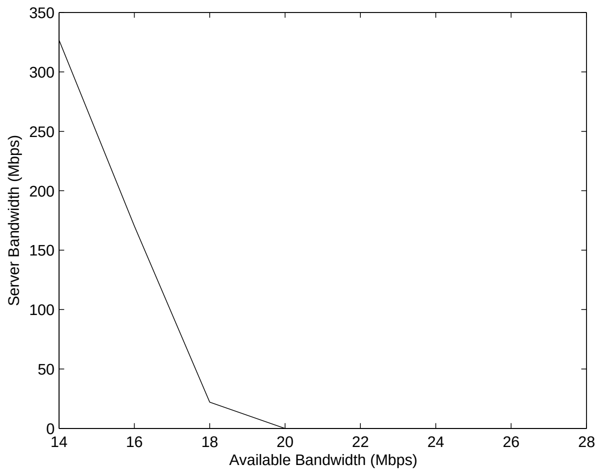


Figure 3. Bandwidth used by the server.

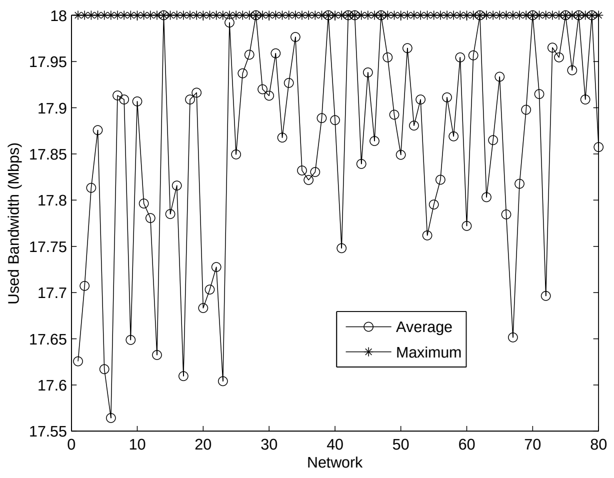


Figure 5. Bandwidth used in the networks.

B. Results

In this section, we show the influence of the *network bandwidth* constraint and the placement of the server in the Internet on the bandwidth used in the networks, the bandwidth used by the server and the number of necessary hops to satisfy the VoD request. A router-level topology resulted from the *Network Cartographer (nec)* [9] is used as our network topology. We randomly select 80 edge routers with degree equal to one, and we attach to each of them a number of elements following a normal distribution $N(7, 1.4)$. We assume that during the peak hour only a 75 % of the clients located in a network are using the VoD service. On the other hand, the erasure coding rate is assumed to be 10/12, and every substream has a streaming rate of 200 Kbps. Finally, parameters β and θ are set to 1.

1) Influence of the network bandwidth constraint: Figures 2, 3 and 4 show the influence of the *network bandwidth* constraint. In these experiments, the bandwidth restriction of networks changes from 14 to 28 Mbps. As it can be seen in Fig. 2, the average bandwidth increases linearly till a bandwidth restriction of 18 Mbps, but from that point it is stabilized around 18 Mbps. On the other hand, the maximum bandwidth used increases linearly till a bandwidth restriction of 24 Mbps, and after that the increasing rate is reduced. The bandwidth used by the server (represented in Fig. 3) is reduced when the network bandwidth constraint increases, and it is almost 0 from a 20 Mbps local bandwidth restriction. Finally, Figure 4 shows the influence of this parameter on the number of necessary hops that every VoD requests needs to be satisfied.

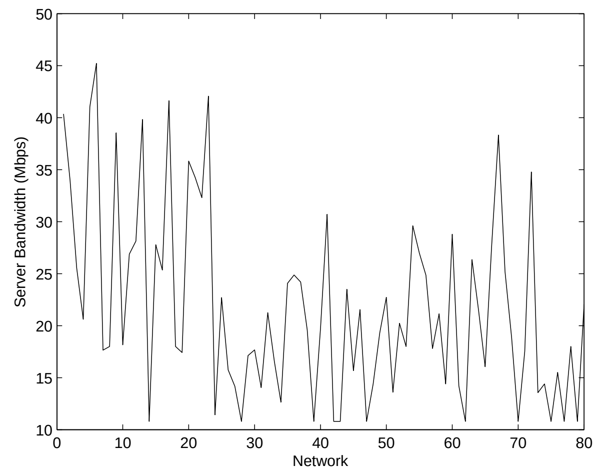


Figure 6. Bandwidth used by the server.

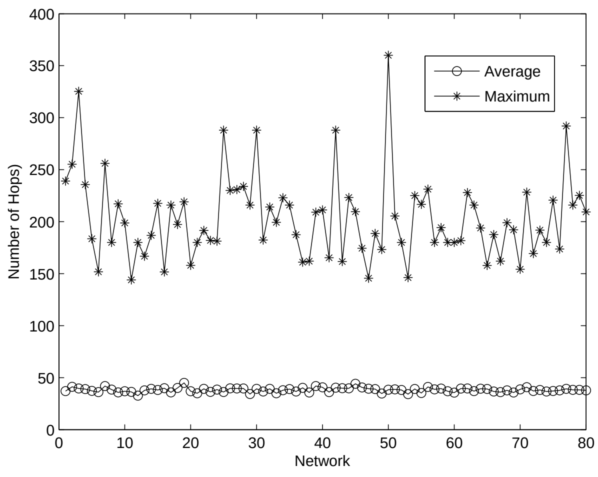


Figure 7. Number of necessary hops.

2) Influence of the placement of the server: Figures 5, 6 and 7 show the influence of the the placement of the server in the Internet. In these experiments, the bandwidth restriction of networks is fixed to 18 Mbps. The x-axis of the previous figures represents the identifier of the network in which the streaming server is located, and it changes from 1 to 80. These results can help us to find out what the best location of the server is in order to reduce the associated costs.

VI. CONCLUSION

Box-based P2P VoD systems make use of the storage capacities at STBs, combined in a P2P approach, to assist the CDN of the video provider. In these systems, movies are broken into small substreams which are pre-cached through the P2P network during off-peak hours. In this scenario, the VoD service provider controls two different resources: the allocation of substreams to the selected boxes, and the parameters associated to the streaming servers.

In this paper, we have presented the Resource Assignment Problem (RAP) which tries to optimize the allocation of substreams an the parameters associated to the streaming servers in order to reduce the associated costs: the number of traversed routers to transport the data, the use of the access networks of network operators and the use of the streaming servers.

ACKNOWLEDGMENT

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of “Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)”.

REFERENCES

[1] F. Thouin and M. Coates, “Video-on-demand networks: Design approaches and future challenges,” *IEEE Network*, vol. 21, no. 2, pp. 42–48, 2007.

[2] N. Laoutaris, P. Rodriguez, and L. Massoulie, “Echos: edge capacity hosting overlays of nano data centers,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, pp. 51–54, 2008.

[3] Y. Chen, Y. Huang, R. Jana, H. Jiang, M. Rabinovich, J. Rahe, B. Wei, and Z. Xiao, “Towards capacity and profit optimization of video-on-demand services in a peer-assisted iptv platform,” *ACM Multimedia Systems*, vol. 15, no. 1, pp. 19–32, 2009.

[4] A. Nafaa, S. Murphy, and L. Murphy, “Analysis of a large-scale vod architecture for broadband operators: A p2p-based solution,” *IEEE Communications Magazine*, vol. 46, no. 12, pp. 47–55, 2008.

[5] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, “Push-to-peer video-on-demand system: Design and evaluation,” *IEEE Journal of Selected Areas in Communications*, vol. 25, no. 9, pp. 1706–1716, 2007.

[6] Y. Chen, B. Han, J. Leblet, G. Straub, and G. Simon, “Network-friendly box-powered video delivery system,” in *ITC 21 : 21st International Teletraffic Congress*, 2009, pp. 1–8.

[7] B. Wong, A. Slivkins, and E. G. Sirer, “Meridian: a lightweight network location service without virtual coordinates,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 85–96, 2005.

[8] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, “Avoiding traceroute anomalies with paris traceroute,” in *IMC ’06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 153–158.

[9] D. Magoni and M. Hoerdt, “Internet core topology mapping and analysis,” *Computer Communications*, vol. 28, no. 5, pp. 494 – 506, 2005.

Towards Peer-to-Peer Virtualized Service Hosting, Discovery and Delivery

David Bailey and Kevin Vella
University of Malta
Msida, Malta
Email: david@davidbailey.info, kevin.vella@um.edu.mt

***Abstract*—This paper introduces a peer-to-peer framework for providing, locating and consuming distributed services that are encapsulated within virtual machines. We believe that the decentralized nature of peer-to-peer networks acting in tandem with techniques such as live virtual machine migration and replication facilitate scalable and on-demand provision of services. Furthermore, the use of virtual machines eases the deployment of a wide range of legacy systems that may subsequently be exposed through the framework.**

To illustrate the feasibility of running distributed services within virtual machines, several Hadoop benchmarks are executed on a compute cluster running our framework, and their performance characteristics are evaluated. While I/O-intensive benchmarks suffer a penalty due to virtualization-related limitations in the prevailing I/O architecture, the performance of processor-bound benchmarks is virtually unaffected. Thus, the combination of peer-to-peer technology and virtualization merits serious consideration as a scalable and ubiquitous basis for distributed services.

***Keywords*-Virtualization, distributed systems, peer-to-peer computing, service-oriented computing**

I. Introduction

In recent years, data centre operations have experienced a shift in focus away from managing physical machines to managing virtual machines. Renewed exploration of this well-trodden path is arguably driven by virtualization's mantra of enhanced operational agility and ease of management, increased resource utilisation, improved fault isolation and reliability, and simplified integration of multiple legacy systems. Virtualization is also permeating the cluster and grid computing communities, and we believe it will feature at the heart of future desktop computers and possibly even advance a rethink of general purpose operating system architecture.

The performance hit commonly associated with virtualization has been partly addressed on commodity computers by recent modifications to the x86 architecture [1], with both AMD and Intel announcing specifications for integrating IOMMUs (Input/Output Memory Management Units) with upcoming architectures. While this largely resolves the issue of computational slow-down and simplifies hypervisor design, virtualized I/O performance will remain mostly below par until I/O devices are capable of holding direct and concurrent conversations with several virtual machines on the same host. This generally requires I/O devices to be aware of each individual virtual machine's memory regions and demultiplex transfers accordingly. We assume that this capability or a similar enabler will be commonplace in coming years, and that the commoditization of larger multi-core processors will reduce the frequency of expensive world-switches as different virtual machines are mapped to cores over space rather than time.

This paper introduces Xenos [2], a proof-of-concept implementation of a framework that enables the dynamic provision, discovery, consumption and management of software services hosted within distributed virtual machines. Xenos uses a decentralised peer-to-peer overlay network for advertising and locating service instances and factories. It also leverages techniques such as live virtual machine migration and replication to enhance operational agility and ease of management, and to lay the foundations for deploying fault-tolerant services. The primary objective is to shift the focus away from managing physical or virtual machines to managing software services.

This paper is organized as follows. Section II refers to some related work. Section III describes our proposed framework and the implemented prototype. Section IV presents an evaluation of the framework, and Section V discusses some topics for future investigation. We conclude in Section VI.

II. Related Work

The ideas presented here are influenced by the Xenoservers project [3], initiated by the creators of the Xen hypervisor. Xenoservers was designed to "build a public infrastructure for wide-area distributed computing" by hosting services within Xen virtual machines, while Xenosearch [4] locates Xenoservers using the Pastry peer-to-peer overlay network. A Xenoservers implementation is not generally available, hence our decision to build and conduct experiments with Xenos.

WOW [5] also uses a peer-to-peer overlay network to maintain self-organizing virtual links between virtual machines. IP connectivity is thus preserved across virtual machine migrations. However, WoW does not support the discovery of services on the overlay network.

Several other publications have focused on the use of peer-to-peer overlay networks to implement distributed resource indexing and discovery schemes in grid frameworks, such as [6], [7] and [8]. Wadge [9] investigates the use of peer groups to provide services in a grid, as well as transferring

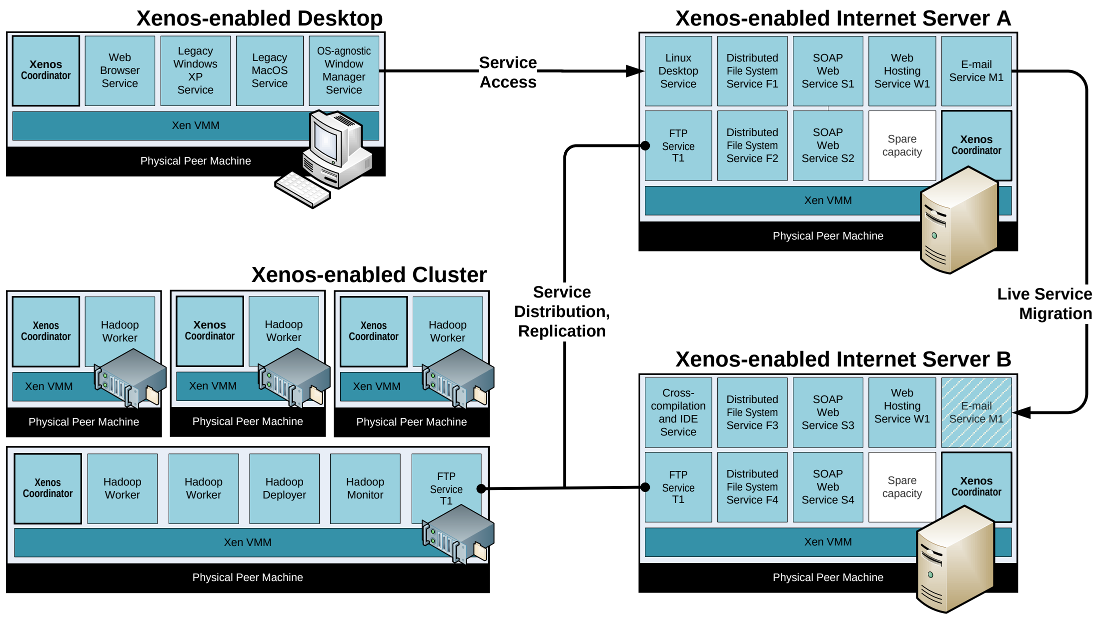


Figure 1. A selection of computing platforms running the Xenos framework and hosting several interacting services.

service code from one node to another for increased fault-tolerance and availability. This is achieved through the use of dynamically loadable non-virtualized plug-ins that can be shared by peers. SP2A [10] is a service-oriented peer-to-peer architecture which provides resource sharing in a non-virtualized grid.

The virtualized deployment of application software with specialized hardware requirements has also been explored in the literature, particularly in the context of high-performance and cluster computing: [11], [12] and [13].

III. The Xenos Framework

Xenos is built on top of Xen, a virtualization platform that has gained traction as a stable and mature virtualization solution, but any hypervisor with the appropriate hooks and programming interfaces will suffice in principle, including a hypothetical ROM-based hypervisor. The JXTA framework is currently used to maintain a peer-to-peer overlay network for service advertisement, discovery and, optionally, transport. However, we feel that a more specialized or expressive latter generation peer-to-peer framework would better fit our requirements.

A. Physiology

Figure 1 illustrates a scenario with different hardware platforms running Xenos and a variety of services. A compute cluster service enables users to dynamically create computation service instances, such as Hadoop map-reduce nodes. A pair of servers in a data centre offer web hosting, FTP, email and other services, all hosted within virtual machines and discoverable by users and other services across the Xenos cloud. Xenos also runs on a desktop computer, hosting several light-weight services (virtualized Google ChromeOS, for instance).

B. Architecture

Each Xenos-enabled physical machine runs the Xen hypervisor using a paravirtualized Linux kernel in Domain 0, which is a privileged domain capable of controlling the guest domains that will host services on the same physical machine. The Xenos coordinator is a Java application that executes in Domain 0 whose primary function is to incorporate the physical machine into Xenos's peer-to-peer overlay network and advertise services running on that physical machine. Services running within guest domains do not normally join the overlay network directly, but are registered with the coordinator in Domain 0 which acts as a 'notice board' for all local services. Xenos provides an XML-RPC programming interface for users and services to discover, locate and manage services.

Service delivery itself may be accomplished without the involvement of Xenos, and is not restricted to any particular network protocol or address space. However, the direct use of network protocols beneath layer three (for example, Ethernet) would oblige communicating services to share a physical network or a physical machine. Figure 2 illustrates the architecture of a single physical machine in the framework.

In order to accommodate multiple instances of the same service and service migration, each service type has a template associated with it that enables the automatic configuration of new service instances and their Xen domains. When replicating a service or creating a new service instance, a new copy of the relevant template is used. Service templates will automatically replicate on other Xenos hosts as required so that service instances can be spawned anywhere on the Xenos cloud. Migration of service instances makes use of Xen's virtual machine migration mechanism with a

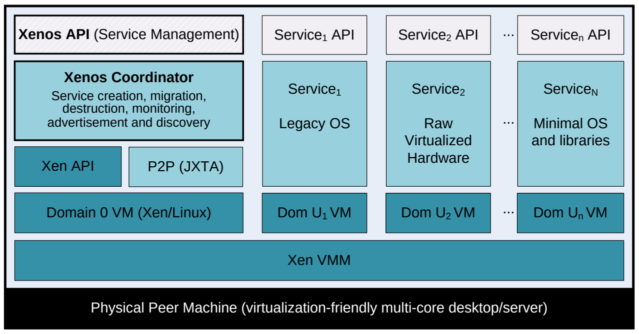


Figure 2. A Xenos-enabled physical machine.

slight modification to transfer virtual machine disk images along with the virtual machine configuration. Our current implementation inherits a Xen restriction limiting live virtual machine migration to the local area network, though this may be overcome as discussed in Section V.

IV. Performance Analysis

A series of preliminary tests were conducted in order to assess the viability of our approach. The test cases all involve deploying multiple instances of a Hadoop map-reduce wrapper service using a separate distributed coordination service. We aim to explore three principal avenues, namely (1) the automatic and dynamic deployment of the Hadoop service to Xenos hosts and the migration of the master Hadoop node from a failing physical machine; (2) the performance of file I/O within virtual machines, which is crucial for services with large-volume data processing requirements (this is particularly relevant since Xenos requires virtual machine images to exist in files rather than as physical disk partitions); and (3) the performance of a series of virtualized Hadoop map-reduce processing jobs.

A similar evaluation of running the Hadoop map-reduce framework within a virtualized cluster is carried out by Ibrahim *et al.* [14]. They argue that a virtual machine-based Hadoop cluster can offer compensating benefits that overshadow the potential performance hit, such as improved resource utilization, reliability, ease of management and deployment, and the ability to customize the guest operating systems that host Hadoop to increase performance without disrupting the cluster's configuration.

A. Map-Reduce and Hadoop

In our experiments we used the HDFS (Hadoop Distributed File System) and MapReduce components of the Apache Hadoop framework. The map-reduce programming model, introduced by Dean *et al.* [15], is aimed at processing large amounts of data in a distributed fashion on clusters. HDFS is a distributed file system suitable for storing large data sets for applications with heavy data processing, such as typical map-reduce jobs. The Hadoop map-reduce implementation involves a master node that runs a single *JobTracker*, which accepts jobs submitted by the user, schedules the job across worker nodes by assigning map or reduce tasks to them, monitors these tasks and re-executes failed ones. Each worker node runs a single *TaskTracker* which is responsible for executing the tasks assigned to it by the job tracker on the master node.

B. Deploying Hadoop Services

Each Hadoop map-reduce node needs to be configured with specific settings, such as the host name, host certificates and HDFS and map-reduce settings that are common throughout the cluster. Setting up a non-virtualized environment usually involves manually configuring a single node, then cloning the hard disk to the rest of the cluster, either manually or via shell scripts and *rsync*.

Our approach is to encapsulate a pre-configured Hadoop installation inside a virtual machine and automatically distribute it across the Xenos network as a service. To facilitate the distribution, we developed a Java/JXTA application to connect to the Xenos cloud, and used the Xenos programming interface to deploy a Hadoop slave worker service. One of the hosts on the cluster, which we refer to as the master host, is configured with a template of the Hadoop slave service as well as an instance of the Hadoop master service, from where we issue commands to deploy services and execute Hadoop jobs.

C. Evaluation Platform and Results

The evaluation platform was a thirteen-host cluster, connected over a 1GB/s Ethernet connection through a D-Link DGS-1224T switch. Each physical machine in the cluster has an Intel Core 2 Duo E7200 CPU, with 3MB of L2 cache clocked at 2.53GHz, 2GB of DDR2 RAM, and a 500GB SATA2 hard disk. In all of our tests, the virtual machine that we use as the Hadoop slave template which is replicated is configured with a 10GB disk image, a 1GB swap image, the *vmlinuz-2.6.24-27-xen* kernel, one VCPU (virtual CPU), 384MB of RAM and a DHCP client. Domain 0 is set to use 512MB of memory, leaving the rest to be allocated to service-hosting virtual machines, and has no restrictions on the number of physical CPUs it can use. One of the cluster hosts is dedicated to hosting the Hadoop slave template and the master service instance, and configured so that no slave services are replicated on it. No optimizations to Hadoop or any other software component were made to suit this particular cluster. In all results, PHY-Cluster refers to a Hadoop cluster on native Linux, while VM1-Cluster, VM2-Cluster and VM4-Cluster refer to Xenos-enabled virtualized clusters with one, two and four virtualized Hadoop slave services deployed per physical host respectively.

1) Replication and Migration of Hadoop Service Templates and Services: The unoptimized replication process took around 45 minutes to deploy a template and a single slave service instance to each of the twelve remaining cluster hosts, which included a network transfer of 132GB as well

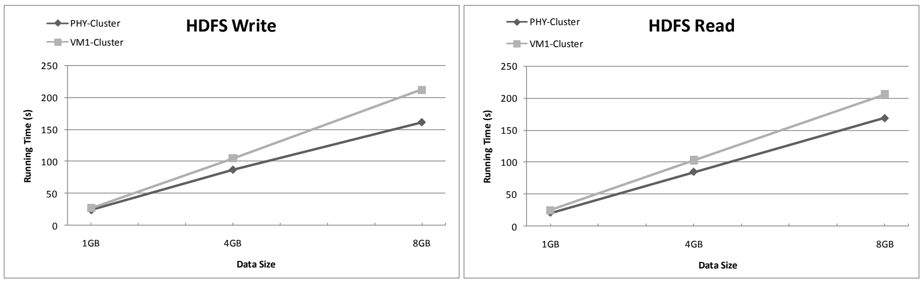


Figure 3. PHY-Cluster *vs* VM1-Cluster with varying data sizes.

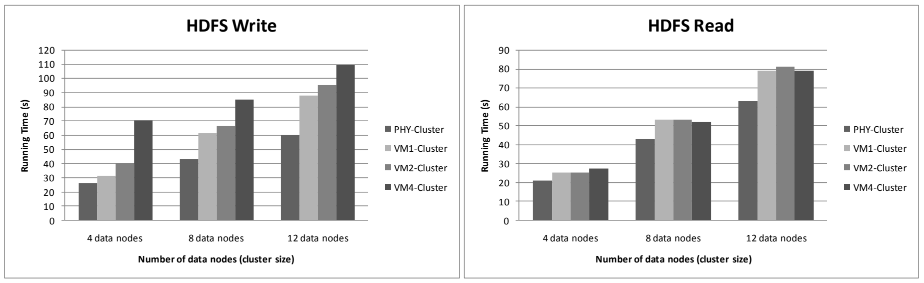


Figure 4. PHY-Cluster *vs* VM clusters with varying data nodes (cluster size) and virtual machines per physical machine.

as another 132GB in local data copying; this translates to a network throughput of around 40MB/s and a disk throughput of around 25MBs/s. Since the process mostly involves transferring domain files over the network or copying them locally, its performance depends on the hardware platform that the services are being deployed on, as well as the size of the domains that contain the service. Once the required templates have been automatically deployed and replicated throughout the cluster, activating existing services takes a tiny fraction of this time. Additionally, the Hadoop master service was successfully migrated between hosts to simulate a physical host that needs to be dynamically repurposed.

2) HDFS Performance: As shown in Figure 3, reading and writing operations on virtualized HDFS suffered a performance drop when compared to a non-virtualized cluster configuration. For small data transfers and clusters, the gap is negligible, but increases with larger data sets. However, as shown in Figure 4, increasing the number of virtualized services per physical host did not cause the read performance of HDFS to deteriorate. Ibrahim *et al.* also make this observation in one of their tests, indicating that the write performance gap increased markedly but it increased only slightly when reading.

3) Hadoop Benchmarks: Figure 5 indicates that increasing the number of computation nodes by adding more virtualized service instances on each physical machine benefits certain processor-intensive Hadoop jobs. In this case, the PiEstimator benchmark performed significantly better when more computing nodes were available. However, jobs that are I/O-intensive and that deal with large data sets suffered a performance hit due to degraded HDFS performance; this was evident in the Wordcount and Sort benchmarks

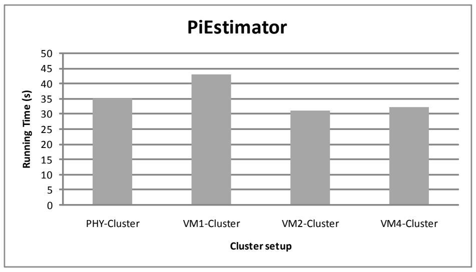


Figure 5. PiEstimator execution on PHY-Cluster and VM clusters with varying virtual machines per physical machine.

as illustrated in Figure 6 and Figure 7. In the Wordcount benchmark, Ibrahim *et al.* fared better on their virtualized clusters with 2 and 4 VMs per physical host than their physical cluster; however each host in their evaluation was equipped with 8 cores, so their CPU core to VCPU ratio was always 1 or greater. Our Sort benchmark results are similar to Ibrahim *et al.*'s: we also observed that once the reducer tasks start executing, the entire job slows down considerably.

As discussed in Section I, we expect future improvements in virtualization technology to further minimize the gap between native and virtualized I/O performance, thus strengthening the case for deploying Xenos and other such platforms.

V. Topics For Further Investigation

With some effort, Xenos can fill the role of a test-bed to facilitate experimentation with a variety of emerging issues in distributed virtualized services, some of which are briefly discussed here.

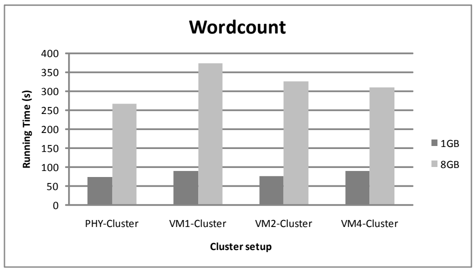


Figure 6. Wordcount execution on PHY-Cluster and VM clusters with varying data input size and virtual machines per physical machine.

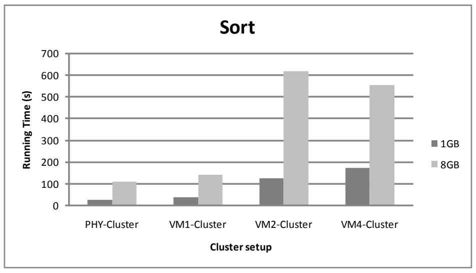


Figure 7. Sort execution on PHY-Cluster and VM clusters with varying data input size and virtual machines per physical machine.

A. A Library of Essential Services

The core functionality provided by the Xenos framework can be further extended and abstracted away through additional services. Examples include service wrappers for load-balancing and fault-tolerance (virtual machine checkpointing is invisible to the service(s) hosted within), virtual machine pooling and replication, service deployers such as the Hadoop deployer discussed previously, platform emulators, legacy services supporting a range of operating systems, and a Xenos-UDDI adapter that can be used to search for Xenos services via UDDI (Universal Description Discovery and Integration). Xenos does not impose a single method for actual service delivery, thus web services, Sun RPC, and even services using raw Ethernet may be advertised.

B. Seamless Wide-Area Service Migration

The issue of live virtual machine migration over WANs has been addressed by several authors and a number of prototypes are available. Travostino *et al.* [16] approach the problem of preserving TCP connections by creating dynamic IP tunnels and assigning a fixed IP address to each virtual machine, which communicates with clients via a virtual gateway interface that is set up by Xen. After migration, a virtual machine retains its address, and the IP tunnels are configured accordingly to preserve network routes – this is completely transparent to TCP or any other higher level protocol. Bradford *et al.* [17] combine the IP tunneling approach with Dynamic DNS to address the problem of preserving network connections. More importantly, the authors also implement a pre-copy approach for transferring the disk image attached to a virtual machine, using a mechanism similar to that used by Xen when live migrating the state of a virtual machine. This greatly minimizes downtime even if the actual migration takes long owing to poor network performance. Harney *et al.* [18] suggest using the mobility features in the IPv6 protocol to preserve network communication sessions, an approach that is viable in the long-term.

C. Alternative Transport Methods For Service Delivery

Applications featuring fine grained concurrency spanning across virtual and physical machines stand to gain from inter-virtual machine communication path optimizations such as shared memory communication for services residing on the same physical machine, and hypervisor-bypass network communication for distributed services. In both instances, the secure initialization of each communication path would be delegated to Xenos, allowing the data to move directly between the participating virtual machines and virtualization-enabled I/O devices. In some cases, an I/O could be permanently and exclusively bound to a specific service for low-latency dedicated access.

D. Security, Authentication and Service Provisioning

A number of underlying mechanisms could be inherited from the Xen hypervisor and the JXTA peer-to-peer framework or their respective alternatives. To our benefit, JXTA provides several security and authentication features, as discussed by Yeager *et al.* [19]; these include TLS (Transport Layer Security), and support for centralized and distributed certification authorities. Xen provides a basis for automated accounting and billing services that track service consumption as well as physical resource use. However, Xenos should at least provide unified and distributed user, service and hierarchical service group authentication and permission mechanisms, a non-trivial undertaking in itself.

E. The Operating System-Agnostic Operating System

Software architectures in the vein of Xenos could fit the role of a distributed microkernel in a virtualization-embracing operating system that consists of interacting light-weight services hosted within virtual machines, including a multi-core thread scheduler, file systems (a stripped down Linux kernel), and device drivers. Each operating system service would run within its own light-weight Xen domain and expose itself through Xenos services (reminiscent of system calls). Xenos services would also host legacy operating systems and applications, presented to users through an operating system-agnostic window manager hosted in a separate virtual machine. Applications with particular resource requirements or requiring isolation, such as computer games or web browsers, may easily be hosted in their own virtual machines, supported by a minimal application-specific kernel or library or even executing on ‘bare virtualized metal’. Xen, and virtual machine monitors in general, have been described as “microkernels done right” [20], although others have argued that the drawbacks that muted the adoption of microkernels [21] still apply.

VI. Conclusion

This paper briefly investigates an approach to building distributed middleware. The Xenos framework extends well-established solutions for virtualization hypervisors and peer-to-peer overlay networks to deliver the beginnings of a fully decentralized solution for virtualized service hosting, discovery and delivery. We expect forthcoming hardware support for virtualization to further reduce the gap between virtualized and native I/O performance pinpointed in our results, while simplifying and possibly commoditizing hypervisors. This will further consolidate the virtual machine's position as a viable alternative for hosting both computation- and I/O-intensive tasks. The combination of peer-to-peer technology and virtualization merits serious consideration as a basis for resilient distributed services.

References

[1] P. Willmann, S. Rixner, and A. L. Cox, "Protection strategies for direct access to virtualized I/O devices," in *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2008, pp. 15–28.

[2] D. Bailey, "Xenos: A service-oriented peer-to-peer framework for paravirtualized domains," Master's thesis, University of Malta, Submitted 2010.

[3] K. A. Fraser, S. M. Hand, T. L. Harris, I. M. Leslie, and I. A. Pratt, "The XenoServer computing infrastructure," University of Cambridge Computer Laboratory, Tech. Rep., 2003.

[4] D. Spence and T. Harris, "XenoSearch: Distributed resource discovery in the XenoServer open platform," in *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 216.

[5] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "WOW: Self-organizing wide area overlay networks of virtual workstations," in *In Proc. of the 15th International Symposium on High-Performance Distributed Computing (HPDC-15*, 2006, pp. 30–41.

[6] V. March, Y. M. Teo, and X. Wang, "DGRID: a DHT-based resource indexing and discovery scheme for computational grids," in *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007, pp. 41–48.

[7] Q. Xia, R. Yang, W. Wang, and D. Yang, "Fully decentralized DHT based approach to grid service discovery using overlay networks," *Computer and Information Technology, International Conference on*, pp. 1140–1145, 2005.

[8] D. Talia, P. Trunfio, J. Zeng, and M. Hgqvist, "A DHT-based peer-to-peer framework for resource discovery in grids," Institute on System Architecture, CoreGRID - Network of Excellence, Tech. Rep. TR-0048, June 2006.

[9] W. Wadge, "Providing a grid-like experience in a P2P environment," Master's thesis, University of Malta, 2007.

[10] M. Amoretti, F. Zanichelli, and G. Conte, "SP2A: a service-oriented framework for P2P-based grids," in *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*. New York, NY, USA: ACM, 2005, pp. 1–6.

[11] S. Thibault and T. Deegan, "Improving performance by embedding HPC applications in lightweight Xen domains," in *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*. New York, NY, USA: ACM, 2008, pp. 9–15.

[12] M. J. Anderson, M. Moffie, and C. I. Dalton, "Towards trustworthy virtualisation environments: Xen library OS security service infrastructure," Hewlett-Packard Laboratories, Tech. Rep. HPL-2007-69, April 2007. [Online]. Available: http://www.hpl.hp.com/techreports/2007/HPL-2007-69.pdf

[13] E. Van Hensbergen, "P.R.O.S.E.: Partitioned reliable operating system environment," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 2, pp. 12–15, 2006.

[14] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating MapReduce on virtual machines: The Hadoop case," in *CloudCom*, 2009, pp. 519–528.

[15] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[16] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 901–907, 2006.

[17] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*. New York, NY, USA: ACM, 2007, pp. 169–179.

[18] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the internet," in *VTDC '07: Proceedings of the 3rd international workshop on Virtualization technology in distributed computing*. New York, NY, USA: ACM, 2007, pp. 1–7.

[19] W. Yeager and J. Williams, "Secure peer-to-peer networking: The JXTA example," *IT Professional*, vol. 4, pp. 53–57, 2002.

[20] S. Hand, A. Warfield, K. Fraser, E. Kotsovinos, and D. Magenheimer, "Are virtual machine monitors microkernels done right?" in *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2005.

[21] G. Heiser, V. Uhlig, and J. LeVasseur, "Are virtual-machine monitors microkernels done right?" *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 95–99, 2006.

Benefits of an Implementation of H-P2PSIP

Isaias Martinez-Yelmo, Roberto Gonzalez, Carmen Guerrero
Universidad Carlos III de Madrid
Av. Univerisidad 30. 28911 Leganes. Madrid. Spain
Email: {imyelmo,rgonzale,guerrero}@it.uc3m.es

***Abstract*—In this paper, we report on the results of experiments with an implementation of H-P2PSIP, which allows the exchange of information among different DHTs (Distributed Hash Tables) making use of a hierarchical architecture. This paper validates our previous H-P2PSIP proposal in an environment with a real TCP/IP stack close to a real scenario. The results show how the benefits of this real H-P2PSIP implementation in terms of routing performance (number of hops), delay and routing state (number of routing entries) are better than a flat DHT overlay network and how the exchange of information among different DHT overlay networks is feasible.**

***Keywords*-Hierarchical DHT, P2PSIP, Implementation, Evaluation, Performance.**

I. Introduction

The traffic related with Peer-to-Peer overlay networks has increased considerably last years. Thus, the research related with Peer-to-Peer in the Internet and how to measure accurately this kind of traffic [1] is currently a hot topic. Many applications generate Peer-to-Peer traffic in Internet, where the most relevant ones are probably eMule [2] with its KAD network [3], Bittorrent [4], [5] and Skype [6], [7]. Some of them are based on the same Peer-to-Peer network, such as trackerless Bittorrent or KAD network from eMule that make use of Kademlia [8], but they cannot establish communication among them although their purpose is the same.

H-P2PSIP [9], [10] presents an architecture based on two levels of hierarchy, which takes into account the ongoing design on the IETF P2PSIP Working Group. The P2PSIP WG aims to develop a protocol that allows the implementation of any Peer-to-Peer network but it does not consider the exchange of information between different domains based on different DHTs. The main usage expected for P2PSIP is a distributed replacement of SIP [11], [12], although P2PSIP has the flexibility to support other ones [13], [14]. Our proposal consist on a hierarchical DHT architecture that allows the exchange of information between different heterogeneous DHTs deployed in different domains, (see Figure 1). The idea is to support the exchange of information between different domains. In relation with a SIP usage, our hierarchical architecture replaces some of the proxy/registrar SIP functionalities.

This proposal is analysed mathematically and validated through simulation in [9], [10]. However, if we consider the complexity of Peer-to-Peer technologies due to their decentralised architecture, an implementation with a real TCP/IP stack is necessary to have a full validation of this proposal. In this paper, we report on the results obtained from a real im-

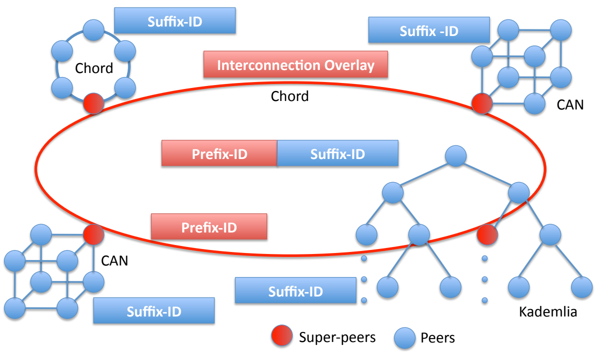


Fig. 1. H-P2PSIP architecture

plementation of our H-P2PSIP proposal. This implementation verifies that the exchange among different DHT overlays is possible without losing performance in terms of hops, delay or routing state. Furthermore, the hierarchical DHT architecture can be used to develop additional features and services with the flexibility offered by H-P2PSIP, since any DHT can be used if desired.

The structure of this paper is as follows. Section II summarizes the related work with respect to the topic of hierarchical Peer-to-Peer networks. A description of H-P2PSIP is given in Section III, detailing the key points of the proposal and helping to understand the results provided in this paper. Section IV explains the main development steps to implement H-P2PSIP. Section V describes the scenario used to validate the implementation. The results obtained from our experiments are detailed in Section VI. Finally, Section VII summarises the conclusions and future work related with this paper.

II. Related Work

One of the first and more relevant publications in hierarchical Peer-to-Peer networks is the work in [15]. This paper describes the benefits of hierarchical Peer-to-Peer networks and presents a mathematical analysis. Subsequently, other publications related with hierarchical Peer-to-Peer networks have been published. For instance, Hieras [16] minimizes in a multilevel hierarchy the delay experienced by queries. Hieras groups each level of the hierarchy according to the delay among peers; the higher is the delay among peers, the higher is the level of the hierarchy where they are grouped. Queries are launched in the lowest level of the hierarchy and if a resource is not found, the next level is used. The main drawback of Hieras is the increase of routing state and maintenance traffic

Prefix-ID (n-bits)	Suffix-ID (m-bits)

Fig. 2. Hierarchical-ID

in all peers, since each peer needs to maintain each level of the hierarchy. Other issues related with hierarchical ring topologies are published in [17]. Additional improvements and algorithms that are more complex can be used if necessary. Cyclone [18] builds a hierarchical DHT overlay network in similar manner with respect to [16]. However, it assures that the number of links maintained by each peer is similar to the flat counterpart. Other approaches (i.e., Canon [19]) optimize the delay, but they limit the Routing State that is maintained by peers. Unfortunately, in most of these proposals, the design of the hierarchical architecture is heavily coupled to the DHT used to build the hierarchy. Our proposal, H-P2PSIP [9], [10], offers a great flexibility at this point since the architecture does not depend on the DHTs being used. In addition, different types of DHT networks can be used if necessary without compromising the performance of the proposed design. We demonstrate these advantages of H-P2PSIP with the implementation presented in this paper.

III. H-P2PSIP

H-P2PSIP [9], [10] defines a hierarchical DHT overlay network composed of two levels of hierarchy; an example is given in Figure 1. The purpose of this two-level hierarchy is the exchange of information among different domains where any DHT overlay network is deployed inside of them. The lower level is composed of different domains that want to exchange information among them. Each domain is independent of the others; therefore, the peers in each domain implement a DHT overlay network according to the domain preferences. Thus, the DHT overlay network can be different in each domain. On the other hand, the upper level is composed of only one DHT overlay network, named Interconnection Overlay. This Interconnection Overlay provides a service similar to a directory service among the different domains from the lower level of the hierarchy. The purpose of this Interconnection Overlay is to route the queries among different domains when a peer of one domain wants to retrieve information stored in other domain. This Interconnection Overlay can be based on any DHT overlay network. In order to support these functionalities, H-P2PSIP uses a hierarchical space of identifiers composed of Hierarchical-IDs (see Figure 2). A Hierarchical-ID contains two concatenated IDs: a Prefix-ID and a Suffix-ID.

The Prefix-ID is used to route queries in the Interconnection Overlay among the different domains. This implies that all the peers or resources belonging to the same domain share the same Prefix-ID. On the other hand, the Suffix-ID is used only in the domain where a peer is attached and permits to localize any resource in the overlay network of that domain. Thus, this design allows the routing of queries among different domains. When searching for a resource in another domain, the query is routed to the desired domain using the Prefix-ID. Finally, the desired resource in the external domain is found through the Suffix-ID.

A. Hierarchical-ID generation

The next paragraphs explain how we generate the Node-IDs and Resource-IDs (based on Hierarchical-IDs) by applying hash functions to a regular URI. The starting point for this discussion is that each domain has a domain name (e.g., `example.com`) and its resources are identified by an URI (e.g., `resource@example.com`).

Node-ID generation. A Node-ID identifies each node participating in the overlay network of a domain. The generation of a Node-ID depends on the security level desired in the system. If we have a domain named `example.com`, the Prefix-ID is generated as follows: `Prefix-ID=hash(example.com)`. This construction of the Prefix-ID allows its generation if the domain that wants to be contacted is known in advance. On the other hand, the Suffix-ID must be different for each peer. Several mechanisms exist to generate this Suffix-ID. One option is to apply a hash function to some parameter (e.g., `Suffix-ID=hash(ip_address)`), but this option is not completely secure. A secured solution implies a central authority that assigns Node-IDs with signed certificates. The central authority uses a random number generator for Suffix-IDs and avoids unnecessary Node-ID collisions tracking the previously generated IDs.

Resource-ID generation. If we have a resource associated to a regular URI such as `resource@example.com`, a Hierarchical-ID can be created using this URI. The Prefix-ID in a Resource-ID must have the same value used in the Node-IDs. In fact, this is possible since the URI contains the domain where the resources are located. Therefore, the Prefix-ID must be: `Prefix-ID=hash(example.com)`. On the other hand, the Suffix-ID must be different for each resource if possible; thus, the Suffix-ID is generated hashing the whole URI: `Suffix-ID=hash_a(resource@example.com)`. The hash functions `hash` and `hash_a` can be identical or different. Once the mapping between the URIs and Hierarchical-IDs is established, any resource can be stored with its Resource-ID and the original URI in order to perform disambiguation in resources with the same Resource-ID. Taking into account this Resource-ID generation, the information associated to a domain is stored in that domain since Resource-IDs are couple to each domain through the Prefix-ID. However, it is possible to store pointers to resources from other domains if necessary.

B. Super-Peer role in H-P2PSIP

Super-peers are necessary to interconnect the different domains and to route the queries among these domains; see Figure 1. We mentioned previously how the routing is based on the defined Hierarchical-ID, and now we explain how to use this Hierarchical-ID. In order not to overload most of the peers with additional tasks, we use super-peers to realize the necessary additional operations. Basically, super-peers forward the queries to the right external domain specified in the query. The super-peers must participate in the Interconnection

Overlay as well as in their own domain as regular peers. Taking into account the role of super-peers, their main tasks are enrolment and maintenance operations in the Interconnection Overlay and forwarding of inter-domain queries. Since super-peers have to manage the Interconnection Overlay, they have to manage an additional overlay routing table in addition to the regular that peers use to perform queries inside their own domain. The forwarding of queries implies a higher load in terms of CPU and bandwidth consumption in super-peers.

An open issue is the management of super-peers. H-P2PSIP suits very well to create a distributed VoIP signalling protocol where the location information is stored in the DHTs of the different domains (different usages can be also applied to H-P2PSIP if desired). Considering VoIP scenarios, two scenarios mainly require attention: operator-based scenarios or community networks based scenarios. In operator-based scenarios, operators want the control of super-peers to get track of whole signalling in order to charge users. Therefore, operators have to provide the necessary super-peers and take care of their full availability since they are the key for charging users. On the other hand, in community networks based scenarios, a mechanism is necessary to select the most suitable super-peers dynamically and perform load balancing among them. There are several published proposals on the selection of super-peers in a proper way [20]–[22]. These mechanisms can be integrated in the maintenance operations of any Peer-to-Peer protocol if necessary.

C. Signalling - URI based routing

The H-P2PSIP signalling is associated to all the operations in the overlay network. However, the most interesting operations are the storage and retrieval of resources. An example of the signalling on the proposed hierarchical scenario is shown in Figure 3 (the figure omits the intermediate hops in each DHT). Several aspects are taken into account in order to understand the signalling flow. The peer in `domain.a` performs the storage operation inside its domain. Later, peer in `domain.b` requests the information of `user1@domain.a`. Here, an important issue is that the query in the Fetch message contains the whole URI as plain text. This method allows rebuilding the Hierarchical-ID independently of the hash function used in the different domains. Thus, once the super-peer in `domain.b` receives the query, it performs a query of `hash(domain.a)` in order to obtain the information related with the super-peers in `domain.a` through the Interconnection Overlay. Inside this query, the hash used in the other domain ($hash_a$) is included and a request for the desired item is built as `hash_a(user1@domain.a)` and sent to the super-peer in `domain.a`. The super-peer from domain `domain.a` forwards the information to its overlay and waits for the answer. Once the answer is received, the super-peer from `domain.a` forwards the resource information to the super-peer from `domain.b`. Finally, the super-peer from `domain.b` sends the response to the peer that generated the original query. This response path is the same that the request path in order to be compliant with the ongoing design in the IETF P2PSIP WG.

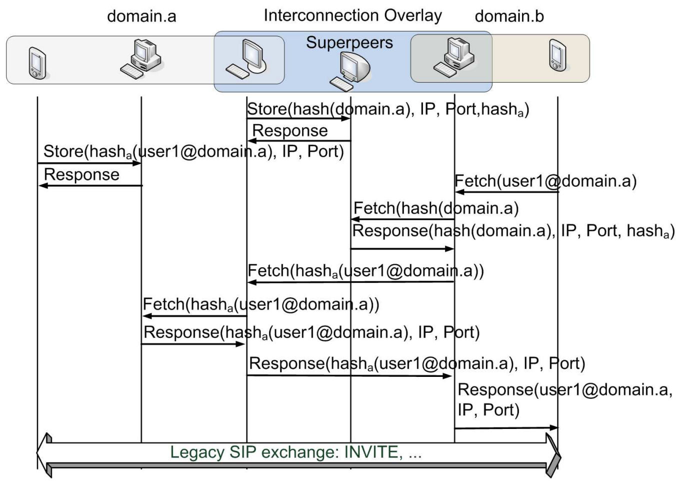

domain.a
Interconnection Overlay
domain.b
Superpeers
Store(hash(domain.a), IP, Port,hash$_a$)
Response
Store(hash$_a$(user1@domain.a), IP, Port)
Response
Fetch(user1@domain.a)
Fetch(hash(domain.a)
Response(hash(domain.a), IP, Port, hash$_a$)
Fetch(hash$_a$(user1@domain.a))
Fetch(hash$_a$(user1@domain.a))
Response(hash$_a$(user1@domain.a), IP, Port)
Response(hash$_a$(user1@domain.a), IP, Port)
Response(user1@domain.a, IP, Port)
Legacy SIP exchange: INVITE, ...

Fig. 3. H-P2PSIP Signalling

Additionally, the example shows a possible SIP usage of H-P2PSIP, since H-P2PSIP is used to locate the end-points of the communication, replacing the SIP proxy/registrar functionalities. Later, a direct legacy SIP point-to-point negotiation is established to setup the parameters for a VoIP call. Finally, we highlight that the signalling can be transmitted over TLS [23] /DTLS [24] to increase the security and to avoid URI tracking from adversaries.

D. Characteristics of H-P2PSIP

H-P2PSIP proposal has several advantages. In order to see these advantages, we compare it with a flat overlay network counterpart. First, the operations or primitives of the DHTs used in H-P2PSIP are not modified. The routing state in legacy peers does not increase with respect to a flat overlay network because the number of maintained peers per overlay network is the same. It is not necessary to store information of peers from other domains; peers only need to store their super-peer to reach them. Hence, the number of the peers in each domain limits the number of routing entries, although connectivity with other P2PSIP domains is possible, which was a prerequisite in the design rules. If we consider that the Routing State in a Peer-to-Peer network usually depends on the logarithm of the number of peers, we have the fact that the Routing State in our approach is $O(\log_B M)$ where M is the number of peers in a domain (B is the base of the ID space). If we compare this Routing State with a single flat P2PSIP domain that contains all P2PSIP domains, we obtain that the number of peers in the flat overlay network is $M \cdot K$, where K is the number of domains and the Routing State is increased up to $O(\log_B(M \cdot K))$. Thus, using the hierarchical architecture, legacy peers save $log_B(K)/log_B(M{\cdot}K){\cdot}100\%$ of overlay routing entries in comparison with the flat counterpart. In addition, the Routing State is independent with respect the number of P2PSIP domains, which is also interesting.

The drawback of this approach is the overload of super-peers [25]. This fact implies the maintenance of a larger amount of information with respect to the peers. Actually, our super-peers have to maintain two routing tables: a routing table of size $O(\log_B M)$ for their own domain and a routing table of size $O(\log_B K)$ for the Interconnection Overlay. In this case, the Routing State in super-peers is $O(\log_B M)+O(\log_B K) = O(\log_B(M \cdot K))$, this values corresponds with an equivalent flat overlay network with all peers of the different domains. Therefore, the only drawback with respect to a completely flat overlay network is the bandwidth and CPU consumption associated to the forwarding and processing of queries that belong to other domains.

IV. Implementation

Our implementation is based on the Peer-to-Peer Protocol (P2PP) [26], [27], from the University of Columbia. We choose this protocol because its implementation is available and it is being used as reference with other protocols in the IETF P2PSIP WG to design the ongoing P2PSIP protocol named RELOAD [28]. The full details of our implementation are too complex to be detailed here, but we highlight the key points that are absolutely necessary to provide to P2PP the functionalities of H-P2PSIP. The original design only supports legacy flat overlays. Therefore, it is necessary to add support to Hierarchical-IDs. In addition, it is also necessary to manage the new Hierarchical-IDs. Peers have to check if their queries target their own domain (same Prefix-ID). If the Prefix-ID is different, the query must be forwarded to a super-peer. We implement super-peer role by developing peers that are attached to the overlay network from their domain and the Interconnection Overlay. Super-peers forward the queries to the super-peer that belongs to the destination domain according to the defined signalling (Figure 3). One detail that is not implemented, in order to avoid more complexity in the development of our proposal, is the hashing method to map URIs to Hierarchical-IDs since this mapping is straightforward and it does not affect to the performance of our solution.

V. Scenario Setup

In order to test and develop our implementation, we have a configurable scenario based on network emulator software. The selected software is Modelnet [29] since offers an excellent infrastructure for the debugging and testing of distributed applications. Its emulation properties allow the repeatability of experiments, which helps in the debug process. In addition, it also allows emulating big core networks with different properties (delay, probability of error, etc); this feature allows evaluating the program in conditions closer to current Internet with a real TCP/IP stack.

A. Modelnet setup

Our implementation is tested in a Modelnet scenario running 1000 peers, which are grouped in a number of different domains ranging from 1 to 20 (all of them with the same number of peers). Scenarios with fewer peers are not considered since Peer-to-Peer networks are supposed to have at least thousands of peers. More peers are not considered due to the limitation of our computational resources. Each peer is deployed in an OpenVZ [30] virtual machine. The core network that interconnects the different peers is emulated according to the tools offered by Modelnet. Modelnet, depending on the computational capacity, can emulate any topology with different link properties. In order to have a topology as much realistic as possible, the measurements provided by the IEPM PingER project [31] are used to define the topology used in the experiments. Different developed representative countries are selected to build the core network of our experiment. With the information provided by the PingER project, the links among the different countries and their characterization (bandwidth, delay, losses, etc) are introduced in the Modelnet core emulator in order to have a reasonable representation of the current Internet. This setup allows the usage of a TCP/IP stack in an environment closer to the real world.

B. Peers setup

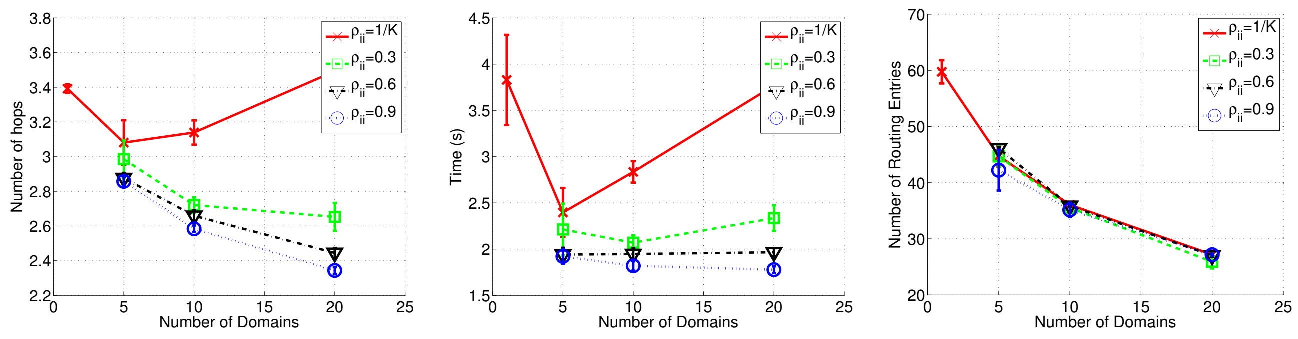
In our experiments, it is also necessary to reflect the peer behaviour. The session time distributions, and churn rate of peers are configured considering the measurements in [3]. However, we do not take the values in [3] directly but we adapt them to our population of 1000 nodes. Therefore, according to the previously mentioned paper, the churn rate is based on a negative binomial distribution. However, the average of the churn rate is scaled to the number of peers in our experiment. If we would not perform this operation, the churn rate would be very high for the number of peers in our experiments since the measurements in [3] are composed by a higher number of peers. For the query generation rate, a Poisson distribution is used since one of the most suitable scenarios for H-P2PSIP is VoIP and telephone calls, which are usually Poisson distributed. The protocol used in all the domains as well as in the Interconnection Overlay is Kademlia. This selection allows the comparison with the equivalent flat Kademlia counterpart of an overlay network with all peers in all domains. The Kademlia [8] implementation found in the P2PP protocol [27] has the following parameters, which are the same with respect to the original implementation: $B = 2$, implying that Kademlia performs routing based on a binary tree. The size of the buckets used to store the information of the routing entries for each level of the binary tree is $k = 20$. Kademlia can perform parallel queries to several peers when a query is being requested, the P2PP implementation uses $\alpha = 1$, thus the parallelization mechanism is not used. One super-peer per domain with high availability (churn does not affect super-peers) is used according to an operator-based scenario (operators take care of the high availability) of H-P2PSIP, which has been described previously.

VI. Results

This section presents the results obtained in scenario described previously; but, first of all, it is necessary to explain how experiments are conducted.

A. Experiments setup

The experiment is divided in two phases. First, we have a transitory phase limited to 30 minutes, during this time the



(a) Average number of hops (b) Average delay (c) Routing State in peers

Fig. 4. Results with respect to the number domains

average number of peers needed to realise the experiment join the hierarchical DHT network. The second phase, it is the stationary state and it has duration of 50 minutes. In this phase, the negative binomial distribution for join and departure rates of peers is used as well as the Poisson distribution for the query generation rate of peers. During this phase, the data are collected to obtain the results that are shown in the next section. The experiments are repeated several times (at least 10 times) in order to obtain results with representative 95% confidence intervals and an error smaller than 10% with respect to the estimated average of the results.

B. Routing Performance

Figure 4(a) shows the number of hops made to reach the destination with respect to the number of domains used to group the peers; this figure also shows this number of hops for different values of the intra-domain hit probability (ρ_{ii}). ρ_{ii} is the probability that a peer has of performing a query inside its own domain instead of any other external domain, this formulation comes from [9], [10]. The first important point is the fact that the expected number of hops is close and slightly better with respect to the simulations in [9], [10]. We explain this behaviour later. In addition, when the queries are randomly made among the different domains (ρ_{ii}=1/K), the red continuous line with cross markers), we observe how the number of hops starts to be smaller in comparison with the flat counterpart (first point of the plot that corresponds to K=1). Furthermore, as the number of domains increases, the number of hops increases slightly but very close with respect to the flat counterpart routing performance. This increment is produced since ρ_{ii}=1/K and the probability of looking in other domain increases, which implies an extra number of hops (one hop to reach the super-peer in addition to the number of necessary hops through the Interconnection Overlay). On the other hand, while the intra-domain hit probability increases (ρ_{ii}>1/K), the number of hops decreases since extra hops to reach other domains are unnecessary.

Figure 5(a) shows the number of hops made to reach the destination with respect to the intra-domain hit probability. This plot gives a view of the effects of the intra-domain hit probability. If the intra-hit domain probability increases, more queries are performed in the own domain of each peer and the average number of hops is smaller. Furthermore, the plot gives a clear comparison of the Routing Performance with respect to the flat counterpart, represented with a dotted blue line with round markers, that is clearly different. We see how the number of hops is smaller than the equivalent flat DHT.

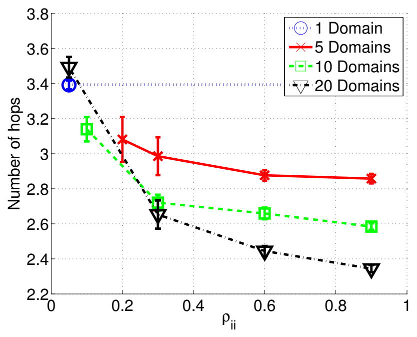
We have also figures that show the average delay suffered by the queries in our scenario. Figure 4(b) represents the delay with respect to the number of domains and the figure includes the legend for different values of ρ_{ii}. We see that a certain correlation exists among the results in Figure 4(b) and Figure 4(a). When ρ_{ii}=1/K, the delay increases if the number of domains increases. The extra number of hops needed to reach the information in other domains causes this effect (Figure 4(a)). On the other hand, if ρ_{ii} increases, the average delay is reduced considerably since the number of queries to other domains is reduced and fewer hops are needed. Finally, Figure 5(b) shows how the delay depends more on ρ_{ii} rather than the number of domains.

C. Routing State

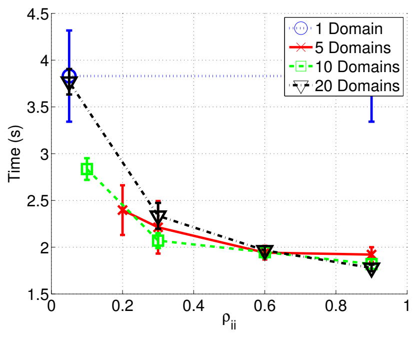
In previous section, the Routing Performance and delay are analysed; nevertheless, it is also interesting to study the load sustained by peers to support the structure of the DHT overlay network in order to route the queries correctly. In order to do that, we collect the average number of routing entries of the peers in order to estimate the information that must be maintained by them. These routing entries not only consume memory on peers, they also imply a bandwidth consumption to maintain them updated. Therefore, the number of entries in peers also reflects partially the effort undergone by peers to assure the validity of their overlay routing information.

Figure 4(c) shows how the number of routing entries needed decreases if the number of domains increases. This effect is expected due to the limitations of our testbed, we must maintain a constant number of peers in our experiments. Thus, if the number of domains increases, the number of peers per domain decreases and consequently the number of routing entries per peer decreases.

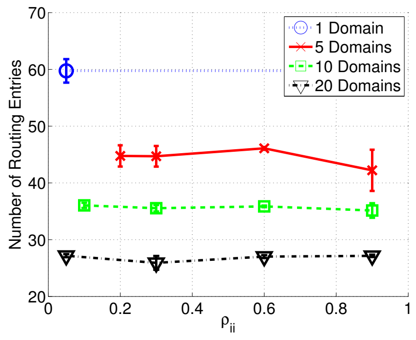
On the other hand, Figure 5(c) demonstrates that the intra-domain hit probability (ρ_{ii}) has a negligible effect on the



(a) Average number of hops

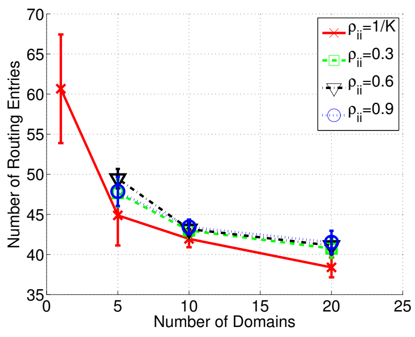


(b) Average delay

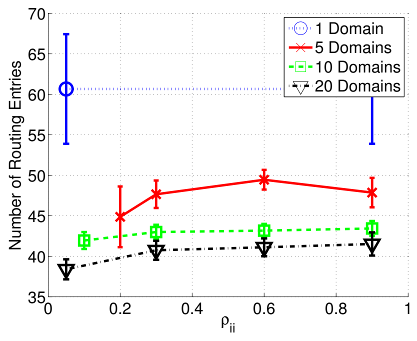


(c) Routing State in pees

Fig. 5. Results with respect ρ_{ii}



(a) With respect to the number of domains



(b) With respect to ρ_{ii}

Fig. 6. Routing State in super-peers

number of entries needed by every peer. The overlay routing tables must be maintained with independence of this parameter and they depend on the number of peers in the overlay network. If the plot is examined carefully, a slight decrease of the number of entries is observed for 5 domains and ρ_{ii}=0.9. However, it is appreciated in the figure how the confidence interval is bigger for this value (the error here is around 10%) and how the upper bound of the confidence interval is close to the other values and cannot be considered an atypical value. However, we plan to study this effect further in the future.

The number of routing entries in both figures is larger than the values obtained in the simulations. Therefore, the Kademlia implementation in P2PP stores in average a higher number of routing entries rather than the Kademlia protocol in the simulation. This fact explains the previously observed better routing performance of the emulated experiment with respect to the simulated experiment. More populated routing tables in peers allow finding the destination in fewer hops. This increment in the average number of routing entries is motivated by some differences in the procedure to update the overlay routing tables, which can be expected from different implementations. Nevertheless, the obtained results are considered valid since the number of routing entries is among the theoretically expected values [8].

In addition to these results, Figure 6(a) illustrates the Routing State information of super-peers. The super-peers must maintain the overlay routing table associated to the Interconnection Overlay, in addition to the own overlay. In fact, Figure 6(a) shows the average number of routing entries taking into account both routing tables. The size of the overlay routing table that belongs to the own domain is quite similar to the given in Figure 4(c). The extra number of entries corresponds to the overlay routing table of the Interconnection Overlay. The difference is not very large since the number of interconnected domains is small due to the limitations of our testbed. In Figure 6(a), we see how the number or routing entries decreases if the number of domains increases and how the intra-domain hit probability affects negligibly to these values in Figure 6(b). These results also give an overview about the requirements in peers to support our proposal.

VII. Conclusions and Future Work

This paper presents the results obtained from our implementation of H-P2PSIP based on the P2PP [27] protocol. The evaluation of this implementation is based on an experiment with 1000 nodes running on a Modelnet emulation framework using the information from PingER project.

In detail, if the queries are randomly distributed among the different domains, the required number of hops and delay increases if the number of domains increases, but they are always below the flat overlay counterpart. This fact is really interesting since the size of routing tables to achieve this performance is smaller (both peers and super-peers) in comparison with the equivalent flat overlay network. This fact is a good advantage over the traditional flat overlay networks. However, the main advantage of our design is the fact that any DHT overlay network can be used in the H-P2PSIP architecture. Therefore, each domain chooses the overlay network that suits better its requisites, which offers a great flexibility for any deployment.

Our implementation demonstrates that the interoperability among different overlay networks is possible, although the price of this feature is the dependency on super-peers.

Our future work plan consists on performing measurements with our testbed to quantify the exchanged traffic depending on the number of domains to evaluate accurately the maintenance scalability of our solution with respect to the bandwidth perspective. In addition, we also plan to perform measurements with different number of peers per domain in order to be closer to a real operation environment. We also plan to study super-peer selection mechanisms for DHTs networks to make feasible our solution in community network scenarios.

VIII. Code Availability

The code of H-P2PSIP developed until this moment is available at http://hdl.handle.net/10016/8382

Acknowledgments

We thank A. Bikfalvi, M. Gramaglia, I. Soto and anonymous reviewers for their insightful comments to improve this paper.

References

[1] W. John, S. Tafvelin, and T. Olovsson, "Trends and differences in connection-behavior within classes of internet backbone traffic," *Passive and Active Network Measurement*, pp. 192–201, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79232-1_20

[2] Y. Kulbak, D. Bickson, A. Prof, and S. Kirkpatrick, "The emule protocol specification," DANSS (Distributed Algorithms, Networking and Secure Systems) Lab School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Tech. Rep., 2005.

[3] M. Steiner, T. En-Najjary, and E. W. Biersack, "A global view of kad," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 117–122.

[4] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, *Peer-to-Peer Systems IV, LNCS*. Springer, 2005, vol. 3640, ch. The Bittorrent P2P File-Sharing System: Measurements and Analysis, pp. 205–216, 10.1007/11558989_19. [Online]. Available: http://dx.doi.org/10.1007/11558989_19

[5] D. Erman, D. Ilie, and A. Popescu, "Bittorrent session characteristics and models," *River Publishers Series in Information Science and Technology. Special Issue in Traffic Engineering, Performance Evaluation Studies and Tools for Heterogeneous Networks*, vol. 1, pp. 61–84, 2009.

[6] S. A. Baset and H. G. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–11.

[7] D. Rossi, M. Melia, and M. Meo, "A detailed measurement of skype network traffic," in *In IPTPS 2008*, 2008.

[8] P. Maymounkov and D. Mazieres, *Peer-to-Peer Systems: First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002. Revised Papers*, ser. Lecture Notes in Computer Science. Springer, 2002, vol. 2429/2002, ch. Kademlia: A peer-to-peer information system based on the XOR metric, pp. 53–65. [Online]. Available: http://www.springerlink.com/content/2ekx2a76ptwd24qt/?p=d9a88bd0609d4ac3902f147d1c183345&pi=0

[9] I. Martinez-Yelmo, A. Bikfalvi, R. Cuevas, C. Guerrero, and J. Garcia, "H-p2psip: Interconnection of p2psip domains for global multimedia services based on a hierarchical dht overlay network," *Computer Networks (Special Issue on Content Distribution Infrastructures for Community Networks)*, vol. 53, no. 4, Mar. 2009. [Online]. Available: http://hdl.handle.net/10016/3390

[10] I. Martinez-Yelmo, A. Bikfalvi, C. Guerrero, R. Cuevas, and A. Mauthe, "Enabling global multimedia distributed services based on hierarchical dht overlay networks," *International Journal of Internet Protocol Technology (Special Issue on Future Multimedia Networking)*, vol. 3, no. 4, Dec. 2008. [Online]. Available: http://hdl.handle.net/10016/4300

[11] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "A sip usage for reload," March 2010, internet Draft draft-ietf-p2psip-sip-04.txt.

[12] Y. Gao and Y. Meng, "A new sip usage for reload," July 2009, internet Draft draft-gaoyang-p2psip-new-sip-usage-00.txt.

[13] S.-K. Ko, Y.-H. Kim, S.-H. Oh, B.-T. Lee, and V. P. Avila, "Iptv usage for reload," July 2009, internet Draft draft-softgear-p2psip-iptv-02.txt.

[14] J. Wang, J. Shen, and Y. Meng, "Content sharing usage for reload," June 2009, internet Draft draft-shen-p2psip-content-sharing-00.txt.

[15] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller, "Hierarchical p2p systems," in *Euro-Par 2003 Parallel Processing*, Springer, Ed., vol. 2790, 2004, pp. 1230–1239.

[16] Z. Xu, R. Min, and Y. Hu, "Hieras: a dht based hierarchical p2p routing algorithm," in *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, 2003, pp. 187–194.

[17] A. Mislove and P. Druschel, "Providing administrative control and autonomy in structured peer-to-peer overlays," *Peer-to-Peer Systems III*, pp. 162–172, 2005. [Online]. Available: http://www.springerlink.com/content/XR2X7MXUFB2FRYEX

[18] M. Artigas, P. Lopez, J. Ahullo, and A. Skarmeta, "Cyclone: a novel design schema for hierarchical dhts," in *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on*, Aug.-2 Sept. 2005, pp. 49–56.

[19] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in g major: Designing dhts with hierarchical structure," *icdcs*, vol. 00, pp. 263–272, 2004.

[20] S.-H. Min, J. Holliday, and D.-S. Cho, "Optimal super-peer selection for large-scale p2p system," in *Hybrid Information Technology, 2006. ICHIT'06. Vol 2. International Conference on*, vol. 2, 2006, pp. 588–593.

[21] A. T. Mizrak, Y. Cheng, V. Kumar, and S. Savage, "Structured superpeers: leveraging heterogeneity to provide constant-time lookup," in *Internet Applications. WIAPP 2003. Proceedings.*, 2003, pp. 104–111.

[22] M. Sanchez-Artigas, P. Garcia-Lopez, and A. F. G. Skarmeta, "On the feasibility of dynamic superpeer ratio maintenance," in *P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 333–342.

[23] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFC 5746. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[24] T. Phelan, "Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)," RFC 5238 (Proposed Standard), Internet Engineering Task Force, May 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5238.txt

[25] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Data Engineering, 2003. Proceedings. 19th International Conference on*, 2003, pp. 49–60.

[26] "P2PP webpage," last access date: 2010-07-13. [Online]. Available: http://www1.cs.columbia.edu/~salman/peer/

[27] S. Baset, H. Schulzrinne, and M. Matuszewski, "Peer-to-peer protocol (p2pp)," November 2007, internet Draft draft-baset-p2psip-p2pp-01.txt.

[28] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "Resource location and discovery (reload) base protocol," July 2010, internet Draft draft-ietf-p2psip-base-09.txt.

[29] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 271–284, 2002.

[30] "OpenVZ webpage," last access date: 2010-07-13. [Online]. Available: http://wiki.openvz.org

[31] W. Matthews and L. Cottrell, "The pinger project: active internet performance monitoring for the henp community," *Communications Magazine, IEEE*, vol. 38, no. 5, pp. 130 –136, may 2000.

Experiences with a P2P Infrastructure for Massive Simulations

Cristoforo Caponigri, Gennaro Cordasco, Rosario De Chiara and Vittorio Scarano

ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli",
Università degli Studi di Salerno, Fisciano, Italy.
Email: {cordasco, dechiara, vitsca}@dia.unisa.it

***Abstract*—Massive Multiuser Virtual Environments (MMVEs) are rapidly expanding both in the number of users and complexity of interactions. Their needs of computational resources offer new challenges for the computer scientists. In this paper we present an implementation and some early tests of a Massive Simulation Environments, a particular MMVE, distributed over a Peer-to-Peer infrastructure. We provide some analysis of the problems related to the workload distribution in this environment. Simulation tests show a good grade of scalability and the communication overhead, due to the peers interaction, is dominated by the computational power provided by them.**

***Keywords*-Massive Simulation, Peer–to–Peer, Load Balancing.**

I. Introduction

The recent growth and popularity of online Massively Multiuser Virtual Environment (MMVE) have raised a lot of interests for the development and research of novel platforms for next-generation MMVEs. Examples of this trend are *World of Warcraft* and *Second Life* which have reached around 10 million subscribers worldwide and roughly 1 million of active users [1].

The design and management of MMVE, due to their highly interactive nature, poses many unique challenges compared to traditional network domains [2]. A single server, or even a small number of servers, is not able to handle the load generated by such systems. So, even if the client/server approach is quite common for small-size MMVEs, it is mandatory for next-generation MMVEs to use the computing power of a group of many servers with dedicated responsibility.

Distributed Virtual Environment (DVE) is an emerging research field which combines 3D graphics, networking and behavioral animation with the purpose of simulating realistic and immersive virtual environments offering a high degree of interactivity. The distributed nature of these systems widened the scenarios of use that now ranges from online videogames to serious games for training including online cooperative systems for learning and problem solving.

Acknowledging the fact that client/server paradigm does not fit well the MMVE scenario, one of the main issues that should be considered designing a DVE system is how to split the responsibilities between the servers/workers. Several approaches have been proposed. For instance, every server can have a different assignment (communication, artificial intelligence, physics, game state) or, on the other hand, a single server acts as a *factotum* server for a portion of the whole environment (a.k.a. *shard*). In this case the actors which belong to different shards cannot interact with each other (each shard represents a distinct copy of the whole environment). Another approach to achieve scalability, named Geographic decomposition, exploits the locality of the environment, decomposing the map on which the game is played into different regions, each of which is associated to a worker. Geographic decomposition suits particularly well to support a completely distributed MMVE, built on top of Peer-to-Peer (P2P) infrastructure, where the responsibility of maintaining the whole environment is shared among all its users. In this approach the workload balancing is essential for both the overall performance and scalability. In the context of DVEs we are particularly interested in Distributed Massive Simulation Environments (DMSEs), which, for the same reasons, appear as a suitable problem that can greatly benefit from the use of a P2P approach.

A. Massive Simulation Environments (MSEs)

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since the 1980s with the purpose of simulating a group of entities, dubbed *autonomous actors*, whose movements are related to social interactions among group members.

A classical example of use of this approach is the simulation of a flock of birds in the most natural possible way. Elements of this simulated flock are usually named *boids* (from *bird-oid*) and got instilled a range of *behaviors* that induces some kind of *personality*. A widespread approach to this kind of simulations has been introduced in [3]. Every boid has its own *personality* (e.g. the trajectory of its flight) that is the result of a weighted sum of a number of *behaviors*. The simulation is performed in successive steps: at each step, for each boid and for each behavior in the personality, the system calculates a request to accelerate in a certain direction in the space, and sums up all of these requests; then the boid is moved along this result. The behaviors are, in the most of cases, simply geometric calculations that are carried out for each boid considering the k-neighbors it is flying with: for example the behavior called *pursuit* just let the boid to pursuit a moving target (e.g. another boid). Each boid reacts to its k-neighbors, which constitute its neighborhood. Given a certain boid out of a flock of n boids, the most simple way of identifying that boid's neighborhood is by an $O(n^2)$ proximity screening, and for this reason the efficiency of the implementation is still to be considered an issue.

B. Designing a P2PMSE

For the purposes of this paper we refer to MSEs as a family of complex interactive 3D environments whose main functionalities can be divided in Simulation and Rendering. We intend to expand the number of simulated actors in a MSE by distributing the computational load to various PCs connected to the system. In a proper DMSE both Simulation and Rendering can be accomplished by a distributed network of computer. Each of the workers offers computational power and in exchange yields the right to visualize the simulation.

Each user connected to the system will have two different subsystems running on his PC: a simulation engine and a visualization engine. The simulation engine will take the responsibility of simulating a small part of the actors in the system, while the visualization engine will let the user to choose which part of the map to visualize. The system architecture is based on a *Distributed Hash Table* (DHT) that will let the user to dynamically connect to the system in a totally distributed manner: once a new peer is available in the system, it will receive part of the simulation and will receive the updates from the system.

The visualization engine will let the user to freely place a camera in the environment and this camera will define an Area of Interest (AOI). AOI is a fundamental concept, as even though many actors and events may exist in the simulated environment, the user, as in the real world, is only interested by nearby actors or events. AOI thus specifies a scope for information which the system should provide to the user. Notice that each subsystem, simulation and visualization, will have its own AOI: the AOI for the simulation, henceforth neighborhood, is defined by the position of actors the peer is simulating, the AOI of the visualization is defined by where the user placed the camera.

C. Our result

We present our experiences with the design and implementation of a fully distributed Massive Simulation Environment. We report and analyze several experimental results we have obtained with our system. Simulation tests show a good grade of scalability and the communication overhead, due to the peers interaction, is dominated by the computational power provided by them.

D. Paper structure

In section II we describe the system architecture and the details of the implementation, while in section III we report the tests setting, the test results and some discussion. In section IV we discuss some possible future works.

II. An architecture for a DMSE

The core of this paper is to describe our experiences in designing, implementing and assessing performances of a DMSE built on top of a P2P system. The motivation for such architecture lays in the usage we envision: users will connect to such system in order to simulate more and more complex scenarios, and the system will exploit the computational power provided by connected computers. This scenario of use allows the system performances to scale together with the number of users, while a multi-processor architecture would bound the scalability to the number of processors that are readily available. The study has been carried out starting from our past experiences in designing Massive Battle [4]. Massive Battle is a MSE capable of animating autonomous actors with the purpose of reconstructing interactive scenes from a battlefield showing a number of platoons fighting each others.

A. Background

Peer–to–Peer systems: In peer-to-peer (P2P) network, computers can communicate and share files as well as other resources. As opposite to the client-server approach, participants are at the same level (peers). By means of a client users can connect to the network; communication among nodes is done by message exchanges aimed to: announce their presence in the network, ask for resources, serve resources requests. After the initial popularity of centralized Napster and flooding based networks like Gnutella, several research groups have independently proposed a new generation of P2P systems which are completely distributed and use a scalable Distributed Hash Table (DHT) as a substrate. A DHT is a self–organizing overlay network that allows to add, delete, and lookup hash table items. Proposed systems are based on various forms of distributed hash tables, they include Chord [5] (based on the hypercube), CAN [6] (based on the torus), P-GRID [7] (based on trees), Pastry [8], or Tapestry [9]. One of the reasons for the success of the DHT approach is that DHTs provide a generic primitive that can benefit a wide range of applications.

Massive Battle: Massive Battle is an example of *serious game* system that offers an effective way of simulating historical battles for the purpose of learning (e.g. providing new insights for battles to engage students) and to carry out historical researches (e.g. what-if scenarios). The simulation of historical battles also imposes some constraints on the number of agents the system is capable of simulate: as an example the Waterloo Battle involved ≈ 250000 soldiers, while Massive Battles, running on an off-the-shelves PC, is capable of simulating only ≈ 5000 units, at an interactive rate (≈ 25 frames-per-seconds). Massive Battle is implemented in C++ and is based on Ogre3d, a rendering engine, for this reason Massive Battle can be considered a reasonable approximation of a real DVE as a Massive Multiplayer Online Role-Playing Game (MMORPG).

B. Design Issues

The design of the DMSE has been carried out by addressing four main issues: world partitioning, world state propagation, self-synchronization and load balancing [10].

World Partitioning: A scene in Massive Battle is defined by a map, platoons and checkpoints: each platoon pass through the checkpoints assigned to it. To achieve scalability, we adopt a Geographic decomposition approach: the whole environment map is partitioned into regions. Regions are assigned to peers, by mapping both regions and peers to the DHT key space: regions as well as peers are associated with an ID computed by using a consistent hash function [11]. The peer whose ID is the closest to the region ID is dubbed *Region Master* and is responsible for that region (cf. Fig. 1).

Each Region Master is responsible for:

- Simulate all actors which belong to the region;
- Deliver the state of the region (that is the state of each actor which belongs to the region) to the peers whose AOI overlaps with the region;
- Handle *handovers* of actors between regions.

The choice of the world partitioning technique is important for the efficiency of the whole system. Two key factors need to be considered:

- Static or Dynamic Partitioning;
- The *granularity* of the world decomposition.

Dynamic partitioning can be used, for instance, in order to balance the workload across the peers, but on the other hand, the management of dynamic regions requires a large amount of communication between peers that consumes bandwidth and introduces latency [12], [13]. For this reason, in this work, we opt for a Static Partitioning.

Similarly the granularity of the world decomposition (that is, the region size and, consequently, the number of regions, which a given map is partitioned into) determines a trade-off between load balancing and communication overhead. The finer is the granularity adopted, the higher is the degree of parallelism that, ideally, can be reached by the system. However, due to regions' interdependency and system synchronizations, fine granularity usually determines a huge amount of communication. Our system is designed to be used with different granularity.

World State Propagation: In order to implement a P2P architecture for MMVEs, a communication infrastructure is needed to deliver messages to a wide group of users. Being multicast communication not available on geographical network, application-layer multicast provides a workaround [14]. A well-known mechanism used to propagate world state information is based on the Publish/Subscribe design pattern: a multicast channel is assigned to each region; users then simply subscribe to the channels associated with the regions which overlap with their AOI to receive relevant message updates. For instance, SimMud [15] uses Scribe [16], an application-layer multicast built on top of the DHT Pastry [8]. Scribe is decentralized and highly efficient because it leverages the existing Pastry overlay.

Self-synchronization: One of the goal of the design is to implement a *self-synchronizing* system (the whole simulation should proceed simultaneously, without the use of a central coordinator, which might represent a bottleneck). We use a standard approach to achieve a consistent synchronization of the distributed simulations. Each simulation is decomposed in time slots (henceforth steps). Each step is associated with a fixed state of the simulation. Regions are simulated step by step. Since the step i of region r is computed by using the states $i-1$ of r's neighborhood (the regions which confine with region r), the step i of a region cannot be executed until the states $i-1$ of its neighborhood have been computed and delivered. In other words, each region is synchronized with its neighborhood before each simulation step. The number of steps since the beginning of the simulation is used as a clock so that each event can be associated with a system timestamp.

Load Balancing: One of the motivations of the P2P infrastructure described here is to address the needs for more computing power. In order to better exploit the computing power provided by the peers of the system, it is necessary to design the system so that the simulation always evolves in parallel, avoiding bottlenecks. Since the simulation is synchronized after each step, the system advances with the same speed provided by the slower peer in the system. For this reason it is necessary to design the system in order to balance the load between the peers. We addressed this problem by relying on two factors: (i) the node id on a DHT are distributed uniformly, and this means that each peer in the system has equal probability of receiving a region (ii) it is possible to tune the granularity of world decomposition. This decision allowed us to implement a totally decentralized system: more effective load balancing techniques would have required some degree of coordination.

C. System Architecture

Like SimMud [15], we decided to adopt FreePastry, the open source version of Pastry [8], as the underneath network infrastructure and we used Scribe [16], the multicast infrastructure built on top of Pastry, to disseminate the simulation state and, at the same time, synchronize the system.

It is worth annotating here how we addressed the problem of distributing the simulation which is carried out by Massive Battle. Massive Battle has been designed and

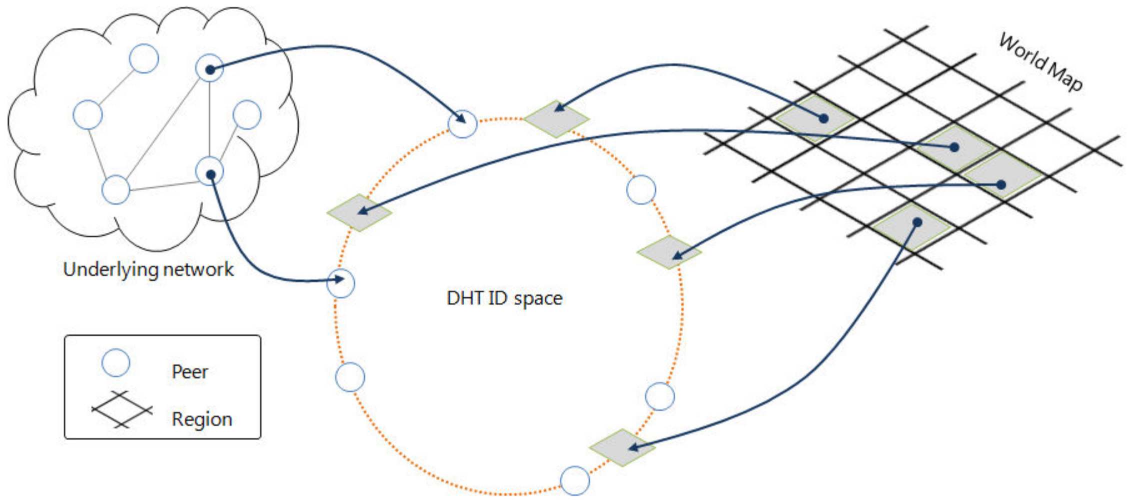


Figure 1. Geographic decomposition of the environment map over a DHT key space.

implemented as a simulation written in C++ to be executed on a single PC and this needed to be adapted for the network infrastructure that was implemented in Java. To address this problem we used Java Native Interface (JNI) that allowed us to invoke Java method from C and vice-versa. Once the technological issues have been worked out, we had to take a decision on how to distribute the computational load of the simulation. We just added a World State Propagation step after the Simulation and Rendering steps: each region r publishes the updates to all regions that are subscribed to r and (ii) r waits for the updates from all the regions r is subscribed to (r's neighbourhood). The World State as well as the self-synchronization logic is implemented in the Java part, while the Simulation methods are invoked once the buffer contains all the necessary information to perform a step. The P2P infrastructure is totally agnostic respect to the payload that is propagated among regions: upon receiving/transmitting the updates are handled (marshalling/unmarshalling) in the Simulation engine.

III. Tests

We performed a number of tests of the system in order to assess both scalability and the resilience of the system to uneven loads of calculation.

Test setting: Simulations were conducted on a scenario consisting of 64 regions (a 8×8 grid). On each run of the simulation the distribution of the regions to peers is decided by assigning randomly DHT identifiers to both regions and peers. Sixteen platoons of 100 soldiers (overall 1600 actors) were placed on the map. Each platoon follows a prefixed path which guarantees that all the regions become non-empty at least once during the simulation. It is worth noting that both the number of actors to be simulated and the number of iterations to be performed represent the workload of a test (i.e., the number of computations required for performing the whole simulation). Since the performances of the system are also influenced by some arbitrary factors (for example, the allocation of ID to peers and regions can lead to more or less balanced workload), we executed each test 10 times (we empirically observed that 10 runs of test are enough to obtain stable results). For each test we will present the results in terms of both means and boundaries. All experiments are performed on 16 mid-range PC having similar characteristics: Intel Xeon dual-core processor running at 2.80 GHz, with 2 GB of main memory. All the PCs are interconnected with a Gigabit Ethernet network. For the purposes of the performances evaluation we decided to not perform any rendering, in this way all the computational power is devoted to simulation, communication and to maintain the P2P infrastructure.

Scalability: The rationale behind this test is to evaluate the performance improvements obtained by increasing the number of peers. Thus, the question we would like to answer is: is the communication overhead, due to peers' interaction, dominated by the computational power provided by peers? The interaction between peers is a direct consequence of the fact that exchanging information between peers (e.g., actor positions, events, transitions across regions) is usually needed. The efficiency of the system is measured by evaluating the completion time of 500 simulation steps. We ran the simulation described above with $1, 2, 4, 8, 12$ and 16 peers (each peer is executed on a dedicated PC), in order to depict the scalability trend of our architecture. Figure 2 depicts the obtained results: the top chart shows both the mean (diamonds) and the maximum/minimum values registered during the tests. The bottom chart depicts a comparison between the average simulation times against the optimal linear speedup. The tests show that our schema presents

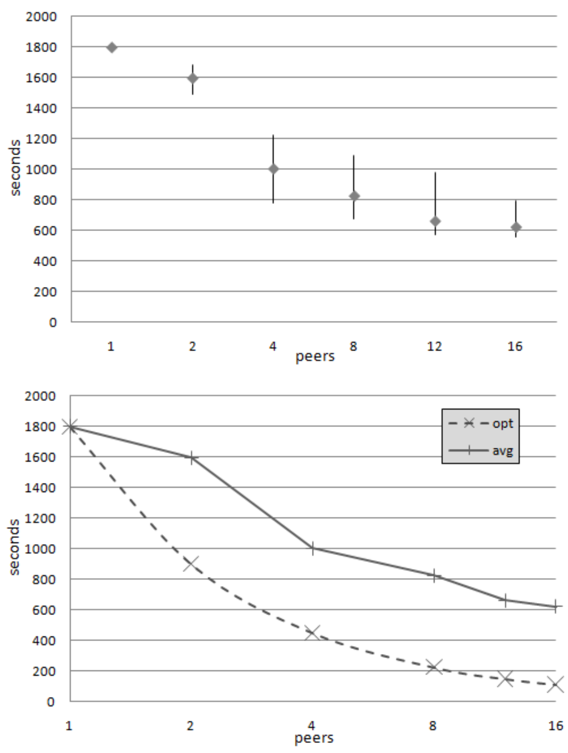


Figure 2. Simulation duration: (top) average and boundaries (bottom) compared with the optimal linear speedup.

almost linear scalability. However, as the number of peers increases, the gap between the results achieved by our architecture and the ideal speedup grows. We conjecture that this trend is due to the fact that in throughout the tests we used the same granularity of decomposition (64 regions) while, in order to obtain better performances, the granularity of the decomposition should be tuned in such a way that the ratio between the number of regions and the number of peers is constant.

Time distribution: The second test is focused on how the whole simulation time is spent by each peer involved in the computation. In this test the number of peers involved into the computation is always 16 while the number of simulation step is 1000. Our purpose is to determine how effective is the straightforward load balancing strategy adopted by our architecture and how much unbalanced distributions may affect the whole completion time. In order to evaluate the load balancing we compute, for each peer, the total computation time spent. By extrapolating this time from the whole simulation time we obtain the idle time (that is, the time spent for both communication and synchronization). The results show that the computation time spent by the peers during the computation is pretty variable. In order to provide the reader a better perception of the results, we show the time measures obtained by two simulations which reflect the best and the worst observed case. In Figure 3 each bar represents a peer, the dark portion of each bar represents the computation time.

Even in the best case (cf. Figure 3 (top)) we observe that more than 70% of the whole computation time is executed by only 5 peers. The best case simulation took 1198 seconds while the worst case took 3208 seconds; we also run a test of pure computation on a single PC without the P2P infrastructure and we measured, under the same circumstances, a running time of 2820 seconds. We also observed that the load unbalancing is not only due to an unbalanced distribution of regions to the peer. Indeed, the computation time required by each region is different and varies during the computation.

As said before the idle time comprises: (i) the communication time, which represents the time needed to deliver the state of the region after the computation of each step. This time does not depend on the workload distribution and is equal for each peer; (ii) synchronization time, that is the time spent by each peer waiting for slower peers. The latter time strongly depends on the system load balancing. We can estimate the communication time by considering the idle time of the most loaded peer. This peer has synchronization time close to 0 because it is always the last peer to complete the simulation step. Let us consider the second bar (from left) in Figure 3 (down): the communication overhead for the peer that obtained such performance, and consequently, for each other peer in the simulation, is only 38 seconds. As a result we have that the synchronization time represents a very big portion of the idle time. This consideration explains why the completion time is strongly influenced by the workload distribution.

Discussion: The tests show great variance of the system performances; nonetheless it is possible to see some potentialities of such architecture: in each of the test setting we had really fast simulation runs, together with slow runs. The running time is influenced by uneven load distribution and how to measure such a load can be a serious issue. It appears that the number of regions per peer is not correlated with the measured performance (see the first two bars in bottom part of Figure 3). More sophisticated load balancing techniques need to take into account the number agents each peer has to simulate.

IV. Conclusion

DMSEs, due to their scalability requirements, appear to be a natural application for P2P architectures. However DMSEs are quite different from classical P2P applications which are mainly devoted on sharing files as well as storages. On DMSEs the shared resources consist of CPU cycles while the purpose of the architecture is to maintain a distributed data storage (that represents the state of the simulated environment) keeping the latency as small as possible.

We presented an infrastructure that implements a DMSE and tests to assess the performances of such DMSE. We

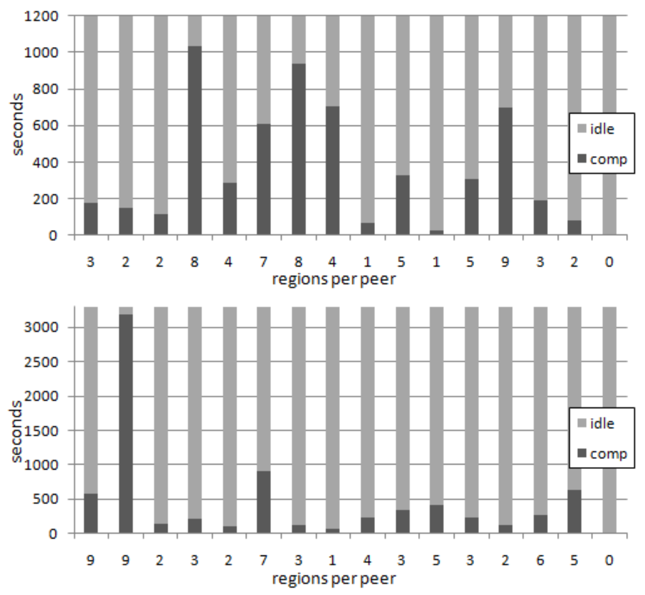


Figure 3. The figures depict how the simulation time is spent by each peer involved in the computation. The total simulation time is split into: computation time, time spent by the peer simulating a region of the map; and idle time (shown shaded) which comprises both the communication and synchronization time. Two cases are showed which corresponds to: (bottom) the worst case (maximum unbalancing); (top) best case occurred during all the tests. It is reported the number of region associated with each peer.

addressed and solved, in a totally distributed manner, the four main aspects of such architecture: world partitioning, world state propagation, synchronization and load balancing. The tests revealed that the architecture presents a quite good scalability, the communication overhead due to the peers interaction is dominated by the computational power provided by the peers. Unfortunately, such scalability is hard to achieve because of variability of the load balancing. A reasonable next step is to address the problem of load balancing by employing more sophisticated techniques, the challenge is to obtain such balancing in a distributed manner, without relying on a centralized load balancer. Preliminary results show that, in the most desirable situation of a balanced workload, the system obtains significant improvements in the measured performances.

References

[1] D. Pittman and C. GauthierDickey, "A Measurement Study of Virtual Populations in Massively Multiplayer Online Games," in *Proc. of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07)*. New York, NY, USA: ACM, 2007, pp. 25–30.

[2] J. Waldo, "Scaling in games and virtual worlds," *Commun. ACM*, vol. 51, no. 8, pp. 38–44, 2008.

[3] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[4] A. Boccardo, R. De Chiara, and V. Scarano, "Massive Battle: Coordinated Movement of Autonomous Agents," in *Proc. of the Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, 2009.

[5] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," in *IEEE/ACM Transactions on Networking (TON), Volume 11, No. 1*, Feb. 2003, pp. 17–32.

[6] S. P. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. of ACM Special Interest Group on Data Communication (ACM SIGCOMM '01), San Diego, CA, US*, Aug. 2001, pp. 161–172.

[7] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-grid: a self-organizing structured p2p system," *SIGMOD Rec.*, vol. 32, no. 3, pp. 29–33, 2003.

[8] P. Druschel and A. Rowstron, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. of the 18th IFIP/ACM Inter. Conference on Distributed Systems Platforms (Middleware '01)*, Nov. 2001, pp. 329–350.

[9] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," in *Tech. Report No. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley*, Apr. 2001.

[10] G. Cordasco, R. De Chiara, U. Erra, and V. Scarano, "Some Considerations on the Design of a P2P Infrastructure for Massive Simulations," in *Proceedings of International Conference on Ultra Modern Telecommunications (ICUMT '09), October 2009, St.-Petersburg, Russia*, 2009.

[11] D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, 1997, pp. 654–663.

[12] E. Buyukkaya, M. Abdallah, and R. Cavagna, "VoroGame: A Hybrid P2P Architecture for Massively Multiplayer Games," in *Proc. of the 6th IEEE Consumer Communications and Networking Conference (CCNC '09)*, Jan. 2009, pp. 1–5.

[13] H.-Y. Kang, B.-J. Lim, and K.-J. Li, "P2P Spatial Query Processing by Delaunay Triangulation," in *Proc. of the 4th InternationalWorkshop on Web and Wireless Geographical Information Systems (W2GIS 2004)*, 2004, pp. 136–150.

[14] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An Evaluation of Scalable Application-Level Multicast Built Using Peer-to-Peer Overlays," in *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, 2003.

[15] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games," in *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, 2004, p. 107.

[16] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, pp. 100–110, 2002.

[17] A. Nandan, M. G. Parker, G. Pau, and P. Salomoni, "On index load balancing in scalable p2p media distribution," *Multimedia Tools Appl.*, vol. 29, no. 3, pp. 325–339, 2006.

An Efficient JSD-Based Search on Interest-Based Hierarchical Clustering of Overlay Networks

Hasan Bulut*, Asil Yardimci*, Sercan Demirci**, Yagiz Kaymak†, Muge Fesci-Sayit**, E. Turhan Tunali†
{hasan.bulut, asil.yardimci, sercan.demirci, muge.fesci}@ege.edu.tr, {yagiz.kaymak, turhan.tunali}@ieu.edu.tr
* Department of Computer Engineering, Ege University, Bornova, Izmir 35100 Turkey
** International Computer Institute, Ege University, Bornova, Izmir 35100 Turkey
† Faculty of Engineering and Computer Sciences, Izmir University of Economics, Balcova, Izmir 35330 Turkey

***Abstract*— In P2P networks, peers share contents, especially video files, which represent their interests. However, the underlying P2P topology may not represent this interest distribution. Thus, one important aspect of constructing an efficient P2P network is to exploit the interest similarity among peers. In this paper, we propose a hierarchical clustering mechanism for constructing an overlay network that takes interest similarity among peers into account. By measuring the similarity among interests of peers and clusters, interest-based hierarchical clusters are formed by using Jensen-Shannon Divergence metric. The clustering performance metrics, accuracy and correctness, are reported on PlanetLab. For limited keyword collections, a novel Jensen-Shannon Divergence-based search mechanism is implemented. It has been observed that the integrated mechanism provides an efficient method and better performance as compared to classical keyword-based search.**

Keywords- peer-to-peer; clustering; interest; search; Jensen-Shannon Divergence

I. INTRODUCTION

Peer-to-peer (P2P) systems and applications provide an environment for sharing contents, instant messaging, videoconferencing, distributed computation and so forth. While some of them are unstructured, others are loosely or highly structured [1]. One of the challenging problems in P2P systems is to locate the shared content. Gnutella's [2] file query method is based on, flooding while FreeNet's [3] is based on random-walk technique. Peers in structured P2P networks exchange information through the overlay network. Structured P2P networks organize peers in some way to enhance the search performance. The most common type of structured P2P networks is the Distributed Hash Table (DHT)-based systems [4][5][6].

Content shared among peers may include a wide range of file types from documents, images to video files. These contents represent peers' interests. The term interest can be considered as metadata name or description of files such as movies, videos, music or contents of files such as documents.

Clustering peers with similar interest can enhance the search performance and message complexity of the query. However, it is difficult to characterize the interest profile of a peer or categorize the shared resource, i.e., a video file.

Constructing an efficient P2P network will depend on representing the interest of the peer and exploiting the interest similarity among peers. A P2P topology that exploits this interest similarity will form interest-based clusters over the overlay network and will gather nodes with similar interests into the same cluster or neighboring clusters. This will have effect on the search mechanism implemented within the system and will improve the search performance, as number of queries, number of messages per query or false-positive rates.

In this paper, we propose a hierarchical clustering mechanism for constructing an overlay network that takes interest similarity among peers into account, and a novel JSD-based search mechanism is implemented for limited keyword collections within the hierarchical system. Jensen-Shannon Divergence (JSD) [7] metric is used to measure the interest distance between two nodes. The JSD is used in information theory to measure the divergence between two probability distributions. In our case the distribution of the keywords provided by the peer or by the description of the video file is used. Peers join the system by measuring its JSD distance to clusters starting from the top of the hierarchy. The hierarchical architecture is then exploited to direct the search using the JSD distance between the query and the clusters. Although the architecture can be used for any type of content, we emphasize using the architecture for file types such as video and music which contain very limited number of keyword rather than documents with large keyword sets. The JSD-based search will exploit the hierarchical structure of P2P network to improve the search performance in terms of number of messages per query or false-positive rates.

The clustering performance metrics, accuracy and correctness, are reported on PlanetLab [8]. It has been observed that the integrated mechanism provides an efficient method and better performance as compared to classical keyword-based search.

The paper is organized as follows: In Section 2, related work is summarized together with their pros and cons. Section 3, presents the system design developed in this study. Performance results are reported in Section 4. Finally, concluding remarks are made in Section 5.

II. RELATED WORK

There have been studies in clustering of peers in a P2P network. In these clustering approaches, clusters are formed by using metrics such as delay [9], interest [10] or both [11], [12]. In delay-based clustering, nearby peers are clustered together to decrease the delivery time. Especially delay, jitter and packet loss ratio are among major performance parameters in video streaming. These parameters are affected from the underlying physical topology. Delay-based clustering is expected to reduce the delivery time, hence positively affect the above parameters. However in delay-based clustering, peers with similar interest profile may be placed in different clusters and this will lengthen the search time. Since interest set of generated queries are expected to be parallel with peers' interest profiles, clustering peers with similar interests is expected to shorten the search time. In [10] and [12] the shared content is hierarchically classified according to some predefined classification. However, it is difficult to classify all contents. Also, a peer may hold many shared content with different interest profile from each other, which makes the clustering of peers difficult if same approach is followed to cluster. Using JSD as an interest distance metric will enable us to distinct peers relative to each other. The study presented in [11] is closer to our approach in the sense that they use JSD to characterize interest/similarity among peers. However, the overlay architecture proposed in [11] displays a flat structure and it uses Dynamic Interest Landmarks (DIL) to place the peer to an interest region or to form a new interest region. A hierarchical structured network can be more advantageous in terms of scalability and message complexity if the nearby peers are clustered together. In [13], peers with similar interests create shortcut to one another. Peers use the underlying overlay network and only create these shortcuts when they issue a query. In [14], the concept of semantic overlay clusters (SOC) for super-peer networks is introduced. The approach is based on predefined policies defined by human experts. Peers join the clusters if their metadata model matches with the cluster's policy. In [15], peers with some common properties are interconnected with a super-peer. The super-peer tries to find the target file on behalf of the peer, by forwarding the request to other super-peers by using the charge-based flooding (CBF) algorithm, a look-up protocol for distributed multimedia objects at the super-peer layer. PAIS [16] classifies contents as books, images, music, etc., which are considered as content categories or subcategories. The architecture presented in [17] is founded on interest-based superpeer paradigm, in which nodes that have a common interest will form a neighborhood relationship and elect super-peers among themselves. The super-peers resolve queries on behalf of those clients. In [17], interest is considered as generic names such as movies, music, etc. However, it is difficult to strictly classify the content or the peer which holds many semantically different contents.

There are various search techniques used in P2P systems. Centralized indexing systems such as Napster [18] has performance bottleneck at the index server. Flooding-based systems such as Gnutella [1], send query every node in the system and consumes a lot of bandwidth and peer resources. Systems that use random walk technique, such as Freenet [3], reduce flooding messages to some extent, but do not prevent wasting bandwidth with excessive messages or duplicate messages. There are variations of random walk technique such as k-walker random walk [1]. Besides random walk, there are many techniques proposed to overcome the disadvantages of flooding, which can be classified as BFS-based techniques; such as iterative deepening [19], intelligent search [20]. In these methods, file names or IDs are queried within the system.

DHT-based systems scale well, but they use (key, value) pair to route queries and retrieve files. Moreover, DHT-based systems have strictly controlled topologies and are efficient, and effective for name-based searches. pSearch [21] describes two algorithms, pVSM and pLSI, where document information, which are represented as vector of terms, is stored in DHT-based overlay networks.

In this paper, we propose a novel JSD-based search mechanism which is explained in subsection B of Section III. In this search mechanism, the query is forwarded to a cluster based on JSD measure between the request interest set and the video interest set of that cluster. So the query will be forwarded to clusters semantically closer and extra messaging will be avoided. This method gives lesser number of messages and false-positive ratio per query with a very closer hit ratio compared to flooding and keyword-based search.

III. System Design

A. *Interest-based Hierarchical Clustering*

We propose a hierarchical clustering architecture for constructing a P2P overlay network that exploits the interest similarity among peers. Clusters are formed based on peers' interest proximity. Cluster leaders are elevated to the next higher level in the hierarchy to form another cluster together with other cluster leaders. Clusters except level 0 clusters are formed by cluster leaders from the clusters in previous level.

1) System Architecture

In this paper, we use the term node both for peer and cluster. In our architecture, a cluster at the lowest level (level 0) is formed from peers and clusters at level 1 and above are formed from leader peers of clusters from the previous level.

Each cluster consists of at most N nodes and one of the nodes in the cluster acts as the cluster leader. N can be determined based on the message complexity within the cluster. M ($M \leq N$) neighboring clusters form another cluster at the next higher level in the hierarchy and one of the nodes in that cluster again acts as the cluster leader of the newly formed cluster. In Figure 1, a 2-level hierarchy is depicted.

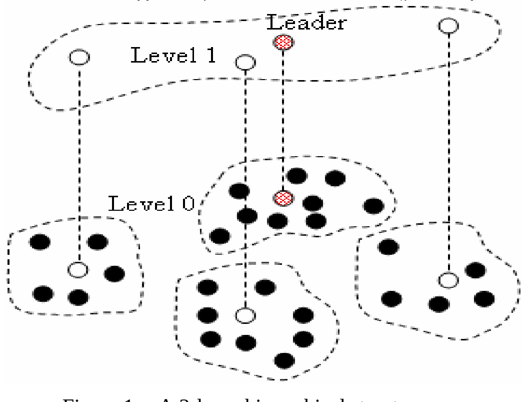


Figure 1. A 2-layer hierarchical structure

2) *Overview of JSD and Cluster Interest Set*

Let V be the set of all words in the vocabulary of all peers, $P_i(V)$ denote word frequency histogram in peer i, $v \in V$ be a word in the vocabulary and $p_i(v)$ be the percentage of the word in $P_i(V)$. Then, the Kullback-Leibler Divergence (D_{KL}) between peer i and peer j can be expressed as in [11]:

$$D_{KL} \overset{def}{=} (P_i(V) \,\|\, P_j(V)) \equiv \sum_v p_i(v) \log \frac{p_i(v)}{p_j(v)} \quad \text{(Eq. 1)}$$

The Kullback-Leibler Divergence (D_{KL}) given in (Eq. 1) requires a workaround to prevent division by zero. For this reason, similar to [11], we have also used Jensen-Shannon Divergence (JSD) given below.

$$JSD(P_i(V), P_j(V)) = \frac{1}{2}(D_{KL}(P_i(V) \,\|\, (\frac{P_i(V)+P_j(V)}{2})) \quad \text{(Eq. 2)}$$
$$+ D_{KL}(P_j(V) \,\|\, (\frac{P_i(V)+P_j(V)}{2})))$$

Peers' interests consist of a number of keywords where each keyword has weight associated with it, which represents the frequency of the keyword or the relative importance of the keyword from peer's view. Let I_i be the interest set of node i and w_k^i is the weight of k th word in interest set I_i. $I_i = \{w_1^i, w_2^i, \ldots, w_{ni}^i\}$ set of words in *node i*

We have normalized the interest set as follows: Let $P_i(w_k^i)$ be the normalized histogram value of w_k in node *i*. Clearly, $P_i(w_k^i) = c_k^i \,/\, \sum_{k=1}^{ni} c_k^i$ where c_k^i is the weight of $w_k^i \in I_i$ and $\sum_{k=1}^{ni} P_i(w_k^i) = 1$.

Normalized histogram distribution is computed for each peer interest to be used with JSD. JSD distance between peer i and j is computed as JSD($P_i(w^i)$, $P_j(w^j)$) as given in (Eq. 2).

Let the cluster contain the nodes $i = 1, \ldots, n_c$. Then $I_c = \bigcup_{i=1}^{nc} I_i = \{ w_1^c, w_2^c, \ldots, w_{nc}^c \}$ and $P_c(w_k^c) = (1/n_c)\sum_{i=1}^{nc} P_i(w_k^c)$ where w_i^c is the cumulative of weights of the same word in each peer interest set in the cluster.

JSD distance between a peer and a cluster is also computed as JSD($P_i(w^i)$, $P_c(w^c)$), where $P_c(w^c)$ denotes the cluster interest distribution. Note that it is similar to JSD computation between two peers. In our design, a peer and a cluster is considered as a node. The cumulative interest set of peers within a cluster is normalized to represent the cluster's interest distribution.

Similar to peer-to-peer and peer-to-cluster JSD computations, cluster-to-cluster JSD computations is performed as JSD($P_{ci}(w^{ci})$, $P_{cj}(w^{cj})$), where $P_{ci}(w^{ci})$ is the interest distribution of cluster c_i and $P_{cj}(w^{cj})$ is the interest distribution of cluster c_j.

3) *Process of a New Peer Joining System*

Each peer and cluster in the system has a unique identifier. Similar to other P2P systems [9] [11], there exists a rendezvous point (RP) required for bootstrapping mechanism. A new peer A first communicates with RP and requests to join. If, currently, there is no cluster in the system, RP asks peer A to form a cluster (let ID of the cluster be C00) at level 0 and assign itself (peer A) as the cluster leader of cluster C00. Peer A forms a cluster at level 0, and another cluster (cluster ID: C10) at level 1, whose members are cluster leaders of clusters from level 0.

If there is already a hierarchical structure available in the system, RP sends peer A, the list of nodes of the cluster at the highest level. Peer A, then, measures its interest (JSD) distance between itself and other nodes in that cluster. If peer A finds a node whose JSD distance is below a threshold, peer A joins the cluster. If peer A finds a node at each level until level 0, then it joins to the cluster it finds at level 0. If peer A cannot find a node below the threshold, it forms new clusters starting from that level to level 0, and it joins the cluster where it last satisfied the threshold criteria.

4) *Cluster Splitting*

Cluster splitting is started by the cluster leader if the cluster size exceeds a certain value. The cluster leader maintains nodes' interest sets within the cluster. It first finds two farthest nodes to each other within the cluster. From these two nodes, the farther node to the cluster leader is chosen as the temporary leader of the new cluster. The other node is taken as a reference node in the current cluster. Then the leader measures every node's JSD distance to the new cluster's temporary leader and the reference node in the current cluster. If a node is closer to the new cluster's temporary leader, then it is placed into the new cluster.

Leader election algorithm is run in the new cluster. The leader of the newly formed cluster also joins the cluster at the next higher level. Cluster split algorithm is run recursively for the cluster at the higher level if the cluster size of that cluster also exceeds the threshold. The pseudo code of split algorithm is shown in Algorithm 1.

Algorithm 1: Split Cluster

Input: C: Current cluster
Output: A new cluster (C') or null
Vars: C': New cluster; L: Leader of C; n1, n2: nodes (clusters)

```
 1: if (sizeOf(C) < upperSizeThreshold)
 2:       return null
 3: Find two farthest nodes within the cluster
 4: Assign the node closer to L to n1
 5: Add n2 to C'
 6: foreach node x ϵ C do
 7:       d1 = JSD(x, n1)
 8:       d2 = JSD(x, n2)
 9:       if (d2 < d1) then assign x to C'
10: end
11: return C'
```

5) *Cluster Merging*

Cluster merging is performed if cluster size drops below a threshold. If a cluster's size drops below the threshold, it notifies its parent. The parent cluster knows the number of nodes in each of its child clusters. The cluster which requests for merge, measures its JSD distance with other cluster nodes in the cluster according to the order of list provided by its parent. If it finds a node (cluster) below a threshold it checks whether the sum of cluster sizes is also below a limit. If it is, then merging is performed. Otherwise, it continues its search until it finds one or until the end of the list. The pseudo code of cluster merging is shown in Algorithm 2.

Algorithm 2: Merging Clusters

```
Input: C: parent cluster of Ck
Output: A new cluster (C') or null
Vars: mergeList: list of clusters to be merged
newSize = sizeOf(Ck): size after merge
sortedCList: list of nodes (clusters) within C sorted
according to their size
  1: foreach Ci in C (excluding Ck)
  2:       if (newSize+sizeOf(Ci) < upperSizeThreshold)
  3:           add Ci to mergeList
  4:       newSize += sizeOf(Ci)
  5:       else break
  6: end
  7: Merge clusters in mergeList into C'
  8: return C'
```

6) Leader Election

Leader election algorithm is run within the cluster if the number of joins and leaves after the last election exceeds for a specified number. Cluster leader is the one with the minimum total distance to other nodes in the cluster. Since the cluster leader maintains nodes' interest sets within the cluster, it computes every node's total JSD distance within the cluster. The node with the lowest total distance value is chosen as the cluster leader. If the new leader is same with the current leader, nothing further is done. Otherwise, the new leader is updated in the parent cluster (cluster at the next higher level).

B. Search Mechanism

Our goal is to provide a search mechanism for items represented with a small amount of keywords (we call interest set) such as metadata of video files. Our search mechanism is based on JSD measure between the request interest set and the video interest domain. The search starts by initiating a query from a peer in a cluster at level 0. Note that level 1 and higher level clusters are virtual clusters which help nodes to place itself in an appropriate region. Search algorithm is explained next:

The query is first submitted to the peer's cluster head (CH). CH has the entire video list. So it first makes a simple search on the list to find the video. If the video file is located, then the information of which peer(s) keep(s) the video file is returned to the requestor peer. If it cannot find the video file, then it forwards the query to its parent, which is the cluster head of the parent cluster. Cluster heads, at level 1 and higher levels, do not keep any video list, because it is not feasible and manageable to keep such a big list. Instead of keeping the video list, it keeps the interest set of videos for each of its child clusters. When the cluster head receives a query from one of its child clusters, it measures the JSD distance between the query's interest set and the video interest set of the child clusters. The query is forwarded to the child cluster head with which the measured JSD distance is below a threshold. Otherwise, the query is not forwarded. Hence, excessive messaging is decreased. This reduces extra messaging and number of false-positives.

Parent cluster head waits for the query results forwarded to the child cluster heads. If the cluster head receives fail result from each of its child cluster head to which the query is forwarded, it forwards the query to its parent cluster head. If it is at the highest level of the hierarchy, search terminates as a failed search. The result is sent to the requestor peer.

If the video file is found in one of the clusters at level 0, the result is sent to the requestor peer and also its parent is notified to terminate the search as a successful search. The pseudo code of search algorithm is shown in Algorithm 3.

Algorithm 3: Search Algorithm

```
Input: q: query
Output: -
Vars: p: peer initiating the query, Cp: peer's cluster, Cq:
cluster from which the query is received, Vci: video
interest set of cluster Ci, Clist: list of the clusters to which
the query is forwarded
  1: if ( (level == 0) and (video exists) )
  2:       send the location of the video file to peer p
  3:       if  (Cp != Cq)
  4:           send success result to parentOf(C)
  5:       return
  6: end
  7: foreach Ci in C (excluding Cq) do
  8:       if ( JSD(q, Vci) < jsdThreshold)
  9:           add Ci to Clist
  10:          forward q to Ci
               // each Ci will run this algorithm upon
               // receiving q
  11: if (fail received from all clusters in Clist)
  12:      if  (parentOf(C) is not null)
  13:          forward q to parentOf(C)
  14:      else send fail to peer p
  15:      return
  16: end
  17: if (parentOf(C) is not null)
  18:      send success to parentOf(C)
  19: return
```

IV. PERFORMANCE

We have analyzed the clustering performance; how accurately the peer is located at the cluster and search performance in terms of the number of messages generated, search time and false-positives produced per query.

In our tests, peers have interest sets and may keep many video files. A video file is also represented with an interest set, which is constructed from its title, category, and some descriptive keywords from its summary. Peers' interest set is formed from similar keywords. The number of keywords is expected to be in the order of 100s. So, the interest set forwarded from one peer to another one is in the order of 100s keyword which means around KBs.

In our tests, we have set the JSD threshold value to 0.6 which is determined empirically. So a node can join a cluster if the JSD distance between the joining node and the cluster is below the JSD threshold.

A. *Clustering Performance*

We have analyzed level 0 clusters. The goal of the clustering is to locate the nearby peers (based on interest closeness) into the same cluster as much as possible. That is, the goal of this clustering performance measurement is to analyze the accuracy of this placement.

Correctness and accuracy metrics are provided in [22] to measure clustering performance. We need to adapt these metrics in order to use them to analyze our clustering performance. In addition to these two metrics, we have also measured the diameter of clusters, average distance to clusters and its standard deviation. When we measure the node's distance to its cluster, we first excluded that node from its cluster and then measured the JSD distance to the cluster. We call node's cluster *the reference cluster of the node*. We consider a selection is correct if the JSD distance between a node and its reference cluster is within γ times the distance between the node and the nearest cluster leader, where $\gamma = \dfrac{JSD\ distance\ to\ the\ reference\ cluster}{JSD\ distance\ to\ the\ nearest\ cluster}$.

In this study, we use $\gamma = 1.0$, to determine the closer cluster. $\gamma \le 1.0$ means that the node is correctly placed into the cluster. Let $x \in G_i$, then we define reference cluster as

$$\overline{G_i} = G_i - x \qquad \text{(Eq. 3)}$$

Then, $$\gamma = \frac{JSD(x, \overline{G_i})}{JSD(x, G_j)} \qquad \text{(Eq. 4)}$$

where G_j is the nearest cluster to x. Let L_i and L_j be leaders of i th and j th clusters. Similar to [22] we also define the accuracy metric as follows:

$$acc = 1 - \frac{1}{N'} \sum_{\substack{i,j \in [1 \ldots k] \\ i<j}} \sum_{\substack{x \in G_i \\ y \in G_j}} \frac{|JSD(x,y) - JSD(L_i, L_j)|}{JSD(x,y)} \qquad \text{(Eq. 5)}$$

where $N' = \sum_{\substack{i,j \in [1 \ldots k] \\ i<j}} |G_i||G_j|$ and x is a node in cluster G_i and y is a node in cluster G_j. We define diameter of cluster G_i as $D(G_i) = \max\{JSD(x,y)\}$ (Eq. 6) *where* $x, y \in G_i$. We also define the average distance to the reference cluster as follows:

$$avg\_dist(G_i) = \frac{1}{\#of\ nodes} \sum_{x \in G_i} JSD(x, \overline{G_i}) \qquad \text{(Eq. 7)}$$

where $\overline{G_i}$ is defined in (Eq. 3).

We have used 356 nodes in PlanetLab environment to construct the P2P system based on our hierarchical clustering algorithm explained in this paper.

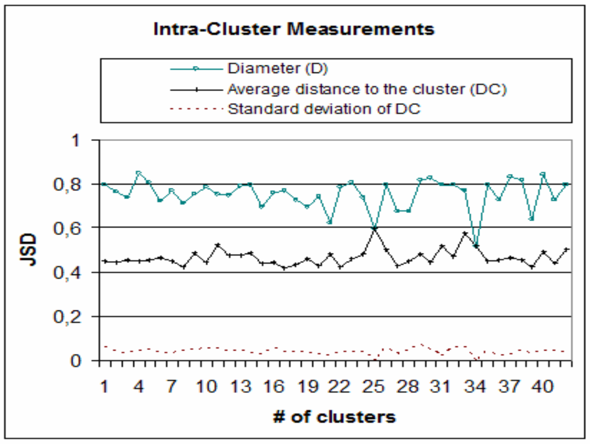


Figure 2. Intra-cluster measurement.

In Figure 2, we have shown the intra-cluster measurements. We have measured the diameter of clusters, average distances of nodes to their reference clusters and the standard deviation of the average distances. We have observed that the average of the diameters is 0,75 and average of the average distances to the clusters is 0,47, which is below the JSD threshold value. We expect the diameter be more than the threshold, because it represents the maximum distance between two nodes in the cluster.

We measured the accuracy of the P2P system we have constructed within PlanetLab environment, using 356 nodes as 86.3%. We have used (Eq. 5) to compute the accuracy. We have also measured the correctness of node placement according to (Eq. 4). Figure 3 shows the value of γ for each node in the system. Among 356 nodes, only 37 of the nodes can find a closer cluster than their reference cluster. That is, the percentage of correct placement of the system we have tested according to the criteria $\gamma \le 1.0$ is 89.6%.

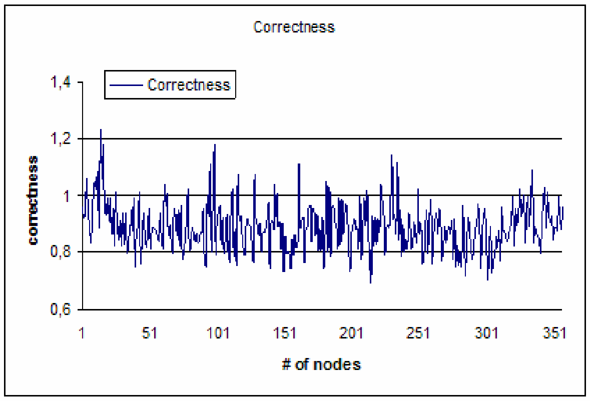


Figure 3. Correctnes of node placement

B. Search Performance

We have distributed video files across P2P system. Each video has a related keyword set (interest set) and an ID associated with it. Videos are distributed to peers according to their interest profiles; peers' interest profiles also reflect the interest set of the videos they maintain. Each peer may have different number of videos.

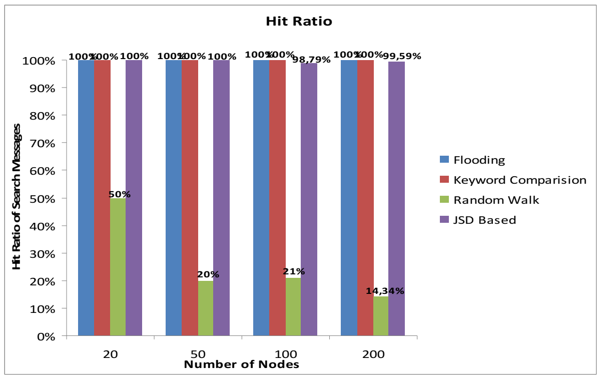


Figure 4. Hit ratio

In search performance evaluation, we have compared JSD-based search with keyword-based, flooding and random walk search. When search is at level 0, we have searched for video ID. However, in level 1 and higher levels, search is performed according to the interest set of the video file. In keyword-based search, we forward the query to a cluster as long as the video keyword set matches with the accumulated video keyword set of the cluster. In flooding-based search, the query is flooded to the entire network. In keyword-based and flooding-based search, we expect hit ratio be 100%. For JSD-based search, we have measured the JSD distance between the requested video file's keyword set within the cluster's accumulated video keyword set. If the JSD distance is less than the threshold, we forward the search to that cluster. We have constructed P2P systems with 50 nodes, 100 nodes and 200 nodes.

The hit ratios of keyword-based, flooding and random walk search and JSD-based search are shown in Figure 4. Hit ratio for JSD-based search remains closer to flooding and keyword-based search. Random walk performs a very poor hit ratio.

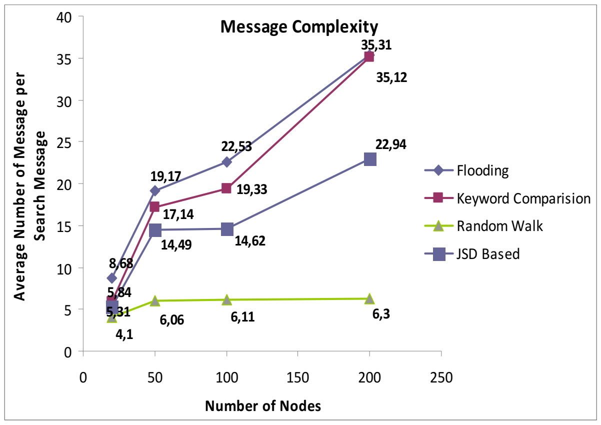


Figure 5. Average number of messages per search

The average number of messages distributed across P2P network per query is depicted in Figure 5. Random walk search has the least number of messages. Flooding and keyword-based search generates the highest number of messages per query. For 200 nodes case, JSD-based search is almost generates 37% less messages than flooding and keyword based search while maintaining over 99% hit ratio.

Comparing search times, JSD-based search performs better than flooding and keyword-based search (Figure 6).

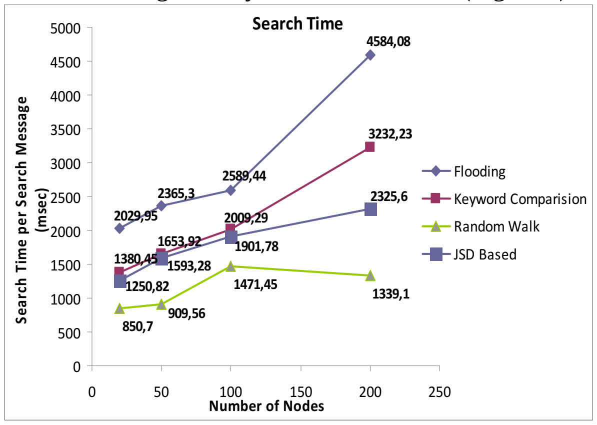


Figure 6. Average search time

As for the average number of hubs (number of nodes a query passes through), JSD-based search passes through less number of nodes (Figure 7). The difference between the average number of messages and the average number of hubs per search is that for the average number of hubs, we only count the number of nodes a search message passes through and for the average number of messages we include the messages used to notify the parent for search result in addition to the search messages.

Another feature to compare the search methodologies mentioned is the average number of false-positives; a query is forwarded to level 0 cluster with the expectation of finding the requested video within the cluster, however the requested video is not found in that cluster. The false-positive rates are shown in Figure 8; JSD-based search gives lower false-positive rate than flooding and keyword-based search.

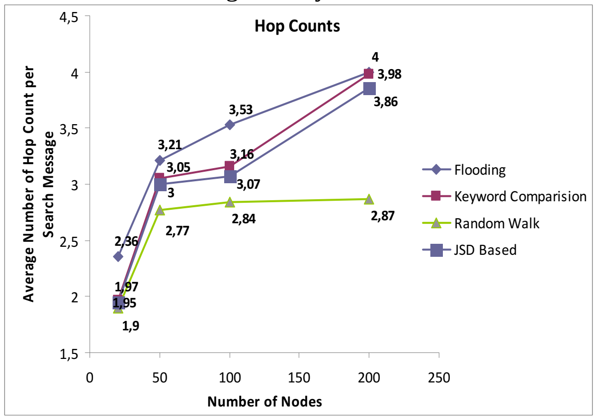


Figure 7. Average hop count per search

Random walk search gives the best result in message complexity, search time, hop count and false-positive rate. However, it produces the lowest hit ratio (14,34% for 200 nodes) while other methods gives more than 99% hit ratio.

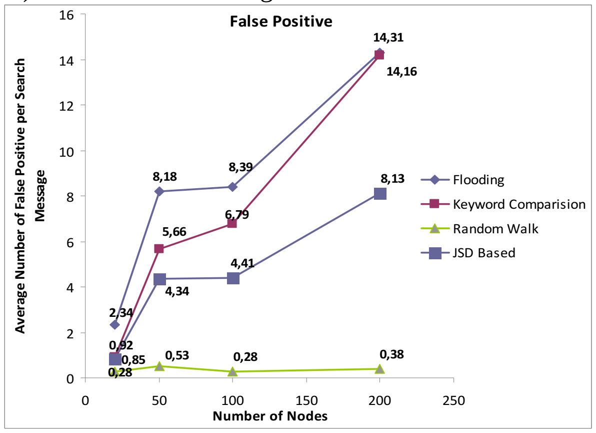


Figure 8. Average number of False-Positives per search

V. Conclusion

We proposed a hierarchical clustering mechanism for constructing an overlay network that takes account of interest similarity among peers. We have implemented a novel JSD-based search mechanism for limited keyword collections within the hierarchical system. We have provided clustering performance results that show how well the clustering mechanism works, with accuracy of more than 86% and with a correctness of more than 89% for the node settings we have used within PlanetLab environment. The overall performance of the search technique we proposed in this paper is better than random walk, flooding and keyword-based search.

VI. Acknowledgement

This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK), Project Number: 108M411.

References

[1] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", in Proc. of the 16th Int'l Conference on Supercomputing, pp. 84-95, New York, USA , 2002.

[2] Gnutella 0.6, http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html (accessed August 2010).

[3] Freenet, http://freenetproject.org/ (accessed August 2010).

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service For Internet Applications", in Proc. ACM SIGCOMM, pp. 149-160, San Diego, CA, USA, 2001.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in IFIP/ACM Int'l Conference on Distributed Systems Platforms (Middleware) , pp.329–350, Heidelberg, Germany, 2001.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-Resilient [Tolerant] Wide-Area Location and Routing," University of California, Berkeley, EECS Department, Tech. Rep. UCB/CSD-01-1141, April 2001, http://www.eecs.berkeley.edu/Pubs/TechRpts/2001/CSD-01-1141.pdf (accessed August 2010).

[7] A.P. Majtey, P.W. Lamberti, and D.P. Prato, "Jensen-Shannon Divergence as a Measure of Distinguishability between Mixed Quantum States", arXiv:quant-ph/0508138, 2005, http://arxiv.org/abs/quant-ph/0508138 (accessed August 2010).

[8] PlanetLab, https://www.planet-lab.org/ (accessed August 2010).

[9] X. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A Construction of Locality-Aware Overlay Network: mOverlay and its performance", IEEE J. Select. Areas Commun., Special Issue on Recent Advances in Service Overlay Networks, vol. 22, pp. 18–28, 2004.

[10] A. Crespo and H. Molina, "Semantic Overlay Networks for P2P Systems", Technical Report, 2002, Available at: http://www-db.stanford.edu/~crespo/publications/op2p.ps (accessed August 2010).

[11] Y. Wang, W. Wang, K. Sakurai, and Y. Hori, "On Studying P2P Topology Construction Based on Virtual Regions and Its Effect on Search Performance", in Proc. of The 3rd Int'l Conference on Ubiquitous Intelligence and Computing, pp. 1008-1018, Wuhan, China, 2006.

[12] X. Bai, S. Liu, P. Zhang, and R. Kantola, "ICN: Interest Based Clustering Network", in Proc. of the 4th Int'l Conference on P2P Computing, pp. 219-226, Zurich, Switzerland, 2004.

[13] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-based Locality" in P2P Systems. In Proc. of IEEE INFOCOM, pp. 2166-2176, San Fransisco, CA, USA, 2003.

[14] A. Löser, F. Naumann, W. Siberski, W. Nejdl and U. Thaden, "Semantic Overlay Clusters within Super-Peer Networks", in Proc. of the Int'l Workshop on Databases, Information Systems and P2P Computing, pp. 33-47, Berlin, Germany, 2003

[15] K. Watanabe, N. Hayashibara, and M. Takizawa, "A Superpeer-based Two-layer P2P Overlay Network with the CBF Strategy", in Proc. of the 1st Int'l Conference on Complex, Intelligent and Software Intensive Systems, pp. 111-118, Washington, DC, USA, 2007

[16] S. Haiying, "PAIS: A Proximity-Aware Interest-Clustered P2P File Sharing System", in Proc. of the 2009 9th IEEE/ACM Int'l Symposium on Cluster Computing and the Grid, pp. 164-171, Shanghai, China, 2009

[17] S. K. A. Khan and L. N. Tokarchuk, "Interest-based Self Organization in Group-Structured P2P Networks", in Proc. of the 6th IEEE Conference on Consumer Communications and Networking Conference, p. 1237-1241, Las Vegas, NV, USA, 2009

[18] Napster, www.napster.com (accessed August 2010).

[19] B. Yang, and H. Garcia-Molina, "Improving search in peer-to-peer networks", in Proc. of the 22nd IEEE Int'l Conference on Distributed Computing Systems, pp. 5-14, Vienna, Austria, 2002.

[20] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A Local Search Mechanism For Peer-To-Peer Networks", in Proc. of the 11th ACM Conf. on Information and Knowledge Management, pp. 300-307, McLean, VA, 2002.

[21] C. Tang, Z. Xu, and M. Mahalingam, "pSearch: Information Retrieval in Structured Overlays", ACM SIGCOMM Computer Communication Review, vol.33, no.1, pp. 89-94, 2003.

[22] X. Zhang, J. Liu, Q. Zhang, and W. Zhu, "gMeasure: A Group-Based Network Performance Measurement Service For Peer-To-Peer Applications", IEEE Global Telecommunications Conference, vol.3, no.17-21, pp. 2528- 2532, Taipei, Taiwan, R.O.C., 2002.

On the Performance Evaluation and Analysis of the Hybridised Bittorrent Protocol with Partial Mobility Characteristics

George C. Violaris[1], Constandinos X. Mavromoustakis[2]
Computer Science Department, University of Nicosia
46 Makedonitissas Avenue, 1700 Nicosia, Cyprus
[1]violaris.g@student.unic.ac.cy
[2]mavromoustakis.c@unic.ac.cy

Abstract— **Engaging mobility with file sharing is considered very promising in today's run Anywhere, Anytime, Anything (3As) environments. The Bittorrent file sharing protocol can be rarely combined with the mobility scenario framework since resources are not available due to the dynamically changing topology network. As a result, mobility in P2P-oriented file sharing platforms, degrades the end-to-end efficiency and the system's performance. This work proposes a new hybridized model, which takes into account the mobility characteristics of the combined Bittorrent protocol in a centralized manner enabling partial mobility characteristics, where the clients of the network use a distinct technique to differentiate between mobile and static nodes. Many parameters were taken into consideration like the round trip delays, the diffusion process, and the seeding techniques, targeting the maximization of the average throughput in the clustered swarms containing mobile peers. Partial mobility characteristics are set in a peer-tracker and peer-peer communication enhancement schema with partial mobility, allowing an optimistic approach to attain high availability and throughput response as simulation results show.**

Keywords- Hybrid Bittorrent protocol; P2P mobility; seeding strategies; performance evaluation.

I. INTRODUCTION

In recent years, the Bittorrent protocol has become an increasingly successful method for delivering end-to-end data, with reliability and efficiency. The tit-for-tat techniques [1], which are built in the protocol require peers to seed back the content they have received. Much research has been inspired in order to improve Bittorrent's performance [2][3][10] and [12]. Different scenarios and algorithms have been implemented and thoroughly tested [2][6][7][8][9], seeking ways to maximize the end-to-end performance using P2P techniques and approaches.

As in other P2P file-sharing schemes, performance depends mainly on the robustness of each node. Robustness depends on the temporal characteristics as well as on the spatial characteristics like whether the nodes are dynamically moving, etc. However there are certain features that need to be taken into consideration in order to enable higher performance onto a node-to-node sharing scenario. These, do not only rely on the behaviour of the connection between nodes, but on the techniques used to ensure quality of service through the protocol itself. In its current state, the protocol relies on the following ways, described by [2], to maintain the connectivity issue as follows:

- *Network size:* The number of peers in a Bittorrent network is important to determine metrics such as the request arrival rate, peer departure rate and the upload/download ratio in the bandwidth of each peer.
- *Efficient distribution:* Peers exchange pieces of a file, by a method called swarming [3]. In order to maintain efficiency, it is important to devise ways so peers do not get the same or very popular pieces. This is the reason the rarest-first policy [1], exists in the Bittorrent protocol; to maximise the potential of efficient distribution among peers.
- *Leech avoidance:* When a peer downloads without retaliation of the content they receive, the peer is called a leech. When there is a high ratio of these free-riding clients, the results are catastrophic for other peers. Therefore, mechanisms have been built to prevent this from happening *(one such example is the tit-for-tat algorithm, giving means to ensure fair transfers of data).*

Enabling these devices with mobility characteristics and utilizing them with the Bittorrent protocol, many restrictions arise. Peers are prone to failures and aggravate the end-to-end performance, whereas short connections times or sudden disconnections *(with chained unpredictable disconnections due to range and battery failures)* reduce the overall resource availability of the MP2P system. Moreover mobile peers are subject to limited bandwidths, both in the download and upload activities. Additionally, the protocol specifications make use of the tit-for-tat policy [1], which essentially means equivalent retaliation of pieces amongst peers. Since mobile peers do not contribute dramatically to other peers due to their limitation in bandwidth, other peers will perceive them as leechers, and therefore they will avoid providing content to them.

The present work, proposes a new hybrid policy for the Bittorrent protocol using P2P strategies enabling nodes with partial mobility characteristics, where the clients of a network use a distinctive technique to differentiate between mobile and statically located nodes. The model has been devised in order to enable the seeding peers that will split the uploading portion of their bandwidths towards a higher number of mobile peers, in order to enable enhanced network mobility. The scheme therefore can be hosted in larger scale Bittorrent clusters. The block-to-block and round trip delays are taken into consideration, enabling peer selection and seeding strategies to take place, targeting the maximisation of the average throughput in clusters containing mobile peers. The proposed scheme utilizes systems resources and comprises of a new model for disseminating information in a P2P system. The proposed scheme, hosts these partial mobility characteristics in a peer-to-tracker and peer-to-peer communication enhancement scheme, allowing an optimized approach to be applied for high resource availability in P2P networks with partial mobility characteristics.

The rest of our paper is organised as follows: Section 2 reviews previous work done on the Bittorrent protocol and similar static and non-static P2P methodologies. Section 3 provides information about the potential of mobility in Bittorrent, analysing the current problems mobile P2P transactions face and proceeds to explain the proposed hybridised model for dynamically changing topology systems, allowing mobility to peers. Section 4 discusses the simulation results and presents a performance analysis of the hybridised model, providing also discussion on seeding techniques and peer selection strategies. Finally, Section 5 concludes with a summary of the findings from the simulation study and discusses the future research directions the current research will take.

II. Related Work

Bittorrent performance is not only dependant to the protocol's peer selection algorithms and the tit-for-tat techniques. Certain simulation experimental studies show that along with optimised algorithms for content distribution, some minor alternations in the protocol's policies could significantly improve long term performance. Since the tit-for-tat policy of Bittorrent only takes place for a single file transfer at a specific moment in time [5], the sharing of old content is not rewarded and/or credited. Therefore incentives that elongate a content's lifetime are needed as files of high resource demands may become unavailable.

An analytical study in [2] has shown through a fluid model of the Bittorrent protocol that the average download time does not depend on the node arrival rate. Also, the study shows that there is a high chance that a peer will hold a specific block which other peers may be in need of. This concept allows for mobile clients to be ‘optimistic’ on having content delivered to them; however, some limitations which apply in the Bittorrent architecture do not enable these kinds of peers to use the full potential of their bandwidths.

It can be observed that Bittorrent uses a fixed default number, $u = 5$ reported in [4][6], of upload connections at any given time. The study reveals two significant problems. Firstly, the availability of full blocks to the network is delayed or postponed due to the high number of concurrent uploads occurring. Due to this, latency is significantly increased. Secondly, the seeding peer may be uploading to the downloading peer faster than the latter can receive blocks. This happens as the peer's bandwidth is congested on the downloading side, thus increasing the number of lost packets, leading to high redundancy in the network and unneeded repetition.

The simulation in [7] shows how Bittorrent works in general, while giving emphasis in super-seeding. Furthermore, the study shows how simulations can produce statistics for large scale experimentation that would otherwise be difficult to obtain. In relation to simulation studies, previous works [8], present interesting results concerning the use of MP2P architectures by using epidemic dissemination of data, resulting in high ratio of successful delivery. By using the storage backup nodes, the potential is to lower the packet delivery failure ratio and data corruption.

III. Mobile Bittorrent Protocol

The Bittorrent protocol is a peer-to-peer file sharing protocol. The protocol is more efficient for the transfer of large amounts of data *(usually in the hundreds of megabytes),* rather than smaller ones. It differs from other P2P techniques, as pieces of a file are divided between peers who enter a network and then exchanged in order to complete a file transfer. This allows peers with low bandwidth to participate in large data transfers.

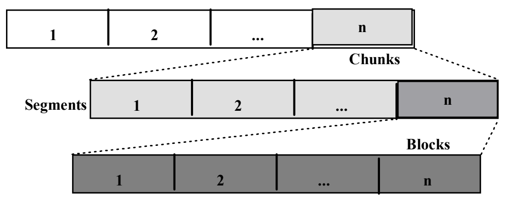


Figure 1: Blocks are the small pieces of data, made up of a few bytes, which are requested by peers. A serialization and reassembly of the blocks received constitutes a piece of the file.

Figure 1 shows the partitioning of a file, cut down compatibly in smaller sections, to be made ready for transfer from one peer to another. This process, known as swarming [1], allows peers in the same network to exchange these pieces *en masse*. The peers use different techniques to make sure they do not receive pieces they already have. If redundancy will occur, the network's latency will be dramatically increased, while throughput would drop.

The Bittorrent approach enables P2P systems to share efficiently any requested resources. However in a mobility-based framework, many different restrictions come to degrade the end-to-end availability. The clustered swarming technique allows mobile peers to exchange data more reliably than other P2P schemes [5], such as Kazaa, Gnutella and DC *(Direct Connect)* [15]. Though, due to the limited bandwidth capabilities which are encountered in wireless devices, the utilization of the Bittorrent protocol often becomes problematic.

A. Current Problems in the existing static framework

Problems of the protocol include increased latency while transferring small files, bandwidth problems, content unavailability, and leeching. One of the major inefficiencies of the protocol arises from a disproportionate distribution of content among peers, discussed in [9]. This kind of distribution allows peers to get different pieces from each other, which optimizes the download/upload rates between seeders and peers, but it also holds the potential of breaking a swarm, since the piece holders may not exist in the network at all times. This is not as common with large swarms; however strategies are needed to promote smaller network sizes for improved delay and maximization of throughput.

When a client first enters a swarm, they need to prove to their neighbors and other peers, as per-the-protocol's specification, that they are trustworthy enough in order to share information with. In order to achieve this, a small trial period of some minutes may pass, in which peers treat new peers with a bias, passing smaller amounts of data to them until they can prove that they will seed back what they've been given [1][3]. After this process takes place, peers start receiving much larger amounts of data. It is understandable therefore, that for smaller files it would not be worth sharing them through Bittorrent.

The transferring of data via the Bittorrent protocol puts a heavy load on the peers' bandwidth, observed by [10]. Since peers use a metadata file to locate pieces they need to download, the actual exchange of data is peer-to-peer and therefore a server is not involved. Thus, the bandwidth load occurs always on the client side and this is the main reason that service providers are opposing the use of the protocol.

There are often cases where a file is not as popular as others. When this is the case for extended periods of time, peers may not see the need to continue sharing this specific file, and therefore the seeding swarm dies out as per the lifetime scenarios of [3]. Content unavailability is a concept which is difficult to find easy solutions to. Even if an archive is unpopular, its value is many times unquestionable and therefore the archive needs to remain in circulation, especially if it is of scientific importance as many foundations may use P2P protocols such as Bittorrent to share these types of data. The essence behind this lays in the fact that even though peers still have the files stored on their storage media, they stop having them available to share in order to save bandwidth. However, even though valuable bandwidth is saved, peers entering a swarm to share an unavailable file will never be able to complete the transfer and the swarm will remain incomplete indefinitely or eventually die out.

One of the purposes of our algorithm aims in eliminating the above phenomenon through the implementation of partial mobility characteristics. This will allow wireless devices to evolve in a swarm through a higher download ratio even though their uploading bandwidths are not as high.*B.*

Hybridised Model with Partial Mobility Characteristics using the Mobile Bittorrent Protocol

The proposed model takes into consideration the difficulties which mobile clients, e.g., wireless devices, face while transferring files from other peers, most often static ones, through a Bittorrent network. Whilst the protocol offers an efficient way to share and distribute content, it has heavy requirements on bandwidth towards the client side. Content distributors benefit from peers using the protocol as they do not need to spend on acquiring large bandwidths and servers to distribute their content; rather only peers spend their bandwidth and CPU power to distribute the content. This is one of the reasons which internet service providers are often congested due to Bittorrent traffic. Users may not realize this, as the protocol makes it rather easy to share; however when it comes down to several network metrics, it is easily observable that content distributors benefit more than clients. In order to lay the grounds for a more efficient experience for the users, many of the problems described should be faced by devising the appropriate functionalities while not violating or altering the Bittorrent protocol.

Our algorithm also presents a way to control the latency between mobile and static peers. If mobile clients request data from other peers, the peers have the option of opening more connections, therefore serving more mobile peers at once. The reason of performing this, is because mobile peers have smaller bandwidths and limited connectivity, hence another peer may split their uploading activities between other mobile peers into greater than the default amount of connections allowed. A static peer is a non-moving peer or a normal peer itself; however it has larger bandwidth capabilities and therefore can provide more simultaneous connections, given that it transfers to mobile peers. This helps decongest not only the arrival requests from mobile clients, but also the network itself. As previously discussed, a Bittorrent seeder may upload to five connections at the same time. By implementing our strategy, peers with high latency issues will drop connections with specific peers in order to allow the faster seeders continue the transfer.

The reason the model is called hybridised, is because when peers with partial mobility characteristics are present in the swarm, the protocol's policies and tracker decisions remain unchanged. Therefore, both static and mobile peers communicate with the tracker on a similar level; however, the tracker makes different kinds of decisions based on what type of peers the requests are coming from. For instance, the tracker decides how many uploading connections a seeder may open, by manipulating metainfo about the downloader's bandwidth limitations, instead of maintaining a default number of connections that it can open. The tit-for-tat policy is still implemented as our model does not violate any of the Bittorrent protocol aspects. When a mobile client makes a request for bits, the tracker acknowledges the request by mapping more connections with available seeders. If the downloading section of the mobile peer's bandwidth is congested, the peer will ask the tracker to map fewer connections towards it. By using this technique, a client can ensure that their connection limitations are being used appropriately. In Figure 2 it can be seen that seeders are allowed to share towards more mobile peers than static peers, whilst their uploading bandwidth is split equally between static and mobile peers.

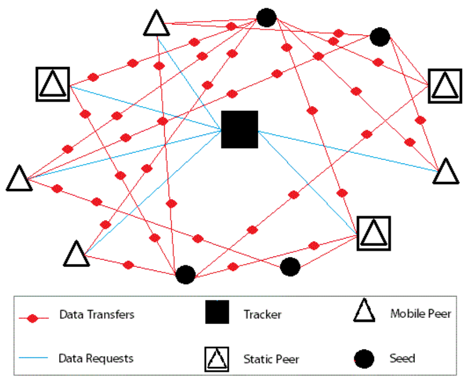


Figure 2: A showcase of the proposed model, presenting the distinction between mobile and static peers.

Bandwidth limitations still apply; therefore non-mobile clients with low bandwidth may not participate in seeding towards mobile peers. On the other hand, to ensure P2P fairness, mobile peers with a high bandwidth ratio may not be regarded as mobile per se.

Our model may be summarized in an algorithmic fashion for better understanding of the implementation, as shown in Figure 3.

```
Get swarmSize(N Peers);
Get_announce(peer, peer_type);
//tracker keeps track of peer_type in tables
Set_peer(peer, index);
//tracker indexes peer
Find_unchoked_peer();
Block_request(sourceIndex, destIndex)
    if destIndex.mobile == true
    {
        ConnectionSize.mobile = sourceIndex.uploadSize /
destIndex.Size;
        while (count != ConnectionSize)
        {
                new Connection(sourceIndex, destIndex, block);
                count++;  //increase counter to check if max
connection size based on bandwidth restrictions has been reached
        }
    }
    else
    {
        ConnectionSize = default; //default = 5.
        while (count != ConnectionSize)
        {
                new Connection(sourceIndex, destIndex, block);
              count++;  //increase counter to check if max allowed
connection size has been reached
        }
    }
Send_blocks(Connection);
```

Figure 3: Pseudocode of the proposed hybridisation model with partial mobility characteristics.

IV. Simulation Results and Performance Analysis

The proposed model's algorithm works without altering the Bittorrent protocol itself, but rather by implementing it on the client side and taking into consideration the dynamic changes in topologies. The use of seeders are proposed, who will have the ability to split their bandwidth capabilities and seed more mobile nodes at once, i.e., when there is a high latency from peer to peer transfer or when mobile peers are not getting a fair share of the content. This way, the overall latency of the transfers will be dropped significantly, thus lifting the bandwidth burden off the clients and allowing the content distributors give benefit to their users, both mobile and not.

A simulation was set up, running both a Bittorrent swarm with seeders who serve normally and a swarm in which seeders could serve more mobile peers simultaneously. The implementation-simulation of the proposed scenario was performed in Java programming language libraries as in [8]. We assume a system consisting of several mobile nodes, e.g., mobile users equipped with notebooks or PDAs and wireless network interfaces and that all devices are following a human-based activity (movements of nodes according to real-time pathways such as roads, streets, corridors, etc). Radio coverage is small compared to the area covered by all nodes, so that most nodes cannot contact each other directly. Additionally, we assume IEEE 802.11x as the underlying radio technology.

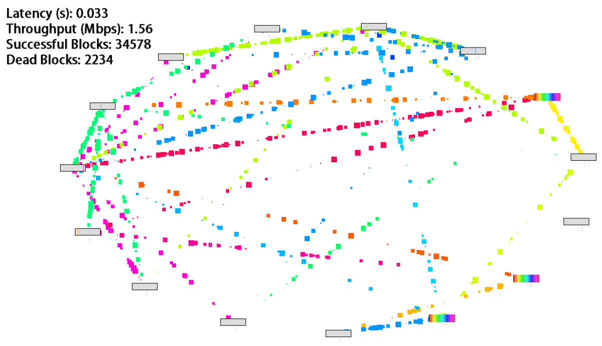


Figure 4: The simulation of the proposed model as viewed with graphical modes in order to enable visual representation of the Bittorrent resource sharing connectivity.

The simulation, presented in a visual format as seen in Figure 4, to further enable us to understand the techniques in which peers use the protocol to share data, derived metrics such as *average latency,* and *throughput*, since these are the targets of our model. Also, in conjunction to these, the *running time* for completing swarms of ratio 10 peers to 1 seeder was measured. In the simulation of the swarm with mobility characteristics, one seed was serving both mobile and non-mobile nodes, while a second was serving only non-mobile peers.

A. *Experimental Results*

As Figure 5 depicts, by comparing both the latency and runs of a normal Bittorrent swarm and a swarm which enables partial mobility characteristics, there is a difference in the latency response. In both cases the swarm contained the same number of peers, and the same number of seeders.

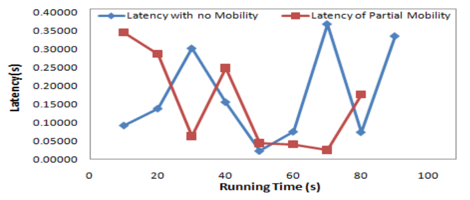


Figure 5: Comparing latency without partial mobility and latency with the running simulation time.

In the proposed algorithm, it is easily recognizable that the average latency from block-to-block has been decreased, due to the seeder being able to fill in the blanks, allowing more simultaneous mobility.

When referring to latency, we speak of block to block latency and not the initial lag which a peer experiences when connecting to a swarm. The reason for this is to minimize the transfer delays and therefore help the overall running time of the download. Equation 1 evaluates the delays from block to block.

$$T_\delta = K(t_x) - K(t_0) \tag{1}$$

where T_δ is the change in time from block to block, $K(t_x)$ the time a block has been released, and $K(t_0)$ the time a new block starts travelling towards destination.

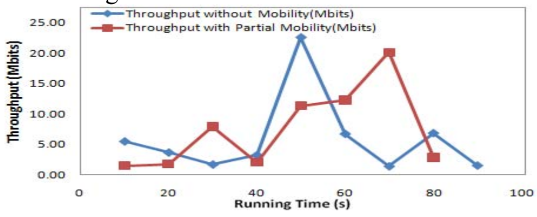


Figure 6: Throughput versus time for both partial mobility characteristics and no partial mobility characteristics in swarm clusters.

Whilst the throughput from peer to peer seems to peak in a normal swarm, this is only momentarily. As observed by the results extracted in Figure 6, the throughput of a swarm with partial mobility has a higher average through time, especially since partial mobility allows for a shorter running time in a network with both mobile and static peers. Formula in equation 2 shows how the throughput is represented.

$$C_{avg} = S / T_\delta \tag{2}$$

where C_{avg} is the average throughput, and S represents the number of blocks which have successfully reached their destination at any given point in time.

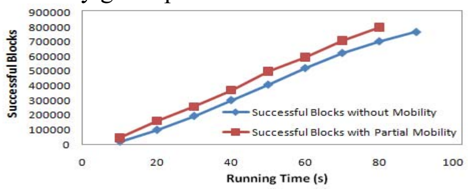


Figure 7: The number of gradual successful blocks through time for each simulation run.

At 50s, a swarm which has the ability of allowing seeders to upload to multiple mobile clients, may deliver 10% more blocks than a swarm without mobile characteristics.

While our model is not expected to behave satisfactorily on small scale swarms, it seems to be effective for large scale transfers, minimizing the network's overall latency while increasing throughput from peer to peer.

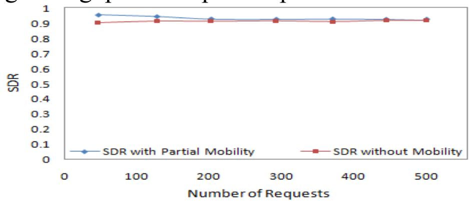


Figure 8: SDR with the number of requests occurring in Bittorrent resource sharing connectivity.

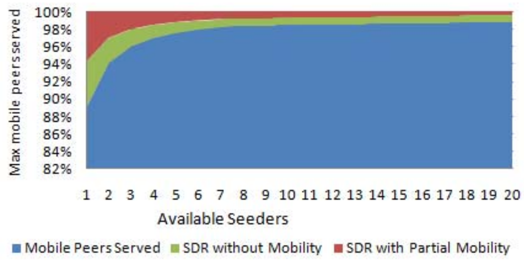


Figure 9: Outlining the maximum percentage of mobile peers which can be served based on the number of available seeders in a swarm.

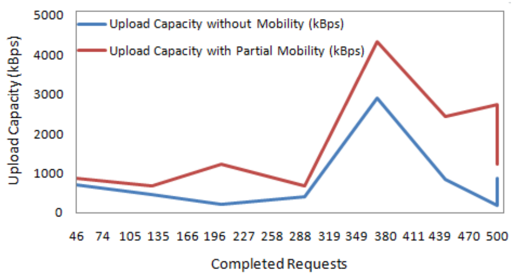


Figure 10: The upload capacity in kilobytes is given as a function of the completed requests. The upload capacity of a network with partial mobility characteristics is significantly higher per request than one with no mobility.

Additionally, not just content distributors would use this method, but any client could make use of the model. Of course, as in the Bittorrent protocol, some limitations still apply. The peers are still expected to keep uploading once they have acquired the entire file in order for the proposed method to work. Nevertheless, this will make it easier for peers to stay connected, since their bandwidth will only be used when other peers experience traffic problems.

B. Optimisation Techniques

1) Peer Selection

Even though some of the existing policies, such as *random piece first* and *rarest piece first*, are working on sufficient levels, the selection strategies for a swarm containing mobile peers cannot be maintained by simply these two techniques. The *choking* algorithm is a peer selection strategy which prefers clients with the highest upload rates [11], and this could work greatly towards the advantage of mobile peers in a swarm which uses our model, as naturally the non-mobile nodes hold the highest upload rates.

When peers finish downloading a file, thus becoming seeders, considering first that they have the available bandwidth, they can open more connections than the default, which is 5; however their uploading bandwidths will be split over those connections. This is done to allow mobile peers to enjoy downloading resources. Seeders could take turns in seeding towards mobile peers. On a similar note, if a peer has the uploading bandwidth to serve multiple nodes fast and efficiently, then they would be beneficial to the swarm, minimizing the latencies from block to block transfers. This follows well with the *observed upload rate* (OUR) in [12], which gives priority to peers which can upload data to other peers in a fast and reliable way.

Unlike the *LiveSwarm* protocol found in [13], our model does not need the seeder to push data to other peers. As we have observed from our experiments, Bittorrent's standard method of peers requesting data works more robustly than pushing for a few reasons. The first reason is that clients who are already choked or even who want to appear choked are not given the possibility of doing so. Secondly, as discussed in previous sections, the proposed model attempts to prevent seeders from uploading faster than mobile peers can download. Thirdly, the Bittorrent protocol does not need to be modified in order for our algorithm to work, as our model changes only the information the tracker and peers exchange.

2) Seeding Strategies

As in [14], optimistic unchokes would not be needed if nodes were able to calculate the upload bandwidth for the peers servicing it. In our model, since all peers communicate continuously with the tracker which constantly updates the metainfo it receives, a node could receive such bandwidth statistics from the tracker, thus eliminating the need for optimistic unchoking between peer and static seeder, and performing the unchoking algorithm only for non-static peers.

3) Efficient Distribution of Data

Concerning the notion of distributing data efficiently, suggestions show the importance of delay-sensitive responses to peer requests. Through the use of such defensive measures taken by seeding peers, the broadcasting of data may be efficiently redistributed by nodes which can make use of multicast technologies. Decisions should be made based on querying the neighbouring peers in a Bittorrent swarm, and through the collection of these feedbacks in order to create the appropriate responses.

V. Conclusion and Future Work

In this work, a new model concerning the involvement of P2P strategies with partial mobility characteristics was proposed, where clients in a network adopt techniques to seed more efficiently to mobile nodes. The round trip delays were considered and strategies for peer selection and seeding policies were suggested. We have entailed the potential of partial mobility characteristics in a peer-tracker and peer-peer communication schema, which allows an optimized approach in attaining high resource availability and lower packet failure ratios in mobile transfers through the use of Bittorrent.

Future research directions include the implementation of the seeding strategies and peer selection techniques. Moreover, the combination of other mobility schemas with our model gives the potential to create a truly mobile Bittorrent implementation.

References

[1] B. Cohen. Incentives build robustness in BitTorrent. In *First Workshop on Economics of Peer-to-Peer Systems*, pages 251–260, Berkeley, CA, May 2003.

[2] D. Qiu and R. Srikant, *Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks*, Portland, Oregon, USA: SIGCOMM, 2004.

[3] P. Michiardi, K. Ramachandran, and B. Sikdar, Modeling and Analysis of Seed Scheduling Strategies in a BitTorrent Network.

[4] A. R. Bharambe, H. Cormac, and V. N. Padmanabhan, Understanding and Deconstructing BitTorrent Performance, Microsoft Research, Tech. Rep., 2005.

[5] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, *Parallel and Distributed Systems Report Series: A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System*, Parallel and Distributed Systems Section. Netherlands: Delft University of Technology, 2004.

[6] A.R. Bharambe, C. Herley, and V.N. Padmanabhan. Analyzing and improving BitTorrent performance. Technical Report MSR-TR-2005-03, Microsoft Research, Redmond, WA, February 2005.

[7] K. Katsaros, V. P. Kemerlis, C. Stais and G. Xylomenos, A BitTorrent Module for the OMNeT++ Simulator. IEEE MASCOTS 2009, PP. 361–370

[8] C. Mavromoustakis and H. Karatza, Under storage constraints of epidemic backup node selection using HyMIS architecture for data replication in mobile peer to peer networks, Journal of Systems and Software, Elsevier Volume 81, Issue 1, January 2008, Pages 100-112.

[9] P. Michiardi, K. Ramachandran, and B. Sikdar, Modeling seed scheduling strategies in BitTorrent, in Networking 2007, 6th IFIP international conference on Networking, May 14 -18, 2007, Atlanta, USA — Also published as LNCS Volume 4479, May 2007.

[10] A. Legout, G. Urvoy-Keller, and P. Michiardi, "*Understanding BiTorrent: An Experimental Perspective.*" Technical Report, INRIA, Sophia Antipolis, November 2005.

[11] Vivek Rai , Swaminathan Sivasubramanian , Sandjai Bhulai , Pawel Garbacki, and Maarten van Steen, A Multiphased Approach for Modeling and Analysis of the BitTorrent Protocol, Proceedings of the 27th International Conference on Distributed Computing Systems, p.10, June 25-27, 2007.

[12] G. Wu and T. Chiueh, "How efficient is BitTorrent?" Proc. of 2006 SPIE Multimedia Computing and Networking Conference (MMCN 2006), San Jose, 2002. CA, 2006.

[13] M. Piatek, C. Dixon, A. Krishnamurthy, and T. Anderson. Liveswarms: Adapting bittorrent for end host multicast. Technical Report UW-CSE-06-11-01, 2006.

[14] A.R. Bharambe, C. Herley, and V.N. Padmanabhan, Some observations on bitTorrent performance, Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, June 06-10, 2005, Banff, Alberta, Canada.

[15] DC++, a Direct Connect client. Retrieved from: *http://dcplusplus.sourceforge.net/. (last accessed 14 June 2010)*

An Aggregation-Based Routing Protocol for Structured Peer to Peer Overlay Networks

Nicolas Hidalgo*, Luciana Arantes*, Pierre Sens* and Xavier Bonnaire†
*Université Pierre et Marie Curie, CNRS
INRIA - REGAL, Paris, France
Email: [nicolas.hidalgo, luciana.arantes, pierre.sens]@lip6.fr

†Department of Computer Science
Universidad Tecnica Federico Santa Maria, Valparaiso, Chile
Email: xavier.bonnaire@inf.utfsm.cl

***Abstract*—Structured peer-to-peer (P2P) overlay networks provide a scalable object location and routing substrate for large scale distributed applications. However, due to the great number of nodes of such systems, message complexity of their routing protocol may considerably increase network traffic and average node hops of a message. This paper presents a novel Pastry-based routing protocol for structured P2P systems, which is specially suitable for handling per node multiple message routing requests. Our protocol exploits message aggregation and implements a multi-slice mechanism which multiplexes the sending of aggregated messages. Experimental results on top of PeerSim show that our protocol can reduce the average number of node hops messages, and thus, the global traffic and load of the network.**

***Keywords*-Peer-to-Peer - Aggregation - Routing**

I. Introduction

Structured peer-to-peer (P2P) overlay networks such as Pastry [1] or Chord [2] are distributed self-organizing substrates which provide efficient routing and object location for large-scale applications. Each node has a unique $nodeId$ which is randomly assigned from an uniform identifier space. Objects have also unique keys taken from a large identifier space. Every key is mapped to the node whose $nodeId$ is numerically closest to the key. An object lookup service then routes an object lookup request to the node responsible for the key of the object. These systems usually present low object lookup latency since their routing protocol is based on Distributed Hash Table (DHT). DHT-based overlays route messages in a logarithmic number of hops, typically $O(Log(N))$, where N is the number of nodes of the network. Such a property allows nodes to maintain routing tables of small sizes. Furthermore, the system can scale up to a few millions of nodes.

In this paper, we are interested in P2P overlay networks where a node may have many lookup message requests to dispatch at a given time, i.e., a burst of messages. This happens, for instance, when a node wants to get several different objects at the same time (e.g. multi-query). A second scenario is hybrid/hierarchical network architectures composed by two types of network: a structured P2P overlay network, whose nodes are continuously connected to the Internet by wired links, and local networks, such as wireless networks, whose nodes can connect to the P2P overlay but such connections are intermittent. Thus, the P2P overlay network provides routing and lookup object service to the nodes that do not belong to it. Since the latter have temporary connections to the P2P overlay network, when they connect to it, several lookup request messages are probably sent to the node of the P2P overlay, which is their point of connection to the network. In other words, the P2P node behaves like a *proxy* to the former and every time a connection between them is established, the *proxy* node will have many messages to route, i.e., a burst of messages. An example of hybrid architectures is the *nano data centers*. Such centers refer to P2P architectures composed of controlled and stable peers, typically set-up-box, where domestic devices (e.g. PDA, PC, Mobile Phone, etc.) can be connected.

In such a context, we propose a routing protocol which combines several messages into a single one. Since the efficiency of a lookup service is usually measured as a function of the number of node hops to route a message to the node responsible for the message's key, the aim of our aggregation-based protocol is to reduce the average number of hops of messages. A second and important goal of our approach is to reduce message traffic for performance reasons.

We have also added to our protocol a multi-slice mechanism that divides an aggregated message into smaller ones which then are simultaneously routed to different continuous area of the identifier logical space. This mechanism reduces the average transmission delay of an aggregated message.

Our aggregation-based routing protocol was built using Pastry overlay [1]. Performance results obtained from experiments conducted on top of PeerSim [3] confirm that our protocol reduces both network traffic and the average number of node hops of a message.

The rest of the paper is organized as follows. Section

III describe our aggregation-based routing protocol and the multi-slice mechanism. Simulation performance results are shown in Section IV, while some related work are described in Section II. Finally, Section V concludes the paper.

II. Related Work

In order to disseminate information about membership in a ring-organized P2P system, Gupta et al. [4] use the concept of aggregation of messages and slice. The circular identifier space is divided into k equal contiguous slice. The members of each slice are represented by a leader node. Whenever a node detects a membership change it notifies its leader. This one aggregates the notification messages it receives into a single one during a period of time t and then dispatches it to all the other leaders. The latter then diffuses the message to the members of their respective slices. Mizrak et al. [5] also propose to split the circular identifier space into arcs (slice) and assign each one to a super-peer node. A super-peer is a high-capacity node which is responsible for routing messages to the nodes of its slice as well as to other super-peers. Similarly to our approach, message aggregation and/or the concept of slice are exploited by these works. On the other hand, in our multi-slice mechanism, it is not the identifier space that is divided into slices but the message M and the range of all keys of M does not necessarily cover the entire ring. Furthermore, aggregation of message is used for routing multiple messages and not for maintenance reasons.

Performance results presented in Section IV have shown that our protocol provides an effective mechanism to reduce lookup hops. Several works found in the literature such as One-Hop Route [4], EpiChord [6], and Kelips [7], have the same purpose. They are able to deliver a message in a fixed number of hops, typically $O(1)$. These protocols provide low latency lookup on small or low churn networks. However, if it is not the case, they usually present a lot of extra traffic for membership maintenance when compared to DHT-based protocols [8]. Contrarily to our approach, whenever the number of node hops is reduced, we observe a reduction in message traffic as well.

Some works [9][10] propose a hierarchical architecture for P2P systems. Usually, nodes are organized in disjoint groups which are connected by a DHT-based overlay network. Messages are routed between groups on the inter-group overlay and then routed to the node responsible for the key on intra-group overlay. Like these works, our aggregation-protocol is quite suitable for hierarchical architecture as explained in the introduction. However, contrarily to them, we consider that connectivity between nodes of different layers are not permanent and thus when it is established, nodes of the inter-group overlay receive many messages to route.

III. The Protocol

In this section, we present our aggregation protocol which is based on Pastry routing protocol. We have modified Pastry in order to support message aggregation, i.e, a single routing message M can be composed of several messages m. Our protocol also exploits a multi-slice mechanism which allows to split an aggregated message into several ones and each one is simultaneously routed to a different contiguous slice of the circular identifier space of the P2P overlay network. We should point out that even though our protocol is a Pastry-based one, it can be easily adapted to other structured P2P overlays such as Chord [2].

Aggregation of messages: The main goal of our protocol is to route multiples messages like a single one in order to reduce both message network traffic and the average number of node hops to deliver a message. To this end, the original protocol functions *route* and *deliver* of Pastry application programming interface (API) have been modified: instead of just one message, the *route* function accepts a buffer as input which contains k messages. The set of these messages, $G_m = \{m_1, m_2 \ldots m_k\}$, is then combined into a single message M, denoted *k-aggregated message*, which is sent over the P2P overlay network. Using the original Pastry routing protocol, M is firstly routed to the node whose id is the closest one to the key of the first message, i.e., m_1, of G_m. When M reaches its first destination, m_1 is delivered (function *deliver*) and the key of the next message, m_2 in this case, is chosen as the next destination of M. Such a routing/delivery process continues until the k messages of M are delivered.

In order to reduce network traffic, the logical proximity in the logical ring of those nodes that will take part in the routing paths of the gathered messages should be exploited as much as possible. With such a goal, messages in G_m are sorted by increasing order of their respective keys before being grouped into the single message M. The sorting is performed by taking into account both the logical proximity, defined by the DHT itself, of the nodes which store the keys of the G_m messages and the identity of the node which gathered the messages into M. Therefore, the first message to be routed is the first one whose destination node is the closest one to the latter. Notice that in the case where one or more messages have the same destination node, they will be consecutive in M and thus they will be delivered at the same time without any additional hop routing. It is also worth remarking that aggregation and sorting of messages are performed just once by the node that initially called the *route* function.

Multi-slice mechanism: The multi-slice mechanism is an extension added to the aggregation protocol presented above. It aims at dividing the k-aggregated message M into several messages and then multiplexing the sending of these messages. After the messages of G_m have been sorted and gathered in M, the multi-slice mechanism splits M in S messages, i.e., the logical space defined by the keys of the first and the last message of M is equally divided up in S messages M_s. It is worth remarking that the M_s messages may not have all the same number of k_s messages since the content of them depends on the distribution of the message's keys of M.

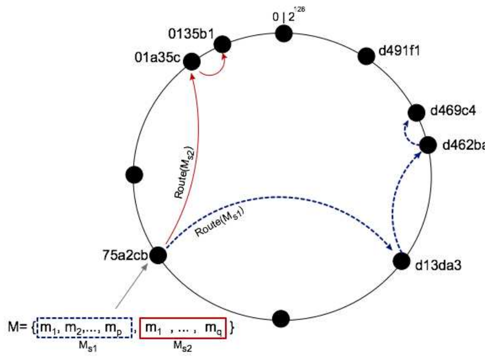


Figure 1. Message routing using the multi-slice aggregation protocol

All messages M_s will then be dispatched at the same time (parallel routing) improving the average message transmission delay when compared to the aggregation approach without the multi-slice mechanism. Each M_s will be firstly routed to the node that corresponds to the key of the first message in the respective M_s. Hence, each M_s will be routed to different slices of the P2P logical ring. In other words, the arc of the logical identifier space ring that encircles the keys of M are divided into S contiguous slices.

An example of the aggregation protocol with multi-slice is shown in Figure 1: the value of S is 2 and the keys of the first message of M_{s1} (p messages) and M_{s2} (q messages) are $d46q1c$ and $01a35b$ respectively.

The number of slices has an influence in the overall performances. On one hand, a high number of slices may induce an important gain in the average transmission delay of messages of G_m; on the other hand, it increases message traffic. Such a behavior is a direct consequence of multiplexing the routing of messages M_s since a smaller number of messages is presented at each ***Ms*** in relation to the single message M.

IV. Performance Evaluation

This section presents a set of results aimed at evaluating the performance of our aggregation-based protocol and the multi-slice mechanism when compared to Pastry.

A. Simulation environment and configuration

Experiments were conducted on top of the java-based P2P simulator PeerSim[3] jointly with a Pastry protocol plug-in.

The PeerSim Pastry plug-in is an implementation of Pastry[1] overlay over PeerSim. It exploits PeerSim event-based driven model and uses a traffic generator which sends random lookup messages to the system.

Our aggregation protocol implementation is an extension of PeerSim Pastry plug-in. In order to implement both the aggregation of messages and the multi-slice mechanism we have modified the event manager and the delivery functions of PeerSim Pastry. For simulating the per node burst of messages in PeerSim, i.e. the sent of k messages by a node, each node stores every message generated for it in a buffer, instead of sending it immediately. Thus, whenever the buffer of the node contains k messages, the node sends them. To this end, it calls the *route* function: either just once in the case of our protocol (a *k-aggregated* message M), or once for each of the k messages in the case of Pastry. The buffer is then emptied and the node waits for k new messages. To be able to simulate such a per node burst of messages, PeerSim is configured for presenting high message traffic. Notice that for a given k, message traffic increases proportional to network size since for all experiments the average number events per node is the same.

Several experiments were conducted with different configuration values for the number of participant nodes, number of messages k of a burst, and number of slices S. Each experiment was repeated 5 times and the results shown in the graphs are the average among the obtained results. The number of nodes of the system was fixed for each experiment, i.e., there was no failure nor churn and nodes did not leave the system. Messages could not be lost either. However, message transmission delays could vary. Aiming at evaluating both the scalability and stability of the protocols, three different sizes of networks were considered: 100, 1.000, and 10.000 nodes. The value of k varied from 20 to 50 while the number of slices was set to 5 and 10 when the multi-slice mechanism was activated.

B. Evaluation Results

The metrics used to evaluate our protocol are:

- *Network communication traffic* : total number of messages transmitted over the network during the experiment.
- *Average message size*: average size of the messages related to the above network traffic.
- *Average number of hops*: average number of hops required to route a message till its destination node.
- *Average message transmission delay*: average delay for transmitting a message till its destination node;
- *No extra hop message delivery*: number of consecutive messages of M delivered to the same node at the same time, i.e., without additional hop.

1) Network communication traffic: Figure 2 shows the message traffic in logarithmic scale for the original Pastry routing protocol and our aggregation protocol with different k values and network size. We can observe that message aggregation has a direct impact on the reduction of the overall number of messages in the network. Intuitively, the factor of network traffic reduction depends on k. In the best case, our protocol obtains up to 50 times less traffic than traditional Pastry routing. However, when the multi-slice mechanism is applied, the message traffic for a given network configuration increases as shown in figure 3 since the *k-aggregated* message M is split in S messages M_s (5 and 10 slices in the figure) which are then routed in parallel. In fact, the increase is proportional to the number of slices.

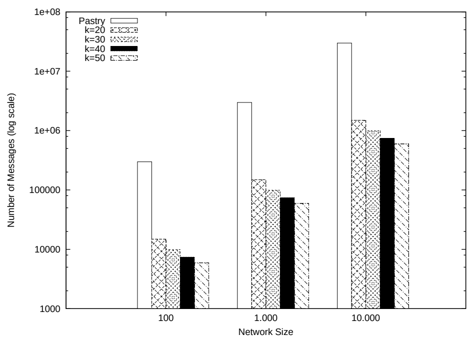


Figure 2. Message Traffic

2) Average Message Size: Table 1 shows the average size of messages for different network sizes, k in M, and number of slices S for the aggregation protocol. Every message generated by the traffic generator of PeerSim Pastry plug-in has 1024 bytes.

k	Slices	Network Size		
		100	1.000	10.000
20	1	10.601	12.718	19.609
	5	2.147	2.440	3.031
	10	1.066	1.217	1.497
30	1	16.353	20.002	30.160
	5	3.356	3.813	4.006
	10	1.677	1.908	1.937
40	1	22.156	27.545	40.560
	5	4.525	5.211	5.054
	10	2.293	2.617	2.411
50	1	28.047	35.451	50.560
	5	5.721	6.691	6.408
	10	2.908	3.347	3.290

Table I
AVERAGE MESSAGE SIZE (IN BYTES)

One direct consequence of the aggregation protocol is that the average size of messages increases considerably and such a grow depends directly on k. On the other hand, the multi-slice mechanism significantly reduces such a size by a factor proportional to the number of slices. However, we can remark in the table that even with the multi-slice mechanism, the average size of the messages increases with network size. This happens because the greater the size of the network is, the greater the number of hops of a message and the smaller the number of messages of M delivered without extra hops (see the "No extra hop message delivery" discussion bellow). In other words, in smaller networks, messages are routed in less hops when compared to larger networks and the size of a routing message M decreases faster than in larger networks due to the multiple delivery of messages to the same node.

3) Average number of hops: When routing a bundle of k messages, the aggregation protocol (AP) provides a significant reduction in the average number of routing hops necessary to deliver a message to the node associated with the key of the message in comparison with Pastry routing protocol, as we can see in Figure 5. Such an improvement is possible without any change in DHT table. It is in fact due to the routing of key-ordered messages which allows the exploitation of logical proximity of the nodes which correspond to the keys included in M. In addition, such a key-ordered approach enables the delivery of more than one message to the same node at the same time as discussed below, which also justifies why the average number of hops for the 100-node network is smaller than 1.

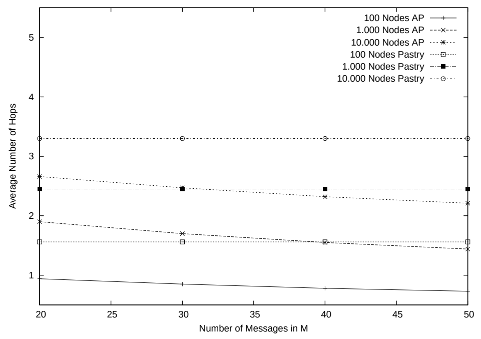


Figure 5. Average number of hops

We can also remark in the same figure that the effectiveness of the aggregation of key-ordered messages increases when the value of k increases for all network configurations. This happens because the probability that the keys of two messages of M have the same destination node increases for higher values of k.

In Figure 4 we can observe the average number of hops necessary to route the k messages of M (*Aggregation*), all the messages of a sliced message M_s when the multi-slice mechanism is applied (5 and 10 slices), and the k messages for Pastry. We can note a slight increase of the average number of hops when the number of slice increases which can be explained by the same reason of the previous figure: when S increases, the number of messages k_s of each M_s decreases, and therefore the effectiveness of the aggregation approach is reduced.

4) No extra hop for message delivery: Figure 6 presents the number of consecutive messages of M which are delivered to the same destination node at the same time, i.e., without extra hops for the aggregation protocol. As we can remark, for a given network configuration, the higher the value of k is, the greater the number of messages delivered without extra hops. On the other hand, for a given k, such a number decreases when the size of the network increases since the probability that two consecutive messages

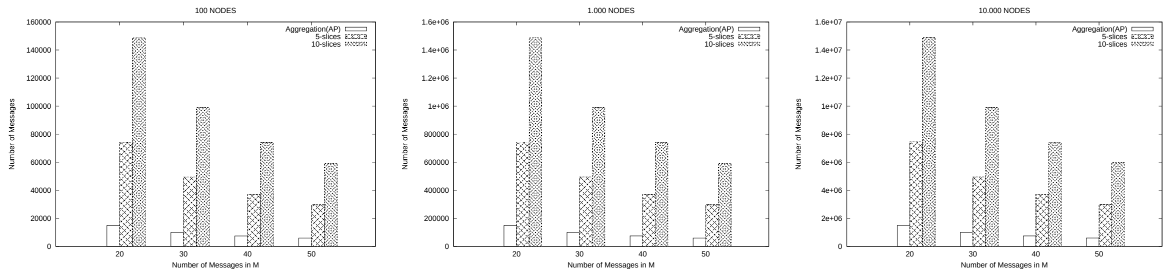


Figure 3. Message Traffic with Multi-Slice Mechanism

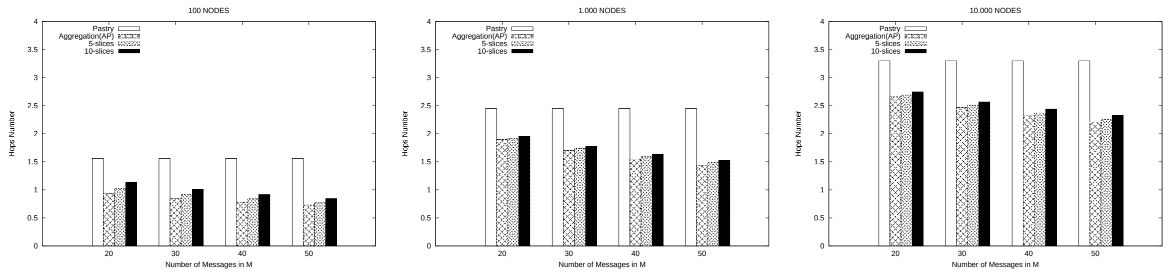


Figure 4. Average number of hops with multi-slice mechanism

of M should be delivered to the same node decreases for larger networks. However, as shown in Figure 7, when the multi-slice is applied for a given network configuration, the number of no extra hop decreases since the number of messages of M_s is inversely proportional to the number of slices S, i.e., the number of message that are delivered with no extra hops decreases when S increases.

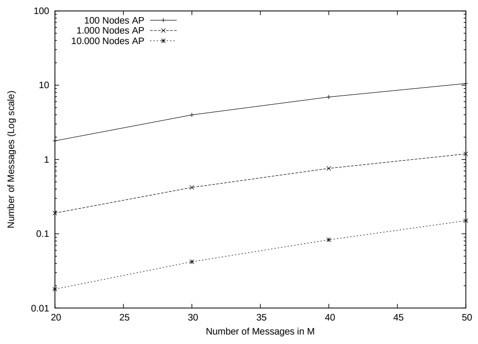


Figure 6. Average number of messages delivered with no extra hops

5) Average message transmission delay: Figure 9 compares the average transmission delay and its standard deviation of both Pastry and our protocol. For the former, such a delay corresponds to the average delay to deliver a bundle of k messages that are individually sent while for the latter it corresponds to the average delay to deliver all the k messages of a *k-aggregated message* M.

As can be observed in the figure, aggregation increases considerably the average accumulated transmission delay of messages. Furthermore, the standard deviation of our protocol is higher than Pastry's which, in its turn, is quite uniform. This happens because in our protocol, transmission delay of a message m of M depends on its position in M, i.e., when M is routed, the last delivered message of M has the highest transmission delay since it has to wait for all the other messages of M to be delivered before it. Such differences on transmission delay explain the standard deviation curves and the fact that it increases with k.

The average transmission delay of messages for different values of k and multi-slice configurations is shown in Figure 8 and its standard deviation.

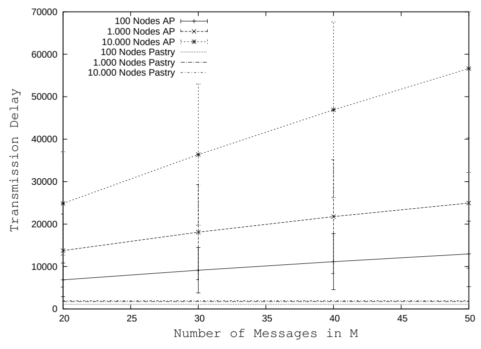


Figure 9. Average transmission delay and standard deviation

The multi-slice mechanism is an effective mechanism to reduce the average transmission delay of messages as well as the corresponding standard deviation. The gain provided by it allows to reduce up to 9 times the message transmission delay when compared to the aggregation approach without

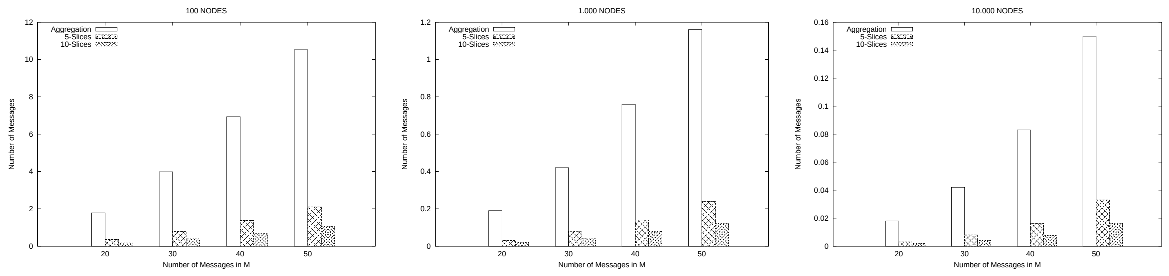


Figure 7. Average number of messages delivered with no extra hops and multi-slice

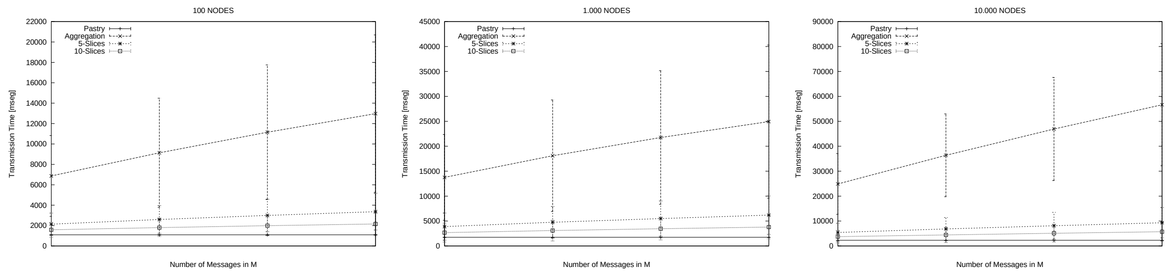


Figure 8. Average message transmission delay and standard deviation in multi-slice mechanism

such a mechanism. The explanation for the reduction is due to the simultaneously routing of M_s messages whose size are smaller than M.

It is worth mentioning that the aggregation protocol with multi-slice provides an interesting tradeoff between the transmission delay of a message and its number of hops: when compared to the aggregation protocol without multi-slice, the transmission delay of a message is reduced, but the average number of hops increases as a result of routing smaller aggregate messages.

V. Conclusion

This paper has presented a novel routing algorithm for structured P2P overlay networks which exploits aggregation of messages to reduce both the message traffic and the load of the network. Performance simulation results have confirmed that our protocol is an effective technique for reducing the average number of node hops of a set of messages without needing any change in the nodes' routing tables. Therefore, our protocol can be easily adapted to other P2P routing protocols.

We have also proposed a multi-slice mechanism extension for our aggregation protocol which provides parallel routing of aggregate messages. In the simulation experiments, an important reduction in the average transmission delay of messages was observed compared to the aggregation protocol without multi-slice.

References

[1] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware*, 2001, pp. 329–350.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[3] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," `http://peersim.sf.net`.

[4] A. Gupta, B. Liskov, and R.Rodrigues, "Efficient routing for peer-to-peer overlays," in *NSDI*. USENIX, 2004, pp. 113–126. [Online]. Available: http://www.usenix.org/events/nsdi04/tech/gupta.html

[5] A. T. Mýzrak, Y. Cheng, V. Kumar, and S. Savage, "Structured superpeers: Leveraging heterogeneity to provide constant-time lookup," in *WIAPP '03*, 2003, p. 104.

[6] B. Leong, B. Liskov, and E. D. Demaine, "Epichord: Parallelizing the chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2005.10.002

[7] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead," in *IPTPS*, vol. 2, 2003.

[8] J. Li, J.Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *NSDI05*, Boston, Massachusetts, May 2005.

[9] M. S. Artigas, P. G. Lopez, and A. F. Skarmeta, "A comparative study of hierarchical dht systems," in *LCN '07*, 2007, pp. 325–333.

[10] L. Garces-Erice, E. Biersack, K. Ross, P. Felber, and G. Urvoy-Keller, "Hierarchical p2p systems," in *Euro-Par 2003*, August 2003, pp. 1230–1239.

Flexible Macroblock Ordering for Video over P2P

Majed Alhaisoni, Mohammed Ghanbari
School of Computer Science and Electronic Engineering
University of Essex
United Kingdom
malhai@essex.ac.uk

Antonio Liotta
Electrical Engineering Department
& Mathematics and Computer Science Department
Technische Universiteit Eindhoven
The Netherlands
a.liotta@tue.nl

Abstract - **Peer-to-peer (P2P) is a promising technology for video streaming, and offers advantages in terms of re-configurability and scalability. It gains advantage from and share the resources owned by the end-users who are distributed around the Internet. P2P has shown an alternative solution for the traditional Client-Server approach limitations. However, due to the churn of peers, issues of video quality arise such as packet loss. This in turn degrades the QoS, then the QoE. Moreover, in current networking conditions, congestions and bottlenecks cannot be circumvented easily due to the increase in Internet traffic. Therefore, this paper introduces a novel combination of two well known techniques known as "locality awareness" and "Flexible Macroblock Ordering" (FMO). Locality-awareness plays a vital role in reducing the transmission cost among the peers whilst FMO is shown to be superior to other error resilience techniques in case of packet loss. However these two approaches have not been studied in conjunction. A comparative simulation-based study has been carried out for the proposed approach against a benchmark system, i.e., without introducing any error resilience technique. The results have shown better performance of the proposed approach in terms of End-to-End delay and video quality, as measured by PSNR.**

Keywords - P2P; Multimedi; QoS; QoE; FMO.

I. INTRODUCTION

Streaming video over P2P is becoming prominent due to the scalability of P2P networking [1]. Real time and video-on-demand services are getting more and more popular with the provided high speed of internet connections [2-3]. However, such application like video, is sensitive to different parameters such as end-to-end delay and packet loss [4]. All these parameters bring an unacceptable effect on the quality of the stream, and then will degrade the perceived quality by the end-users.

P2P systems have shown a good trend of delivering the multimedia to huge number of users over the internet [5]. The streaming is based on best-effort where any circumstances of congestion or bottleneck can not be alleviated easily. Therefore, introducing error resilience techniques without giving much attention to the criterion of selection among the peers would not be beneficial [6]. When peers are not chosen accurately, there is still a possibility that too many hops are involved in the transmission, resulting in congestion and bottlenecks. This will in return generate intolerable packet loss even with error correction techniques.

On the other hand, locality-awareness reduces the transmission cost among the peers and minimizes the packet loss [7]. However, in case of high churn of peers where there are not enough peers for switching over to the nearby peers, congestion takes place and bottleneck arises; so considering only this aspect might not be good enough in terms of packet loss and video quality.

The Internet capacity is increasing quickly. The number of users is fast growing and bandwidth-seeking services, such as video streaming, are becoming prominent. However, heterogeneity and congestion can cause erratic throughput, packets loss and delays. The challenge is therefore on how to provide an adequate quality, even at low bit rates, reliability in terms of loss and lastly low transmission latency.

Consequently, this paper introduces a novel combination of locality-awareness with a consideration of load distribution and FMO (Flexible Macroblock Ordering), which is a technique newly introduced in the H.264/Advanced Video Codec (AVC) [8]. By encoding the source video independently and error concealment at the decoder, error resilience techniques provide the reconstruction of video frames missing some of their constituent packets.

Moreover, delay reduction is also important but for streaming buffering can help to smooth out jitter at a small cost in start-up delay. Paradoxically, we have found that with the correct selection of peers on the overlay and the FMO technique, delay is significantly reduced in comparison with the scenario with no error techniques.

This is because the FMO technique does slice the frames, whereby any received slice can be decoded easily without waiting for the whole frame to be received. Owing to the introduced slicing, packets size become negligible and can be transmitted smoothly over the network, so transmission delay is significantly reduced. The FMO technique will be introduced in Section III.

A comparative simulation-based study has been carried out over a range of congestion levels to adjudicate the performance of the proposed solution. The proposed approach has been compared against a locally-aware system without introducing any error resilience technique.

The results have shown better performance of the proposed approach in terms of End-to-End delay and video quality, as measured by PSNR under varied network congestion levels.

II. PROPOSED APPROACH

The proposed approach combines two well known techniques known as network locality and error resilience techniques called Flexible Macroblock Ordering (FMO). In the following Section, a description of each technique is given, according to the way it is used in this paper.

A. *Locality-awareness*

Network efficiency (locality) is the ability to keep traffic as local as possible, which can be achieved by connecting to those peers which are nearby and changing the sources among the participants. Therefore, in the proposed method, a decision is made among the participant peers based on the measured RTT values by the monitoring system. Peers are prioritized on the order of lower RTT values, and the connections are setup based on these values. Consequently, this will not only maintain the network locality among the inter-communicating nodes but it will also improve the QoS and, hence, the user's quality of experience (QoE).

However, offering network locality only without changing the sources among the peers would be drastically impairing load balancing or, in other words, the load distribution between the network and the computing sources. Therefore, different techniques are embedded to this technique. The main aim of these techniques is to distribute the load among the participants and at the same time having the network locality not impaired. This can be shown in the next Section.

In order to maintain the load balancing among the contributing peers, different handover techniques have been embedded into the proposed approach. Two conditions trigger the handover among the interconnected peers:

Switching over: Since the network may experience various constraints such as congestion, bottleneck and link failures, the RTT values will be severely affected and may not be reliable. Additionally, these stochastic conditions will drastically affect the network locality and degrade the quality of service (QoS) parameters such as throughput, packet loss, and end-to-end delay. There is also another important requirement arising directly from the adoption of P2P: peers are not reliable entities and cannot be assumed to be always connected. Nodes may leave and join at unpredictable times. So, we must adopt a mechanism which allows the receiving peers (in client mode) to maintain a continuing reception of video, although the streaming peers (in server mode) are not constantly available.

One solution to this requirement is that any intending client should regularly update the neighbor's list and re-order them based on the lower RTT values. In our implementation, a switch over is applied to the first three lower RTT peers. This has been chosen according to our results in [9], where we found that the average of the active peers that usually a node is downloading from is 3 to 4. Therefore, in this model, the maximum number of sender nodes has been set to be three. This will help to avoid any action that may happen on the overlay, as the nature of the P2P is a dynamic, so it is highly expected for any node to leave, stop, or crash. In any case, though the receiving node will strive to obtain the stream from those peers that are nearby, handing over to other peers when connectivity is lost.

Enforced handover: Another favorable property in the proposed method is its computational efficiency. This can be achieved when the load is periodically distributed among the peers. Under normal network conditions, peers with lower RTT are selected; but when link latency changes, switch over is applied and new peers having lower RTT values are selected.

Some peers may not experience any constraints such as congestion, bottleneck, and link failures. The RTT values will not be affected severely and may not be changed, so those peers may become the best in every periodical check. Therefore, selecting them regularly would impair computational load balancing among the peers. To avoid this condition, enforced handover is applied.

Furthermore, to avoid pure randomness on the enforced handover process, network locality is applied into clusters of peers, named super-peers, similar to the one adopted in KaZaA [10]. Thus, peers are grouped and they are managed by a special peer, or a super node. Our experiments have confirmed that peers on the same cluster share nearly the same RTT values.

B. *Error Resileince Techniques*

Error resilience methods have been very helpful in reconstructing some of lost packet loss. In this regard, there are various techniques available which help in reconstructing the corrupted parts of lost packets and frames.

Related research on error correction methods in P2P has shown that MDC has mostly taken advantage of the path diversity available through Multiple Description Coding (MDC) or layered video [11]. In MDC, a video stream is divided into more than two slices (called descriptions) where in case of packet loss of one of the descriptions, it is still possible to be decoded notwithstanding at a reduced quality.

In this paper, FMO method is considered. In this error resilience technique, compressed frame data is normally divided into a number of slices each consisting of a set of macroblocks. In the MPEG-2 codec, slices could only be constructed from a single row of macroblocks. Slice resynchronization markers ensure that if a slice is lost then the decoder is still able to continue. Therefore, a slice is a unit of error resilience and it is normally assumed that one slice forms a packet, after packing into a Network Abstraction Layer unit (NALU) in H.264. Each NALU is encapsulated in an RTP packet. Accordingly, for a given frame, the more slices the smaller the packet size.

In H.264/AVC, by varying the way in which the macroblocks are assigned to a slice (or rather group of slices), FMO gives a way of reconstructing a frame even if one or more slices are lost. Within a frame up to eight slice groups are possible. Figure 1 (a) shows a simple way of FMO to carry on a row of macroblocks to a second row. However, this allows disjoint slice groups as explained in [12].

On the other hand, Regions of interest are supported, as shown Figure 1 (b). However, in this paper we have considered Checkerboard (known as disperses) slice group selection as shown in Figure 1 (c). This allows one slice group to help in the reconstruction of the other slice group (if its packet is lost) by temporal (using motion vector averaging) or spatial interpolation.

The checkerboard type stands apart from other types, as it does not employ adjacent macroblocks as coding references, which decreases its compression efficiency and the relative video quality after decode. However, if there are safely decoded macroblocks in the vicinity of the lost error concealment can be applied. Further illustration of the FMO types appears in [12] .

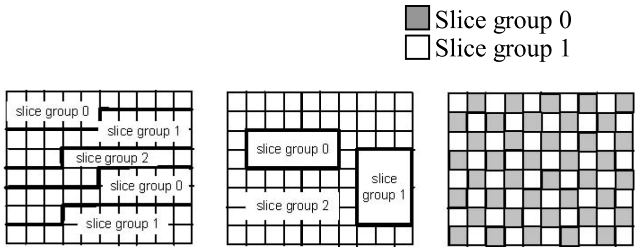


Figure 1. Example FMO slice groups and types (a) Continuing row (type 0) (b) geometrical selection (type 2) (c) checkerboard selection (type 1), adopted from [12]

In this paper, no more than two slice groups for this pattern are used, which is feasible for the CIF (352 X 288 pixel/frame) frames used. To reduce overhead, it is also preferable to choose the option in H.264, which prevents reference outside the slice group, though at some cost in coding efficiency.

There are various error resilience techniques available on H.264/AVC. However, the FMO technique has proved its effectiveness over other error techniques. A comparison study published in [13] has shown the effective of this technique over different techniques, both with error concealments and without it.

Figure 2 confirms the efficiency of the FMO technique among other error resilience methods. The FMO resilience technique is tolerable up to 50% of packet loss ratio. This technique well correlates with the dynamicity and heterogeneity of P2P networking.

III. PERFORMANCE EVALUATION

A. Simulation Setup

The proposed approach was implemented and tested on the ns-2 network simulator [14]. Senders and receivers were randomized and run several times for statistical purposes. Hence, receiver is selected randomly, and then based on that, the senders will be selected according to the locality techniques as shown in Section III (A). This gives the advantage of testing the proposed scenario under different conditions over the used topology.

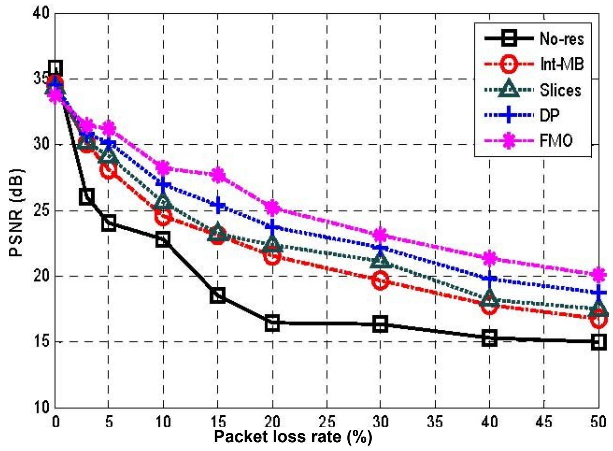


Figure 2. Comparison between several H.264/AVC error resiliency methods and no resilience (No-Res) with isolated errors, adopted from [13]

Various parameters were set on the used topology. First of all, each link has a bandwidth of 2 Mbps with equal length (delay). However, the actual delay will be according to the nodes distance of each other; so, all the participants' peers have the same characteristics. IP as the network protocol and UDP as the transport protocol have been chosen. For simulation of video traffic, the "Paris" video clip of CIF resolution with 4:2:0 format was H.264/AVC coded using JM15.1 and the video packets were sent from 3 peers to the receiver.

Two set of encoding were conducted. The first case without FMO as the video is encoded and decoded without the interference of any error resilience techniques. However, on the second scenario, with FMO, no more than two slice groups for this pattern are used, which is feasible for the CIF (352 X 288 pixel/frame) frames used. As commented earlier, in order to reduce overhead, it is also better to choose the option in H.264 that prevents reference outside the slice group, though at some cost in coding efficiency.

Secondly, in order to overload the network, it was essential to set the CBR background traffic to vary the network load and enable us to study the impact of two techniques under different loading conditions. The CBR traffic was setup from different sources to different destinations, with a 512 byte packet size. This background traffic operates during the whole duration of the simulations. Additionally, different bottlenecks have been created to measure the efficacy of the proposed approach. The congestion level of the network has been increased by adding new traffic to the streams during the simulation.

B. Simulation Scenarios

The following scenarios have been investigated in this paper. These are explained as follows:

- Locality-aware with FMO: in this scenario the video was coded with an added error resilience technique to help in reconstructing the lost packets. FMO correlates well with P2P as it starts decoding the video upon receiving a slice of the required frame and does not need to wait for the whole stream to be buffered.

- Locality-aware without FMO (No Res): in this scenario the video is coded as normal H.264/AVC without any consideration of error resilience technique. Moreover, decoding process is mainly dependant on the receiving of the whole stream which will increase the start-up delay and affect the QoE.

C. *Experimental Methodolgy*

In order to examine the two scenarios, they have been encoded and simulated independently. Every scenario has been run for 10 times where the presented results correspond to the average values of these simulations. In addition to that, the two scenarios have been run on the same network conditions in terms of network congestion and generated packet loss. Moreover, the same techniques of locality-awareness (Section III, A) is applied to the same scenarios which maintains the intercommunication among the peers on the network.

IV. SIMULATION RESULTS

In order to examine the performance of the proposed approach, evaluation metrics have to be carefully chosen. Henceforth, End-to-End delay and PSNR have been selected to adjudicate the impact of the proposed approach.

End-to-End delay reflects the consideration of locality-awareness and load distribution with range of packet drop ratio. On the other hand, PSNR will show the effectiveness of the combined error resilience technique with the locality and range of packet drop ratio. This is defined as follow:

$$PSNR = 10\log\frac{P^2}{E^2}$$

Where p is the peak value for a given pixel resolution, e.g., for 8-bits $p = 255$

Average end-to-end delay is defined as the average time delay incurred from the time when a data packet is sent from its source node until the data packet arrives at its destination node divided by total number of data packets delivered at the destinations. This includes all the possible delays introduced by intermediate nodes for processing and querying of data.

Figure 3 gives an insight to the achieved end-to-end delay of both scenarios. However, it is clear that the locality-aware with the introduced error resilience technique is achieving very low delay, whereas the normal scenario is achieving very high delay. This can be interpreted in two ways.

1) Locality-awareness has supported the stream to be transmitted by the link that has the least RTT values.

2) The proposed scenario with the FMO technique plays a vital role in this regard. During the encoding process of the video, the FMO technique can be encoded into many slices. In this paper, each frame is encoded into two slices as more than this would introduce extra signalling overheads. So, the frames will be constituted out of different slices which transmit minimal packet size. This will minimise the end-to-end delay as shown in Figure 3. It can be noticed that the end-to-end delay is decreasing exponentially with the increase of the packet drop ratio. Comparing the two scenarios to each other, the locality-aware-FMO is showing better results.

Another important factor that shows the effectiveness of the proposed technique is PSNR, which shows the quality of the transmitted stream over the network. In other words, it gives an insight of the expected quality of the stream as received into the end users under the examined conditions. FMO and error concealment were applied. Figure 3 shows the average video quality as a result of packet losses recorded in the packet loss traces in simulations for the data points.

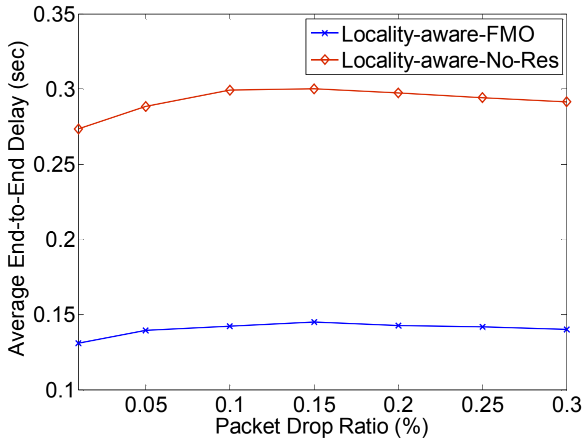


Figure 3. Average End-to-End Delay vs. packet drop ratio

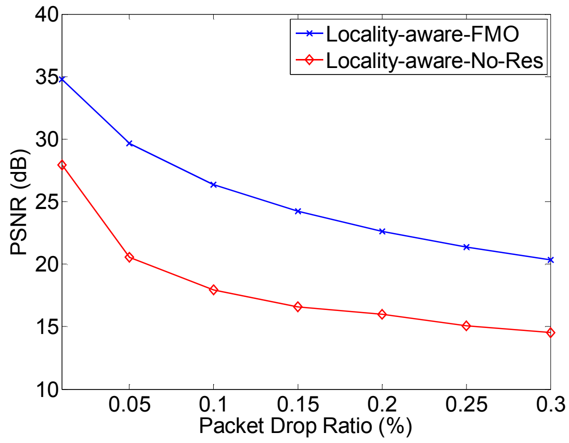


Figure 4. PSNR vs. packet drop ratio

Figure 4 shows how the FMO technique is achieving a higher quality. This quality is accompanied by a smooth end-to-end delay (Figure 3). According to Figure 4, FMO correlates interestingly with P2P networks due to the dynamicity of the network. Also, in case of churn of peers any intending peer can help in transmitting the FMO slices to the receiver, so receiving any slice of a frame gives the chance for the receiver to start the decoding process.

Looking to the other scheme without any kind of error correction, it is clear that above 5% of packet loss, the quality of video will be degraded severely. Accordingly, it is obvious that FMO has proved its effectiveness on transmitting the video even under 30% of packet loss which is very interesting. Table 1 [15] gives an indication of the acceptability threshold of H.264/AVC without any error resilience technique.

Packet loss ratio [%]	QoE acceptability [%]	Video quality playback
0	84	Smooth
14	61	Brief interruptions
18	41	Frequent interruptions
25	31	Intolerable interruptions
31	0	Stream breaks

Table 1. Quality of experience acceptability thresholds

V. RELATD WORK

As we are dealing with network, QoS, QoE, P2P locality awareness and error resilience techniques in this paper, an overview of different studies that have looked at these topics individually is given.

Thomas *et al.* [16] proposed a distributed hash table which is suitable for high dynamic environment. Their work was designed to maintain fast lookup in terms of low delay and number of routing hops. In their work, number of hops was the main metric which used to determine locality-awareness. According to their work, neighboring nodes are grouped together to form a clique. Nodes share the same ID in a clique; moreover, the data will be replicated on all the nodes on the clique to avoid data loss.

Additionally, a clique has an upper and lower bound in terms of the number of nodes, such that cliques are forced to merge or split. Another aspect of their work is to assume that all the nodes are distributed uniformly in a two dimensional Euclidean space. However, this may not work in a large network such as the internet. In addition, the link structure is updated periodically in order to establish a structured network. On the other hand, their proposal is based on pining nodes to join the closet clique which will drastically introduce extra signaling overhead.

Another study similar to [16] was conducted by Shah Asaduzzaman *et al.* [17]; their proposal was built on top of [16], with some modifications by introducing stable nodes (super-node) and replicating the data among the stable nodes only. However, their proposal elects one or more stable nodes of highest available bandwidth in each cluster and assigns special relaying role to them. Their work is based on a combination of tree and mesh architectures where the nodes on the clique form a mesh and the stable nodes are connected in a tree structure.

For each channel, a tree based is formed between the stable nodes including only one stable node in each clique. However, stable nodes are elected based on their live session. So, in this case a clique may have more than a stable node. The downside to this approach is that the relaying nodes (super nodes) are forming a tree, so reconstructing them in case of failures and peers churn will be costly and can introduce some latency.

On the other hand, another study in [18] proposes different techniques where the video stream is divided into different flows that are transmitted separately to increase parallelism and, hence, reduce transmission latency. The authors use the PSQA (Pseudo-Subjective Quality Assessment) technique that gives an estimate of the quality perceived by the user. This study was concerned on how to influence and improve on quality (as measured by PSQA). They introduce three cases: sending a single stream between nodes; sending two duplicate streams via different paths; and sending two disjoint sub-streams whose union recreates the original one.

Overlay locality is also studied by [19], where the authors make use of network-layer information (e.g., low latency, low number of hops and high bandwidth). However, we use a different distance metric based on RTT (round trip time) estimations, to prioritize overlay transmissions.
Hefeeda et al. [20] have proposed a mechanism for P2P media streaming using Collectcast. Their work was based on downloading from different peers. They compare topology-aware and end-to-end selection based approaches.

Authors in [21] propose a technique, where the peers on the overlay are chosen based on their mutual physical proximity, in order to keep traffic as localized as possible. A similar approach is described in [22], where they measure the latency distance between the nodes and appropriate Internet servers called landmarks. A rough estimation of awareness among the nodes is obtained to cluster them altogether, as in [7, 23].

In [24] , authors proposed system for the live and on-demand media streaming using MDC (multiple descriptions coding) layers which presents better performance in case of network congestion.

In [25], congestion control mechanisms using bandwidth estimation models have been proposed. In [26] a combination of MDC and path diversity has been proposed. In this study, a TCP-Friendly algorithm has been used where every peer can send portion of the descriptions over various path.

Another study has introduced MDC over P2P is [27]. Their study is based on active measurements of network links. They have followed the end-to-end selection technique introduced in [20]. Moreover, a cluster approach is introduced to avoid the risks arising from the sharing of same bottleneck link. The video is composed of MDC layers from various peers on the network.

By contrast to the abovementioned works, our proposal aims to introduce the benefits of error resilience technique (FMO) and also to study the impact of this newly technique with the consideration of overlay and underlay networks harmonization. Therefore, we study the combination of two techniques, FMO technique and network locality.

Looking at previous studies, we can say that our main contributions are:

1) To study a new combination of existing techniques (cross-layer optimization, localization, forced handovers, and error resilience technique (FMO)).

2) To take the perspective of the network operator, in trying to harmonize overlay and underlay networks.

3) To quantify the goodness of this proposal under a range of network congestion levels.

VI. CONCLUSION

This paper has investigated the effectiveness of the combination of locality awareness and FMO error resilience technique over P2P. It is found that Flexible Macroblock Ordering in H.264/AVC is a very promising form of error resilience when packet loss rates are relatively high. Moreover, the technique is compatible with existing coding standards and does not require action at intermediate nodes other than to forward the video-bearing packets.

However, in order to exploit the best of error resilience techniques, we still need to make the overlay aware of the physical network which will help in maintaining the intercommunication among the nearby peers. This will give more chance to get benefit of the available error correction methods.

REFERENCES

[1] G. Marfia, G. Pau, P. Di Rico, and M. Gerla, "P2P Streaming Systems: A Survey and Experiments," *ST Journal of Research,* pp. 1-4, 2007.

[2] Zattoo. *Zattoo Home Page*. Available: www.zattoo.com, (Access on July, 2010)

[3] Joost. *Joost Home Page*. Available: http://www.joost.com/, (Access on July, 2010)

[4] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications,* vol. 1, pp. 18-28, 2008.

[5] H. Chang, S. Jamin, and W. Wang, "Live streaming performance of the Zattoo network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, Chicago, Illinois, USA, 2009, pp. 417-429.

[6] M. Alhaisoni, A. Liotta, and M. Ghanbari, "Improving P2P Streaming Methods for IPTV," *International Journal on Advances in Intelligent Systems Volume 2, Numbers 2&3, 2009,* 2009.

[7] B. Zhao, A. Joseph, and J. Kubiatowicz, "Locality-aware mechanisms for large-scale networks," in *in Proc. Workshop Future Directions Distrib. Comput. (FuDiCo)*, Italy, 2002, pp. 80–83.

[8] B. Katz, S. Greenberg, N. Yarkon, N. Blaunstein, and R. Giladi, "New error-resilient scheme based on FMO and dynamic redundant slices allocation for wireless video transmission," *IEEE Trans. on Broadcasting,* vol. 53, pp. 308-319, . 2007.

[9] M. Alhaisoni and A. Liotta, "Characterization of signaling and traffic in Joost," *Peer-to-Peer Networking and Applications,* vol. 2, pp. 75-83, 2009.

[10] Kazaa. *Home Page Kazaa,* . Available: www.kazaa.com

[11] Y. Shen, Z. Liu, S. Panwar, K. Ross, and Y. Wang, "Streaming layered encoded video using peers," in *IEEE International Conference on Multimedia and Expo*, Amsterdam, The Netherlands, 2005, p. 4.

[12] P. Lambert, W. De Neve, Y. Dhondt, and R. Van de Walle, "Flexible macroblock ordering in H. 264/AVC," *Journal of Visual Communication and Image Representation,* vol. 17, pp. 358-375, 2006.

[13] M. Altaf, M. Fleury, and M. Ghanbari, "H.264 Error Resilience Performance for Wireless Video " in *MobiMedia*, London, UK, September 7–9, 2009.

[14] Ns-2. Available: http://isi.edu/nsnam/ns, (Access on July, 2010)

[15] F. Agboma, M. Smy, and A. Liotta, "QoE Analysis of a Peer-to-Peer Television System," in *IADISInt. Conf. on Telecommunications, Networks and Systems*, Amsterdam, The Netherlands, 2008, pp. 365-382.

[16] T. Locher, S. Schmid, and R. Wattenhofer, "equus: A provably robust and locality-aware peer-to-peer system," in *the Sixth IEEE International Conference on Peer-to-Peer Computing*, Cambridge, United Kingdom, 2006, pp. 3-11.

[17] S. Asaduzzaman, Y. Qiao, and G. Bochmann, "CliqueStream: an efficient and fault-resilient live streaming network on a clustered peer-to-peer overlay," in *the Eighth IEEE International Conference on Peer-to-Peer Computing*, Aachen, Germany, 2008, pp. 269-278.

[18] P. RODRÍGUEZ-BOCCA, "Quality-centric design of Peer-to-Peer systems for live-video broadcasting," Ph.D Thesis, Universit´e de Rennes, 2008.

[19] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *In Proc. SPIE, Multimedia Computing and Networking*, December 2001., pp. 186–195.

[20] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," presented at the Proceedings of the eleventh ACM international conference on Multimedia, Berkeley, CA, USA, 2003.

[21] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang, "Location-aware topology matching in P2Psystems," in *IEEE Infocomm*, HongKong, 2004, pp. 1-11.

[22] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *23rd International Conference on Distributed Computing Systems, (ICDCS)*, RI, USA, 2003, pp. 500- 508.

[23] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *Computer Communications Review,* vol. 32, pp. 205-217, 2002.

[24] V. Padmanabhan, H. Wang, and P. Chou, "Resilient peer-to-peer streaming," in *IEEE ICNP*, Atlanta, GA, USA, 2003, pp. 16–27.

[25] N. Aboobaker, D. Chanady, M. Gerla, and M. Sanadidi, "Streaming media congestion control using bandwidth estimation," *Management of Multimedia on the Internet,* pp. 89-100.

[26] J. Kim, R. M. Mersereau, and Y. Altunbasak, "Network-adaptive video streaming using multiple description coding and path diversity," in *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 1*, 2003, pp. 653-656.

[27] M. Mushtaq, T. Ahmed, and D.-E. Meddour, "Adaptive packet video streaming over P2P networks," in *Proceedings of the 1st international conference on Scalable information systems*, Hong Kong, 2006, pp. 423-428.

Effectiveness of Landmark Analysis for Establishing Locality in P2P Networks

Alexander Allan and Giuseppe Di Fatta
School of Systems Engineering
The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
{siu07aja, G.DiFatta}@reading.ac.uk

***Abstract*—Locality to other nodes on a peer-to-peer overlay network can be established by means of a set of landmarks shared among the participating nodes. Each node independently collects a set of latency measures to landmark nodes, which are used as a multi-dimensional feature vector. Each peer node uses the feature vector to generate a unique scalar index which is correlated to its topological locality. A popular dimensionality reduction technique is the space filling Hilbert's curve, as it possesses good locality preserving properties. However, there exists little comparison between Hilbert's curve and other techniques for dimensionality reduction. This work carries out a quantitative analysis of their properties. Linear and non-linear techniques for scaling the landmark vectors to a single dimension are investigated. Hilbert's curve, Sammon's mapping and Principal Component Analysis have been used to generate a 1d space with locality preserving properties. This work provides empirical evidence to support the use of Hilbert's curve in the context of locality preservation when generating peer identifiers by means of landmark vector analysis. A comparative analysis is carried out with an artificial 2d network model and with a realistic network topology model with a typical power-law distribution of node connectivity in the Internet. Nearest neighbour analysis confirms Hilbert's curve to be very effective in both artificial and realistic network topologies. Nevertheless, the results in the realistic network model show that there is scope for improvements and better techniques to preserve locality information are required.**

***Index Terms*—Peer-to-Peer Networks; Landmark Clustering; Hilbert's Curve; Principal Component Analysis; Sammon's Mapping**

I. INTRODUCTION

In Peer-to-Peer (P2P) networks it can be advantageous to be aware of the geographical heterogeneity between nodes as a means of optimising load balancing, routing and search efficiency [1], [2], [3], [4]. These benefits are derived from exploiting the fact that nodes in close proximity enjoy lower communication latency.

In order to reap these rewards the P2P network needs to embed some measure of the topological distribution and the locality of its constituent nodes. Obtaining this information can be problematic as nodes do not have complete knowledge of the network from which to calculate a neighbourhood.

Landmark clustering [2] has been widely used to generate proximity information. If nodes are physically close to each other, they are also likely to experience similar latency in the communication path to selected landmark nodes.

A set of landmarks allows generating a multi-dimensional space, where each node is represented by a landmark vector, i.e. a vector of the typical communication latency to the landmark nodes. Nearby nodes in the network topology are expected to be represented by similar landmark vectors. Landmark cluster analysis allows identifying and quantifying node proximity without the detailed and global knowledge of the network topology.

Landmark spaces are typically high dimensional and techniques to map them to a 1-d space, like Distributed Hash Table (DHT) identifier spaces, have been studied [5], [6]. The general approach first calculates locality at a node via a latency vector to predefined landmark nodes throughout the network. The vector is then reduced to a 1-d index space by means of a dimensionality reduction technique employing a space filling curve known as Hilbert's curve.

It is known that Hilbert's curve possesses good locality preserving properties compared with other space filling curves [7], [8]. Among others, authors in [9] and [10] have studied Hilbert's curve and its locality preserving properties.

Yet there exists little comparison between Hilbert's curve and other general techniques for dimensionality reduction regardless of their practical applicability in the context of large-scale distributed systems.

Dimensionality reduction is a projection from a D-dimensional space onto an K-dimensional one, for $D > K$. A large number of dimensionality reduction techniques have been proposed in the litterature with different characteristics, properties and aims [11], [12]. Typically these techniques are used as pre-processing step in order to cope with the curse of dimensionality before an appropriate learning algorithm is applied to the data (typically $K \ll D$). In other applications, dimensionality reduction aims at the visualisation of multi-dimensional data ($K = 2$).

In the context of the landmark and identifier spaces in P2P systems described above, D is the number of landmark nodes and K is 1.

In this work the effectiveness of Hilbert's curve is compared with two other methods for dimensionality reduction: Principal Component Analysis (PCA) [13], [14], [15] and Sammon's mapping [16].

Principal component analysis is one of the most popular and widely used linear dimensionality reduction methods and provides the optimum projection in terms of the mean-square error.

Sammon's mapping is a visualisation technique which performs a dimensionality reduction and is based on a non-linear approach.

These two techniques were chosen as they are well known examples of linear and non-linear dimensionality reduction. However, as they both require global knowledge of the data space, it would be impractical to implement them in large-scale distributed environments, like P2P systems. They rather serve as a benchmark with which to assess the quality of result produced by Hilbert's curve.

The experimental analysis is based on two network topology models. In the first topology nodes are placed on a 2d plane to generate an artificial distribution and to emphasize locality. This is used as proof of concept.

A second more rigorous simulation is based on a realistic network topology model with a typical power-law distribution of node connectivity in the Internet.

The overall goal of this paper is to provide an argument in support of the use of Hilbert's curve as a dimensionality reduction method via benchmark comparison with two other widely used techniques. An additional goal is to provide a quantitative evaluation of the locality information which is preserved after landmark vector analysis. This should lead to a better understanding of the benefits that can be expected from such a technique and of the margin for improvement.

The rest of the paper is organised as follows. Section II provides an overview of three techniques adopted to convert the landmark vectors into a locality-aware 1-d identifier. Section III describes the methodology adopted for the comparative analysis of the three methods. Sections IV and V provide the experimental results and their interpretation. Conclusive remarks are given in section VI.

II. Overview of Techniques

A. Hilbert's Curve

Hilbert's curve is a continuous fractal space-filling curve of finite granularity. Giuseppe Peano (1858-1932) discovered a densely self intersecting curve in 1890 which passes through every point in a 2-d space (and by extension in an n-dimensional hypercube) [17], [18]. This work was followed in 1891 by that of David Hilbert [19] who published his own version of the space filling curve including illustrations for construction (Figure 1). Hilbert's variant proves to have performance advantages (in terms of how well 'compact regions' of 2-d space are represented) over other space filling curves which explains its attraction as a contemporary multi-dimensional indexing method [20] [21].

The Hilbert's variant proceeds through each step replacing the U shape with an upside down Y. Each corner in the diagram represents an additional number in the sequence. As a mean of dimensionality reduction, it transforms the data from n to 1 dimension by assigning each point in space a number.

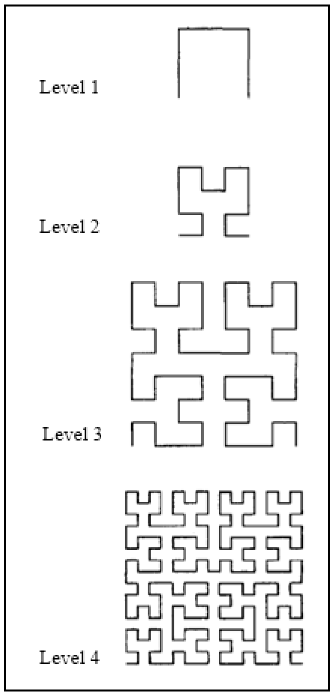


Fig. 1. The first 4 levels of Hilbert's curve in 2 dimensions

B. Principal Component Analysis

Principal Component Analysis (PCA) was introduced by Karl Pearson in 1901 [22], [15]. It employs the Karhunen-Loéve theorem which is similar to a Fourier series and transforms potentially correlated variables into a lesser number of uncorrelated variables known as principal components. PCA in essence aims to cast a projection of the higher dimensional data onto the low dimensional space from its most 'informative' angle retaining as much variability as possible.

The initial principal component chosen attempts to capture as large a range of variability within the data as is possible, with the next principal component chosen to maximize the remaining variability and so forth until all principal components are identified.

C. Sammon's Mapping

Developed by J. W. Sammon Jr in 1969 [16], Sammon's mapping is a non-linear form of dimensionality reduction based on gradient search which attempts to keep as much of the structure of the original measurement of dimensions as possible. Each iteration attempts to minimize an error function known as Sammon's stress, while matching the pairwise distances in the high-dimensional space to those in the lower-dimensional one.

PCA and Sammon's mapping have been shown to be among the best methods for dimensionality reduction in terms of preserving cluster validity [23] for data visualisation ($K = 2$).

III. Comparative Analysis

This work provides a comparative analysis of the three techniques described above for reducing dimensionality to generate a locality aware index ($K = 1$) for the nodes of P2P overlay networks. A randomly generated index for control and an ideal index obtained from global network knowledge are included in the comparison for reference.

A. *Simulation 1: landmark analysis on a 2d plane*

In the first experiment an artificial network topology is considered as proof of concept. 1000 points were arranged on a 2d Euclidean plane following a rectangular perimeter to represent nodes on a network (Figure 2). Six landmarks are randomly selected, based on preliminary work suggesting that this number of landmarks produces the best accuracy within a range restricted by computational resources available. The Euclidean distance from a node to a landmark is used to simulate network latency and provides a 6d vector for every node. This vector was reduced to the 1d node index using all three methods of dimensionality reduction. The Hilbert number $H(n)$ was computed with a recursion free version [24][25] of the Hilbert's curve algorithm. Node indices $P(n)$ and $S(n)$ were calculated by means of the implementations of the algorithms, respectively, PCA and Sammon's mapping, available in the data mining development environment *KNIME* [26]. As a control, a random index R(n) was created in which there was no relation between index values and location.

For each network node n, the 10 nearest neighbours ($nn_1, ..., nn_{10}$) were found by searching the 10 closest indices on the 1d space defined by, respectively, node indices $H()$, $S()$, $R()$ and $P()$. The Euclidean distances were determined from node n to its nearest neighbours ($nn_1, ..., nn_{10}$) on the original 2d plane. The sum (N) of these distances is computed for each node to create distance arrays $N_H[]$, $N_R[]$, $N_S[]$ and $N_P[]$. The value of N was also found for the nearest neighbours of every point on the original 2d plane to produce an ideal neighbour value array $N_I[]$. These 5 arrays were then compared with each other to assess the degrees of locality preservation.

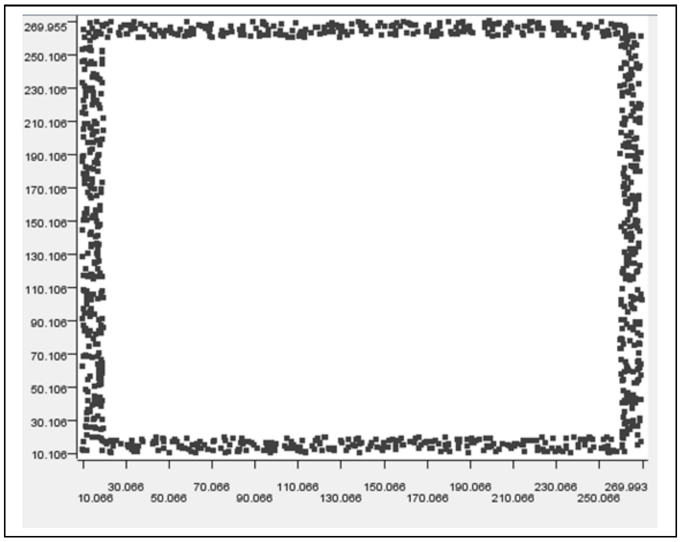


Fig. 2. Layout of the artificial network topology with 1000 nodes (2d plane)

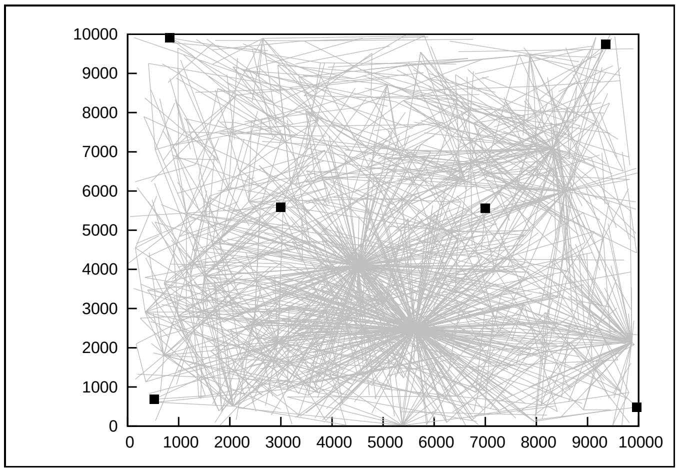


Fig. 3. A 2d representation of the realistic network topology (Inet) with 3037 nodes. (Landmarks are marked)

B. *Simulation 2: landmark analysis in weighted graph*

The second experiment applies the same methodology to a more realistic network topology generated with Inet [27], an Autonomous System level Internet topology generator. Inet generates random networks with characteristics similar to those of the Internet in terms of the power-law distribution of node connectivity. It can approximate Internet-like network topologies to a high degree of accuracy [28] with respect to the degree distribution and the minimum vertex cover size. Inet was used to create a 3037 node graph with 4788 edges whose weights corresponds to latency values. 6 nodes of this graph were chosen as landmarks (Figure 3).

The shortest path between pairs of nodes was computed with the Dijkstra algorithm to determine their communication latency. For each network node a 6d vector was generated with the communication latency to the landmarks. From this vector the indices H, P and S were computed and the random control R was generated. The 5 nearest neighbours for every node in each 1d space defined by the indices H, P, S and R were found and the sum of distance to these neighbours via Dijkstra shortest path was calculated. Similarly to the previous case, the latency arrays $N_H[]$, $N_R[]$, $N_S[]$ and $N_P[]$ were computed. The actual 5 nearest neighbours of every node in the network were found using a Dijkstra algorithm with expanding search radius. The total sum of the path length from each node n to its 5 nearest neighbours ($nn_1, ..., nn_5$) was computed to produce an ideal sum of latencies array $N_I[]$.

IV. Experimental Results

The landmark-vector analysis on the artificial 2d plane topology showed that Hilbert's curve is the most effective method of preserving locality information among the three dimensionality reduction techniques. Table I and Figure 4 show the result for this case. Points indicated as adjacent by the Hilbert index were on average over twice (2.13) as far as the best possible as given by the ideal case but over 19 times closer than if they had been chosen at random. Sammon's mapping came second scoring four times worse than Hilbert's

TABLE I
AVERAGE DISTANCE TO THE 10 NEAREST NEIGHBOURS FOR THE ARTIFICIAL NETWORK TOPOLOGY (2D PLANE)

Method (6 landmarks)	Mean distance to 10 nearest neighbours
Ideal	44.52
Hilbert	94.72
Sammon	404.98
PCA	883.35
Random	1810.78

TABLE II
AVERAGE LATENCY TO THE 5 NEAREST NEIGHBOURS FOR THE REALISTIC NETWORK TOPOLOGY (INET)

Method (6 landmarks)	Mean latency to 5 nearest neighbours
Ideal	269.07
Hilbert	600.54
Sammon	714.34
PCA	744.80
Random	751.18

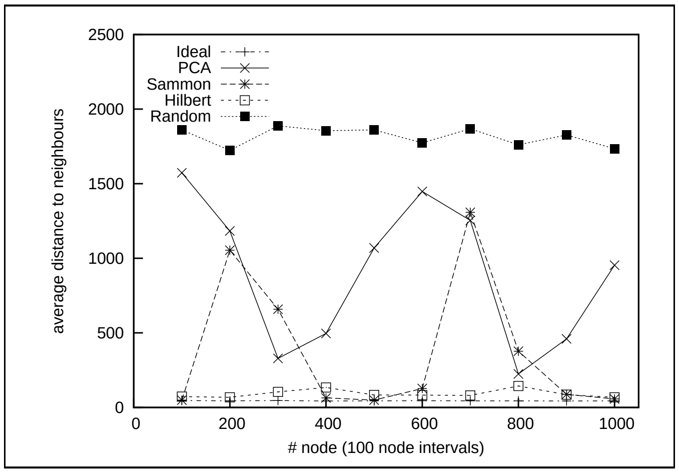


Fig. 4. Neighbour distance (moving average) for the artificial network topology model (2d plane)

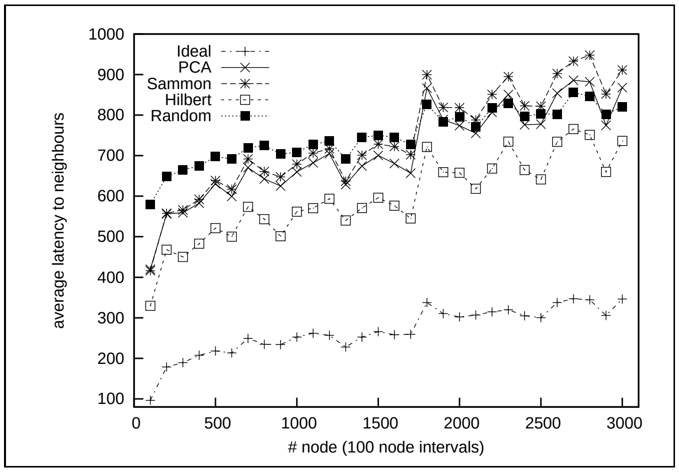


Fig. 5. Neighbour latency (moving average) for the realistic network topology (Inet). Five nearest neighbours are considered for each node

curve, but four times better than random points. PCA scored nine times worse than Hilbert's curve, but still over twice as well as random.

Table II and Figure 5 show a comparison of the different techniques for the more realistic hierarchical topology. The Hilbert's curve confirmed to be the most effective technique with an average latency of just over twice the ideal possible value (2.23 times higher) but with much less improvement over random (1.25 times lower). The PCA approach came second with just 1.05 times smaller latency to neighbours than to neighbours chosen randomly. Sammon's mapping performed the worst showing barely any improvement over random (only 1.008 times smaller on average).

In the artificial 2d plane topology an ideal nearest neighbour distance is on average 40.67 times less than to an index based on randomly chosen neighbours. In this case Hilbert's curve is very close to the ideal index. In the Internet-like topology the ratio between ideal and random indices dropped to 2.79. However, in this more realistic case the performance of Hilbert's curve is far from ideal.

V. DISCUSSION

The results on the 2d plane network model showed good performance for the landmark analysis technique as might be expected when Euclidean distance can be taken to landmarks (as this is the principle of GPS navigation systems). The Hilbert's curve produced results that were much closer to the ideal values than to the random control values. The other two techniques also performed much better than random on this test, but were less effective than Hilbert's curve.

In order to generate a homogeneous Hilbert index across independent nodes, all that is required at each node is the maximum possible size of all input dimensions and a curve width. These can be predetermined constants. For Sammon's mapping, the entire data set would need to be recalculated every time a new node joined the network as the maps produced lack generalisation. Similarly, PCA relies on knowing the entire data set to establish the covariance matrix which is central to the algorithm. For this reason, Sammon's mapping and PCA are wholly unsuitable for distributed geographical indexing purposes. In this test they really serve as a benchmark with which to assess the effectiveness of the Hilbert's curve. These methods are better suited to feature preservation than locality preservation.

In general, the results were not unexpected considering Hilbert indexing is known as one of the best methods of preserving locality, but the degree to which it outperforms other dimensional scaling methods was surprising.

The results from the more realistic topology model indicate that landmark vector analysis via Hilbert's curve can give nodes an awareness of locality. The results also show that Sammon's mapping and PCA are not likely to produce useful results, performing only marginally better than points chosen at random. Considering the chart in Figure 5, the PCA and Sammon's mapping may actually perform worse than random for some nodes (1800 to 3000).

What these results are not able to show for certain is whether the relatively poor performance on the Internet-like network topology when compared to the 2d plane topology is due to the loss of landmark quality when a path is found through a network or whether this degradation is due to the dimensionality reduction. However given that the performance on the 2d plane, it seems likely that a real network represents a more complicated space through which to derive locality so performance is not likely to approach that of a more theoretical realm.

Whether the 25% improvement over random achieved by the Hilbert reduction of landmark vector analysis represents enough locality information to be useful is subject to the individual requirements of the desired application. These results give at least some indication as to the quality of accuracy likely to be achieved by using this, or similar techniques across a real network.

VI. Conclusion

This work has presented a comparative analysis of multi-dimensional scaling techniques for establishing node locality in P2P networks. A set of shared landmarks can be adopted by each node to incorporate locality information in its peer identifier. The experimental analysis on both an artificial topology model and on a hierarchical topology based on a realistic Internet model, has shown the effectiveness of the Hilbert's curve with respect to Principal Component Analysis and Sammon's mapping. Nevertheless, the analysis has also identified scope for improvements; locality preserving techniques based on Hilbert's curve and landmark clustering are far from ideal. The space filling curve used in the H-indexing scheme proposed by Niedermeier et al. [10] purports to outperform Hilbert's curves in terms of locality preservation. These curves shall be incorporated and evaluated in future implementation. Current research efforts are focusing on the optimisation of the simulation code in order to extend the analysis to larger networks, varying number of landmark nodes and neighbours. These factors are currently restricted by computational resources. To provide more realistic results still, an implementation on PlanetLab P2P overlay test bed [29] is planned. A further interesting research direction is the adoption of techniques to cope with missing latency measurements to some landmarks and to introduce robustness to variability of the set of landmarks over the network nodes.

References

[1] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Microsoft Research, Cambridge, England, Tech. Rep. MSR-TR-2002-82, May 2002.

[2] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings*, New York, USA, Jun. 2002, pp. 1190–1199.

[3] Y. Zhu and Y. Hu, "Towards efficient load balancing in structured p2p systems," in *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, Santa Fe, USA, Apr. 2004, p. 20a.

[4] H. Shen and C. Xu, "Hash-based proximity clustering for load balancing in heterogeneous dht networks," *Journal of Parallel and Distributed Computing*, vol. 65, no. 5, pp. 686–702, May 2005.

[5] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, Brown University, USA, May 2003, pp. 500–508.

[6] Z. Xu, M. Mahalingam, and M. Karlsson, "Turning heterogenity into an advantage in overlay routing," in *Proceedings of IEEE INFOCOM*, San Francisco, USA, Mar. 2003, pp. 1499 – 1509.

[7] C. Gotsman and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 794–797, Jan. 1996.

[8] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz, "Analysis of the clustering properties of the hilbert space-filling curve," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 124–141, Jan. 2001.

[9] J. Alber and R. Niedermeier, "On multi-dimensional hilbert indexings," in *Computing and Combinatorics: 4th Annual International Conference, Proceedings*, Taipei, Taiwan, Aug. 1998, pp. 329–338.

[10] R. Niedermeier, K. Reinhardt, and P. Sanders, "Towards optimal locality in mesh-indexings," in *Proceedings of the 11th International Symposium on Fundamentals of Computation Theory*, Krakow, Poland, Sep. 1997, pp. 364 – 375.

[11] M. A. Carreira-Perpinan, "A review of dimension reduction techniques," Department of Computer Science, University of Sheffield, U.K., Tech. Rep. CS-96-09, 1997.

[12] I. K. Fodor, "A survey of dimension reduction techniques," Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, Tech. Rep. UCRL-ID-148494, 2002.

[13] P. Diaconis and M. Shahshahani, "On nonlinear functions of linear combinations," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, pp. 175–191, Mar. 1984.

[14] G. Eslava and F. H. C. Marriott, "Some criteria for projection pursuit," *Statistics and Computing*, vol. 4, no. 1, pp. 13–20, Mar. 1994.

[15] I. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. New York, USA: Springer, 2002, no. XXIX.

[16] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, May 1969.

[17] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Mathematische Annalen*, vol. 36, no. 1, pp. 157–460, 1890.

[18] A. R. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, pp. 314–330, 1968.

[19] D. Hilbert, "Ueber die stetige abbildung einer line auf ein flchenstck," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891.

[20] C. Faloutsos and Y. Rong, "Spatial access methods using fractals: Algorithms and performance evaluation," University of Maryland, Maryland, USA, Tech. Rep. UMIACS-TR-89-31, Mar. 1989.

[21] H. Jagadish, "Linear clustering of objects with multiple attributes," *ACM SIGMOD Record*, vol. 19, no. 2, pp. 332–342, Jun. 1990.

[22] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, Jun. 1901.

[23] D. Marghescu, "Evaluating the effectiveness of projection techniques in visual data mining," in *Proceedings of the Sixth IASTED International Conference on Visualization, Imaging, And Image Processing*, Palma de Mallorca, Spain, Aug. 2006, pp. 186–193.

[24] A. Butz, "Alternative algorithm for hilbert's space-filling curve," *IEEE Transactions on Computers*, vol. 20, no. 4, pp. 424–426, Apr. 1971.

[25] D. Moore. Fast hilbert curve generation and sorting and range queries. Rice University. Texas, USA. [Online]. Available: http://www.tiac.net/∼sw/2008/10/Hilbert/moore/hilbert.c (last accessed: June 2010)

[26] M. Berthold, N. Cebron, F. Dill, G. Di Fatta, T. Gabriel, F. Georg, T. Meinl, P. Ohl, C. Sieb, and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *Proceedings of the Workshop on Multi-Agent Systems and Simulation MAS&S, 4th Annual Industrial Simulation Conference (ISC)*, Palermo, Italy, Jun. 2006, pp. 58–61.

[27] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," University of Michigan, Michigan, USA, Tech. Rep. CSE-TR-456-02, Jun. 2002.

[28] G. Di Fatta, G. L. Presti, and G. L. Re, "Computer network topologies: Models and generation tools," Consiglio Nazionale delle Ricerche, Palermo, Italy, Tech. Rep. CERE-CNR, 5/2001, Jul. 2001.

[29] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *IEEE Transactions on Computers*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

Aggregation Skip Graph: A Skip Graph Extension for Efficient Aggregation Query

Kota Abe, Toshiyuki Abe, Tatsuya Ueda, Hayato Ishibashi and Toshio Matsuura
Graduate School for Creative Cities, Osaka City University
Osaka, Japan
Email: {k-abe,t-abe,tueda,ishibashi,matsuura}@sousei.gscc.osaka-cu.ac.jp

***Abstract*—Skip graphs are a structured overlay network that allows range queries. In this article, we propose a skip graph extension called *aggregation skip graphs*, which efficiently execute aggregation queries over peer-to-peer network. An aggregation query is a query to compute an aggregate, such as MAX, MIN, SUM, or AVERAGE, of values on multiple nodes. While aggregation queries can be implemented over range queries of conventional skip graphs, it is not practical when the query range contains numerous nodes because it requires the number of messages in proportion to the number of nodes within the query range. In aggregation skip graphs, the number of messages is reduced to logarithmic order. Furthermore, computing MAX or MIN can be executed with fewer messages as the query range becomes wider. In aggregation skip graphs, aggregation queries are executed by using periodically collected partial aggregates for local ranges of each node. We have confirmed the efficiency of the aggregation skip graph by simulations.**

***Keywords*—aggregation; skip graphs; peer-to-peer**

I. INTRODUCTION

P2P (Peer-to-Peer) systems have attracted considerable attention as technology for performing distributed processing on massive amounts of information using multiple nodes (computers) connected via a network. In P2P systems, each node works autonomously cooperating with other nodes that constitute a system that can be scaled by increasing the number of nodes.

Generally, P2P systems can be grouped into two major categories: unstructured and structured P2P systems. Structured P2P systems can look up data in logarithmic order of the number of nodes, by restricting the topology of network.

Regarding structured P2P systems, DHT (Distributed Hash Table)-based systems, such as Chord [2], Pastry [3], and Tapestry [4], have been extensively researched. DHTs are a class of decentralized systems that can efficiently store and search for key and value pairs. DHTs also excel at load distribution. However, DHTs hash keys to determine the node that will store the data, and hence a value cannot be searched for if the correct value of the key is not known. Therefore, it is difficult with DHT to search for nodes whose key is within a specified range (range query).

As a structured P2P system which supports range queries, the skip graph [5] has attracted considerable attention. A skip graph is a distributed data structure that is constructed from multiple skip lists [6] that have keys in ascending order. The skip graph is suitable for managing distributed resources where the order of the keys is important.

Aggregation queries can be considered a subclass of range queries. An aggregation query is a query to compute an aggregate, such as the MAX, MIN, SUM, or AVERAGE, from the values that are stored in multiple nodes within a specified range. Aggregation queries are useful and sometimes essential for P2P database systems. Aggregation queries have a wide variety of applications. For example, across a range of nodes in a distributed computing system such as a grid, an aggregation query can be used to obtain the average CPU load, the node with the maximum CPU load, or the total amount of available disk space. An aggregation query can also be used to compute the average or maximum value from sensor data within a specified range on a sensor network. Other possible usage of aggregation queries can be found in [7][8].

While aggregation queries can be implemented by using range query over skip graphs, this is not efficient because every node in the specified range must process the aggregation query message; thus, this method stresses network bandwidth and CPU especially when aggregation queries having a wide range are frequently issued.

In this paper, we propose the **aggregation skip graph**, a skip graph extension that efficiently execute aggregation queries. In the aggregation skip graph, the expected number of messages and hops for a single aggregation query is $O(\log n + \log r)$, where n denotes the number of nodes and r denotes the number of nodes within the query range. Furthermore, computing MAX or MIN can be executed with fewer messages as the query range becomes wider.

We discuss related work in Section II and present the aggregation skip graph algorithm in Section III. In Section IV, we evaluate and discuss the aggregation skip graph. Lastly, in Section V, we summarize this work and discuss future work.

II. RELATED WORK

A. Aggregation in P2P Network

Some research has focused on computing aggregations on P2P networks, to name a few, in the literature [7]–[11].

Most of the existing methods construct a reduction tree for executing aggregation queries, as summarized in [10]. However, this approach incurs a cost and complexity because constructing a reduction tree over a P2P network is equal to adding another overlay network layer over an overlay network.

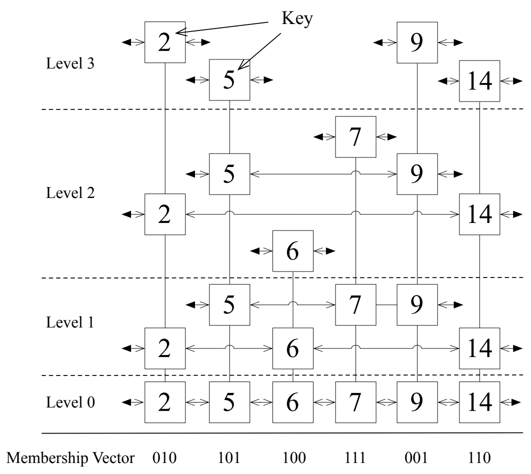


Fig. 1. Example of skip graphs

In addition, to our knowledge, none of the existing methods support computing aggregations in a subset of nodes; they compute aggregates only on all nodes.

As we discuss later, the aggregation skip graph does not require constructing a reduction tree, nor even maintaining additional links to remote nodes; aggregation queries are executed utilizing the data structure of underlying skip graphs. Furthermore, it can compute aggregates within a subset of nodes, by specifying a key range.

B. Skip Graph and Skip Tree Graph

A skip graph [5] is a type of structured overlay network. The skip graph structure is shown in Fig. 1. The squares in the figure represent nodes, and the number within each square is the key. Each node has a membership vector, which is a uniform random number in base w integer. Here, we assume $w = 2$.

Skip graphs consist of multiple levels, and level i contains 2^i doubly linked lists. At level 0, all of the nodes belong to only one linked list. At level $i(> 0)$, the nodes for which the low-order i digit of the membership vector matches belong to the same linked list. In the linked list, the nodes are connected by the key in ascending order. We assume that the leftmost node in the linked list and the rightmost node are connected (i.e., circular doubly-linked list). To maintain the linked lists, each node has pointers (IP address, etc.) to the left and right nodes at each level.

In a skip graph, when the level increases by 1, the average number of nodes for one linked list decreases by $1/2$. We refer to the level at which the number of nodes in the linked list becomes 1 as the *maxLevel*. The maxLevel corresponds to the height of the skip graph. In the skip graph for n nodes, the average maxLevel is $O(\log n)$, and the number of hops required for node insertion, deletion and search is also $O(\log n)$.

With skip graphs, aggregation queries can be easily implemented over range queries, in which one is asked all keys in [x, y]. Range queries require all nodes within the range receive a message. If we denote the number of nodes within the target range of the aggregation query by r, then range queries requires $O(\log n + r)$ messages and hops on average.

The skip tree graph [12] is a variant of skip graph, which allows fast aggregation queries by introducing additional pointers called *conjugated* nodes at each level. Skip tree graphs run range queries in $O(\log n + r)$ messages and $O(\log n + \log r)$ hops.

In either skip graphs or skip tree graphs, the number of messages for range queries increases in proportion to r; thus, these methods are not practical for aggregation queries, especially when aggregation queries with a wide range are frequently issued.

III. Proposed Method

In this section, we describe the detail of the aggregation skip graph.

In the following, we focus on aggregation queries to find the largest value in a specified range (i.e., MAX). However, it is trivial to change the algorithm to the MIN. We discuss other aggregates (such as AVERAGE, SUM, etc.) in Section IV-D.

A. Data Structure

In aggregation skip graphs, each node stores a key–value pair. In the same manner as conventional skip graphs, the linked lists at each level are sorted by key in ascending order. The *value* is not necessarily related to the order of the key, and may change, for example, as in the case of sensor data.

The data stored in each node of an aggregation skip graph are shown in Table I. The key, membership vector, left[] (pointer to the left node at each level), right[] (pointer to the right node at each level), and maxLevel are the same as in conventional skip graphs. Hereinafter, we use the notation P.key to denote the key of node P. Also, we use the notation "node x" to denote the node whose key is x.

In addition to the skip graph, each node of a aggregation skip graph stores agval[] and keys[]. The *value* of the node is stored in agval[0]. P.agval[i] $(0 < i)$ stores the MAX value within the nodes between P (inclusive) to P.right[i] (exclusive) in the linked list at level 0, where P.right[i] denotes the right node of node P at level i. P.keys[i] $(0 < i)$ stores the key set that corresponds to the P.agval[i]. (A set is used because multiple keys may share the same MAX value.) P.keys[0] is not used. We describe the method to collect agval[] and keys[] later in Section III-C.

In skip graphs, the pointers for the left and right nodes point to more distant nodes as the level increases. Therefore, as the level increases, agval[] stores MAX values in a wider range, and agval[maxLevel] stores the MAX value for all nodes.

Fig. 2 shows an example of an aggregation skip graph. The squares in level 0 show the *value* (agval[0]) of a node, and $\text{agval}[i]_{/\text{keys}[i]}$ in level i $(0 < i)$.

Let us consider, as an example, the square at level 2 of node 5. The square contains $5_{/6}$ because the MAX value in

TABLE I
DATA MANAGED BY EACH NODE

Variable	Description
key	Key
m	Membership vector
right[]	Array of pointers to right node
left[]	Array of pointers to left node
maxLevel	First level at which the node is the only node in the linked list
agval[]	Array of collected MAX values for each level (agval[0] is the *value* of the node)
keys[]	Array of key sets that correspond to agval[]

the nodes between node 5 (inclusive) and node 9 (exclusive), which is the right node of node 5 at level 2, is 5 and the corresponding key is 6. Note that all of the nodes contain $9_{/2}$ at the highest level because the MAX value for all nodes is 9 and the corresponding key is 2.

In an aggregation skip graph, insertion and deletion of nodes can be accomplished by using the conventional skip graph algorithm; thus, this is not discussed here.

B. Query Algorithm

Here, we describe the algorithm for aggregation queries.

The algorithm gives the MAX value (and the corresponding key set) within all values stored by nodes whose key is in the specified key range $r = [r.\min, r.\max]$.

1) Algorithm overview: Here, for simplicity, we provide a brief overview of the algorithm; please refer to Section III-B2 for further details.

An aggregation query proceeds, starting from a node on the left side of the specified range (having a smaller key), to a node on the right side of the range.

Let us consider the case where node P issues an aggregation query for range r. If P is within r, P forwards the query message to node Q, where Q is a node known to P, which is outside of range r and the closest to $r.\min$. (If P is outside range r, then read the Q below as P instead.)

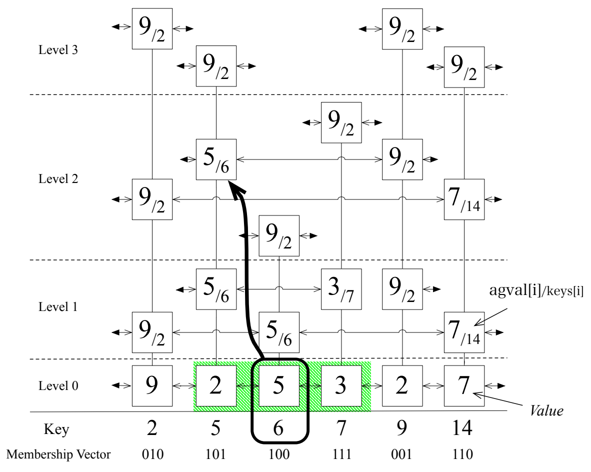


Fig. 2. Example of aggregation skip graph

Let i denote the current level, starting from $\text{maxLevel} - 1$. Let s denote the range from Q to Q.right[i], and x the MAX value in s. Node Q executes one of the following steps, determined by the relation between range r, range s and x. This is depicted in Fig. 3 (1)–(4). In the figure, the arrow represents the pointer from Q to Q.right[i].

(1) **Range s includes range r, and the key of x is within r.**
It is clear that x is also the MAX value for r; thus, x is returned to P as the MAX value and the query terminates.

(2) **Range s has a common area with range r, and the key of x is within r.**
The value x is the MAX value for the common area between range s and range r. However, because an even larger value may exist in the remaining range r, the query is forwarded to Q.right[i]. The value of x and the corresponding key are included in the forwarded message.

(3) **Range s has a common area with range r, but the key of x is not within r**
In this case, no information is obtained about the MAX value within range r; thus, the current level (i) is decreased by 1 and this process is repeated again from the beginning.

(4) **Range s and range r do not have any common areas**
The MAX value in range r does not exist in range s; thus, the query is sent to Q.right[i].
In this case, Q.right[i] acts as the new Q and the process is repeated again in the same manner.

Next, we describe the algorithm for a node that receives a forwarded query message from node Q in case (2). We denote such a node by R. Again, let i denote the current level, starting from $\text{maxLevel} - 1$. Also, let t denote the range from R to R.right[i], and let y denote the MAX value in t. Node R executes one of the following steps. This is depicted in Fig. 3 (5)–(7).

(5) **Range t has a common area with range r, and the key of y is within r**
The $\max(x, y)$ is returned to P and the query terminates.

(6) **Range t has a common area with range r, but the key of y is not within in r**
If $x > y$, x is returned to P as the MAX value and the query terminates. Otherwise, decrease the current level (i) by 1 and repeat this process.

(7) **Range t is included in range r**
The query is forwarded to R.right[i]. The $\max(x, y)$ and the corresponding key are included in the forwarded message.

2) Algorithm details: Here, we present the detailed algorithm in pseudocode.

Node P initiates an aggregation query by calling `agQuery(r, d, P,` $-\infty$`,` $\emptyset$`)`. The parameter d indicates

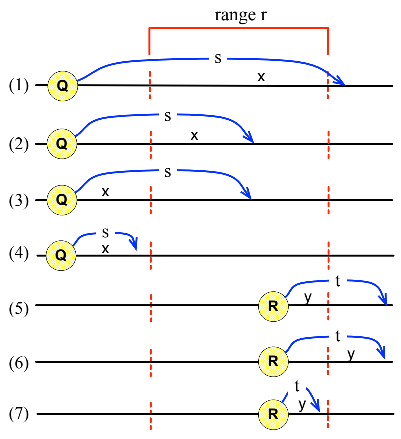


Fig. 3. Relationship between node and range

the direction of the query. The initial value of d is, LEFT if P.key is included in range r, otherwise RIGHT.

The notation $a \prec b \prec c$ means $(a < b < c \ \vee \ b < c < a \vee \ c < a < b)$. ($a \prec b \prec c =$ true if node a, b, c appear in this order in a sorted circular linked list, following the right link starting from node a.)

```
// r: key range [r.min, r.max]
// d: forwarding direction (LEFT or RIGHT)
// s: query issuing node
// v: MAX value that has been acquired thus far
// k: set of keys that corresponds to v
P.agQuery(r, d, s, v, k)
  if elements of P.keys[maxLevel] are included in r then
    send P.agval[maxLevel] and P.keys[maxLevel] to node $s$
    return
  end if
  {When forwarding to the left (trying to reach the left side of range r)}
  if d = LEFT then
    search for the node $n$ that is closest to r.min and satisfies
    (r.max $\prec$ $n$.key $\prec$ r.min) from the routing table of P (i.e.,
    P.left[], P.right[])
    if such node $n$ exists then
      call agQuery(r, RIGHT, s, v, k) on node $n$ [1]
    else
      let $n$ be the node that is closest to r.min and satisfies (r.min
      $\prec$ $n$.key $\prec$ P.key), found in the routing table of P
      call agQuery(r, LEFT, s, v, k) on node $n$
    end if
    return
  end if
  {When forwarding to the right}
  if d = RIGHT then
    if P.key is included in r and a level $j$ exists that satisfies (P.key
    $\prec$ r.max $\prec$ P.right[$j$]) and v > P.agval[$j$] then
      {Corresponds to the case where $x > y$ in SectionIII-B1 case (6)}
      send v and k to node $s$
      return
    end if
    {The process for finding $i$ that satisfies the conditions in the next if
    statement corresponds to (3) or (6)}
    if level $k$ exists such that elements of P.keys[$k$] are included
    within r (let $i$ be the largest value for such $k$) then
      {Update the $v$ and $k$}
      if P.agval[$i$] > v then
        v = P.agval[$i$], k = P.keys[$i$]
      else if P.agval[$i$] = v then
        k = k $\cup$ P.keys[$i$]
      end if
      {Terminate if the query exceeds the rightmost end of $r$}
      if r.min $\prec$ r.max $\prec$ P.right[$i$].key then
        {Corresponds to (1) or (5)}
        send v and k to node s
        return
      end if
      {Corresponds to (2) or (7)}
      call agQuery(r, RIGHT, s, v, k) on P.right[$i$]
      return
    else
      if P.key $\prec$ r.max $\prec$ P.right[0].key then
        {No node exists within the range}
        send null to node s
      else if P.right[0].key is included in r then
        {The case that P is the node directly to the left end of $r$}
        call agQuery(r, RIGHT, s, v, k) on P.right[0]
      else
        {Corresponds to (4)}
        search for the node $n$ that is closest to r.min and satisfies
        (P.key $\prec$ $n$.key $\prec$ r.min) from the routing table of P
        call agQuery(r, RIGHT, s, v, k) on $n$
      end if
      return
    end if
  end if
```

C. Aggregates Collecting Algorithm

Because nodes may join or leave, and the *value* of nodes may change, we periodically collect and update the agval[] and keys[] of each node. We next explain the algorithm used to accomplish this.

1) Algorithm overview: Each node that participates in an aggregation skip graph periodically sends an *update message* to collect agval[] and keys[] for each level. This is shown in Fig. 4. The thick line in the figure indicates the message flow when node 5 sends an update message[2]. The update message is forwarded to the left node, starting from level $\mathrm{maxLevel}-1$. If the left node is the originating node of the update message (node 5), then the level is decreased and the forwarding continues. Finally at level zero, the message returns to the originating node (node 5).

The update message contains v[] and k[] to store the MAX value and the corresponding key set for each level. While the message is being forwarded to the left at level i, the MAX value of agval[i] on the node that the message passes is collected in v[$i+1$]. When the message returns to the originating node, agval[i] ($i > 0$) is calculated using the following formula:

$$\mathrm{P.agval}[i] \leftarrow \max\{v[j] \mid 0 \leq j \leq i\}.$$

The lower part of Fig. 4 shows v[] in an update message that each node forwards to the left node. When the originating

[1]We assume using RPC (Remote Procedure Call) or message passing.

[2]When a node joins an aggregation skip graph, all entries of agval[] are initially set to the *value* of the node. In the figure, it is assumed that node 5 has newly joined the aggregation skip graph.

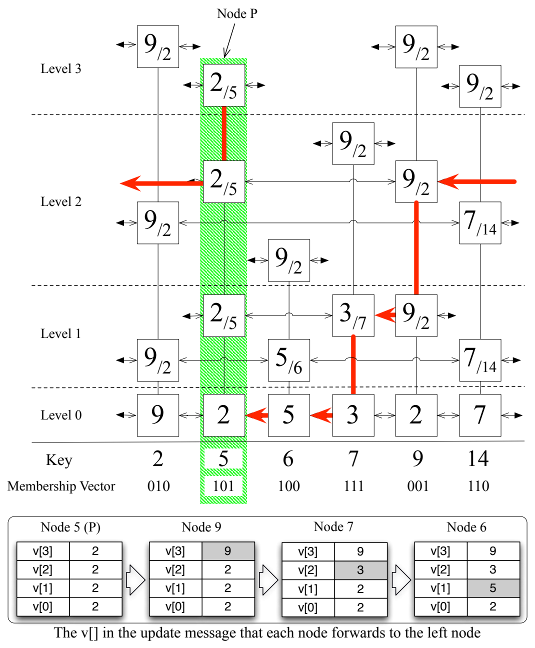


Fig. 4. Message flow of an update message

node (node 5) receives the update message, its agval[] is set to $\{2, 5, 5, 9\}$.

To obtain the correct agval[] and keys[] for every node, each node must execute the update procedure described above maxLevel times because the accuracy of the information stored in an update message level i (i.e., v[i] and k[i]) depends on the accuracy of agval[$i-1$] and keys[$i-1$] of each node that the message passes.

2) Algorithm details: Here, we present the detailed algorithm using pseudocode.

Each node periodically execute `update(maxLevel - 1, P.agval[], P.keys[], P)` for updating its agval[] and keys[] .

```
// lv: level
// v[]: array of aggregated value
// k[]: array of key set
// s: originating node of update message
P.update(lv, v[], k[], s)
  {When the message returns to the originating node}
  if lv = 0 and P = $s$ then
    for $i = 1$ to maxLevel do
      find $j$ where v[$j$] is the MAX value from v[0] to v[$i$]
      P.agval[$i$] $\leftarrow$ v[$j$]
      P.keys[$i$] $\leftarrow$ k[$j$]
      {If multiple $j$'s correspond to the MAX value, k[$j$] is a union set
      of them}
    end for
    return
  end if
  {A larger value is found}
  if v[lv + 1] $<$ P.agval[lv] then
    v[lv + 1] $\leftarrow$ P.agval[lv]
    k[lv + 1] $\leftarrow$ P.keys[lv]
  else if v[lv + 1] = P.agval[lv] then
    k[lv + 1] $\leftarrow$ k[lv + 1] $\cup$ P.keys[lv]
  end if
  {Find a level from the top where the left node is not $s$}
  for i = lv downto 1 do
    if P.left[i] $\neq$ s then
      call update(i, v, k, s) on P.left[$i$]
      return
    end if
  end for
  call update(0, v, k, s) on P.left[0]
  return
```

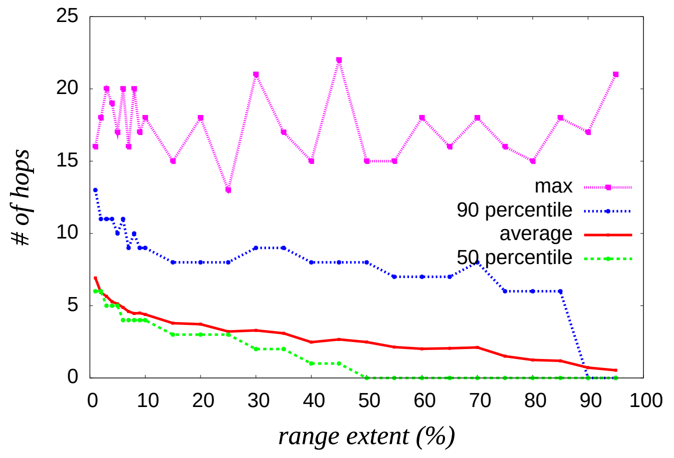


Fig. 5. Relation between range size and number of hops on computing MAX with aggregation skip graphs

IV. Evaluation and Discussion

In this section, we give some evaluation and discussion of the aggregation skip graph. We take n as the total number of nodes, r as the target range for the aggregation query.

A. Cost of the Aggregation Query

Here, we examine the number of messages and hops required for an aggregation query. The algorithm proposed in this paper does not send messages to multiple nodes simultaneously, so the required number of hops and number of messages are equal.

In the worst case, the query algorithm in Section III-B gives the maximum number of hops when the closest nodes on both sides outside of range r store values that are larger than the largest value within r. In such cases, the aggregation query message (1) first arrive at the node immediately to the left of r, and then (2) reach the rightmost node within r. Thus, the upper bound of the number of hops is $O(\log n + \log r)$.

However, in an aggregation skip graph, as the target range of the aggregation query becomes wider, the probability becomes higher that the MAX value stored in agval[] of each node falls within the query range. Therefore, on average the aggregation query is able to be executed in fewer hops. To evaluate the number of hops, we ran the following simulation.

We set the number of nodes (n) as 1,000 for the simulation. We assigned a key and value to each node using a random number between 0 and 9,999. We also assigned the membership vector using a random number.

Next, we registered each node in the aggregation skip graph. We also set the agval[] and keys[] of each node using the algorithm discussed in Section III-C.

We executed the simulation varying the size of the aggregation query range, from 1% to 95% of the key value range (0 to 9,999). (We used 1% steps from 1% to 10%, and 5% steps beyond 10%.) We measured the maximum, the average, and the 50th and 90th percentiles, of number of hops.

The trial was performed 1,000 times for each range size. For each experiment, the initiating node of the aggregation query and the range of the aggregation query were selected using random numbers.

The results are shown in Fig. 5. The x-axis shows the width of the aggregation query range, and the y-axis shows the number of hops.

From the graph, we can confirm that as the target range of the aggregation query becomes wider, the number of average hops decreases. In addition, according to the 50th percentile value, we can see that the query often terminates in 0 hops when the range size is large.

The maximum number of hops is not stable. This is because there is no tight upper bound of hops in skip graphs; it is affected by distribution of membership vectors.

B. Cost of Collecting Aggregates

As discussed in Section III-C, each node must periodically collect aggregates into its agval[] and keys[]. Here, we discuss the cost of this procedure.

On average, the number of hops that the message is forwarded to the left node in one level is about 1 hop, assuming uniformity of membership vectors. Considering the average maxLevel is in $O(\log n)$, an update message sent from a particular node will be forwarded $O(\log n)$ times on average before it returns to the node.

Each node executes this procedure periodically. If the period of this procedure is t, then $O(n \log n)$ update messages are forwarded by all nodes in t. Because these messages are scattered over n nodes, each node processes $O(\log n)$ messages in t on average.

Let us investigate this additional cost is worth paying by comparing the estimated number of messages for an aggregation skip graph with that for a conventional skip graph using simple range queries.

We assume that each node issues q aggregation queries in period t and each query targets pn nodes ($0 < p \leq 1$). In the conventional method, $qn(c_1 \log_2 n + pn)$ messages are issued in t, where c_1 (and c_2, c_3 below) is a constant factor. In the aggregation skip graph, $nc_2 \log_2 n$ messages are issued for collecting aggregates and $qn(c_1 \log_2 n + c_3 \log_2(pn))$ messages are issued for aggregation queries, which sum up to $nc_2 \log_2 n + qn(c_1 \log_2 n + c_3 \log_2(pn))$ messages in t. Note that while we assume the worst case situation on computing the MAX (see Section IV-A), this formula can be also applied to the case when computing other aggregates such as AVERAGE, SUM or COUNT, as we will describe later in Section IV-D.

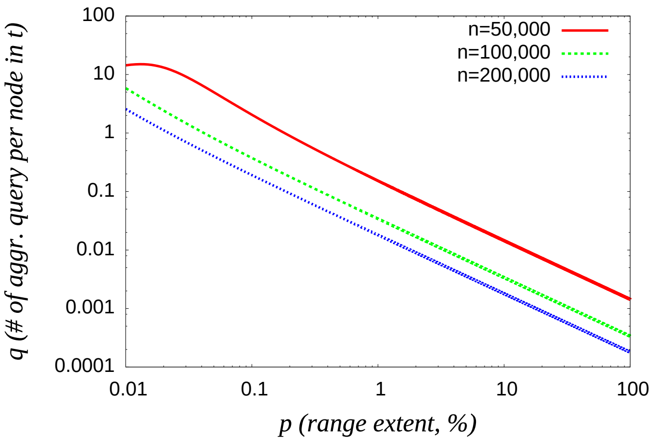


Fig. 6. Graph to determine which, the aggregation skip graph or the conventional skip graph, is efficient. The area under the curve is the region where the conventional skip graph is more efficient.

The aggregation skip graph is more efficient than the conventional method with regard to the number of messages if $qn(c_1 \log_2 n + pn) > nc_2 \log_2 n + qn(c_1 \log_2 n + c_3 \log_2(pn))$, which is equivalent to $q > (c_2 \log_2 n)/(pn - c_3 \log_2(pn))$.

Fig. 6 is a plot of function $(c_2 \log_2 n)/(pn - c_3 \log_2(pn))$ in several n, varying p from 0 to 1. The x-axis shows p and the y-axis shows the q, both in logarithmic scale. We use $c_2 = 1.08, c_3 = 0.91$, that are obtained from a preliminary experiment (c_1 is eliminated in the function). The area under the curve is the region where the conventional skip graph is more efficient. If we assume that $n = 200,000$ and each node issues 0.1 queries in t, the aggregation skip graph is more efficient in the case that the query covers more than about 0.18% of nodes, or 360 nodes. We can conclude that the aggregation skip graph is more efficient than the conventional skip graph unless the query range is very small.

C. Recovering from Failures

Due to a node failure or ungraceful leaving, the link structure of (aggregation) skip graphs might be temporarily broken. In that case, agval[] and keys[] in some nodes might have out-of-date values. However, because these values are periodically updated, this situation is eventually resolved as the link structure of the skip graphs is recovered. (We assume that some repair algorithm for skip graphs is engaged.)

D. Handling Other Aggregates

The algorithm described above targets the MAX as the aggregates, but it is trivial to adapt to the MIN, as mentioned in Section III. To compute other aggregates such as the AVERAGE, SUM or COUNT, the following modification is required.

Instead of storing the MAX value, agval[] stores the sum of the values and number of nodes within each range. When computing the MAX value, the result may be obtained in an early step (i.e., it is not always necessary to reach the nodes at each end of the region.). However, when computing the AVERAGE, SUM or COUNT, it is always necessary to reach the node at both ends. This is equivalent to the worst case situation in computing the MAX (see Section IV-A) and requires $O(\log n + \log r)$ messages and hops.

V. Conclusion and Future Work

In this paper, we have proposed aggregation skip graphs, which allows efficient aggregation queries. The structure of aggregation skip graphs is quite simple; utilizing the structure of skip graphs enables eliminating construction of a reduction tree. Thus, it can be easily implemented over skip graphs. The aggregation skip graph supports computing aggregates on a subset of nodes by specifying key ranges, which cannot be accomplished with conventional aggregation algorithms in P2P networks.

Computing aggregates with aggregation skip graphs requires only $O(\log n + \log r)$ messages, which is a substantial improvement over the $O(\log n + r)$ messages required of range queries over conventional skip graphs (n denotes the number of nodes and r denotes the number of nodes within the query range). In addition, computing the MAX or MIN is quite fast; it requires fewer messages as the query range becomes wider.

The aggregation skip graph has the following drawbacks: (1) additional costs are incurred because each node collects aggregates periodically; and (2) the aggregation query results do not reflect the up-to-date situation because results are based on periodically collected aggregates.[3] However, we believe that aggregation skip graphs are useful for applications that execute aggregation queries over a wide target range.

One of our future work is to devise a method to reduce the cost of collecting aggregates. The following methods should be considered. (1) adaptively adjust the collection period, or (2) update (and propagate) aggregates only when necessary. Another future work is to give an exhaustive comparison between the aggregation skip graph with the existing techniques such as in [7]–[11].

[3]Note that up-to-date aggregates cannot be acquired even in conventional skip graphs or skip tree graphs because some time is required to propagate a query message.

Acknowledgments

We would like to express our gratitude to Dr. Mikio Yoshida at BBR Inc. for his insightful comments and suggestions. This research was partially supported by the National Institute of Information and Communications Technology (NICT), Japan.

References

[1] Toshiyuki Abe, Tatsuya Ueda, Kota Abe, Hayato Ishibashi, and Toshio Matsuura. Aggregation skip graph: An extension of skip graph for efficient aggregation query. In *Proceedings of the 2009 Symposium on Internet and Operation Technology*, pages 75–82. IPSJ, 2009 (in Japanese).

[2] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applic ations. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[3] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[4] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

[5] James Aspnes and Gauri Shah. Skip graphs. *ACM Trans. on Algorithms*, 3(4):1–25, 2007.

[6] William Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33:668–676, 1990.

[7] Carlos Baquero, Paulo Sérgio Almeida, and Raquel Menezes. Fast estimation of aggregates in unstructured networks. In *International Conference on Autonomic and Autonomous Systems (ICAS)*, pages 88–93. IEEE Computer Society, 2009.

[8] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[9] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. Estimating aggregates on a peer-to-peer network. Technical Report 2003-24, Stanford InfoLab, 2003.

[10] Norvald H. Ryeng and Kjetil Nørvåg. Robust aggregation in peer-to-peer database systems. In *Proceedings of the 2008 international symposium on Database engineering & applications (IDEAS'08)*, pages 29–37. ACM, 2008.

[11] Ji Li, Karen Sollins, and Dah-Yoh Lim. Implementing aggregation and broadcast over distributed hash tables. *SIGCOMM Compututer Communication Review*, 35(1):81–92, 2005.

[12] Alejandra González Beltrán, Peter Milligan, and Paul Sage. Range queries over skip tree graphs. *Computer Communications*, 31(2):358–374, 2008.

K-Nearest Neighbor Search in Peer-to-Peer Systems

Hoda Mashayekhi, Jafar Habibi
Computer Engineering Department
Sharif University of Technology
Tehran, Iran
mashayekhi@ce.sharif.edu, jhabibi@sharif.edu

Abstract— **Data classification in large scale systems, such as peer-to-peer networks, can be very communication-expensive and impractical due to the huge amount of available data and lack of central control. Frequent data updates pose even more difficulties when applying existing classification techniques in peer-to-peer networks. We propose a distributed, scalable and robust classification algorithm based on k-nearest neighbor estimation. Our algorithm is asynchronous, considers data updates and imposes low communication overhead. The proposed method uses a content based overlay structure to organize data and moderate the number of query messages propagated in the network. Simulation results show that our algorithm performs efficiently in large scale networks.**

Keywords- Classification; K-Nearest Neighbors; Content Addressable Network; Peer-to-peer systems.

I. INTRODUCTION

Huge amounts of data are available in large scale systems such as peer-to-peer (P2P) networks. Data mining in these systems can utilize the distributed resources of data and computation. Many applications such as smart recommendations, query answering, failure determination, and market analysis can benefit from the hidden information and patterns in the distributed data. Due to large computation overhead, scalability and privacy issues, it is impractical to collect the data at different nodes in a central server. In an alternative approach, each node can execute the data mining algorithm to produce a local model of its data. Some approaches transmit these models or representatives of the data to a central server for integration [3][4]. However, pure distributed data mining approaches do not require a central server [2][5]. Datta et al. present an overview of data mining in P2P networks [10].

Well known classification algorithms like decision tree induction [6], nearest-neighbors [7], Bayesian methods [8] and Support Vector Machines [9] require data to reside on a central site. Majority of the available classification techniques have tried to minimize computation costs and number of disk accesses. Whereas different requirements exist in P2P systems, such as minimizing communication overhead, preserving privacy, etc. Special attempts have been made to design distributed classification algorithms for large scale dynamic P2P environments [11].

In this paper we introduce a distributed K-Nearest Neighbors (KNN) [7] algorithm. We apply our proposed algorithm in the Content Addressable Network (CAN) [1], which is a well-accepted P2P overlay network supporting multidimensional data. In our design neighbor peers collaborate to find the k-nearest neighbors of the query. This collaboration is guided by the CAN structure to form a local algorithm [5]. We generally analyze the complexity of our algorithm. The simulation results show that our algorithm can efficiently perform classification in P2P networks.

The rest of the paper is organized as follows. In Section 2, we review the related work. Section 3 describes the system model and basic assumptions. Section 4 elaborates the distributed classification algorithm. Simulation results are presented in Section 5, and finally, Section 6 presents conclusion and future work.

II. RELATED WORK

Various proposals exist for classification of data in distributed systems [11]. Many of the proposed nearest neighbor algorithms assume a centralized collection of data. Seidl et al. propose a multi-step k-nearest neighbor search algorithm to overcome the shortcomings of single step algorithms [12]. Their algorithm tries to minimize the number of exact object distance evaluations and is suitable for high dimensional and adaptable distance functions. Roussopoulos et al. provide a branch and bound algorithm for processing k-nearest neighbor queries utilizing the R-tree structure [13].

The majority of distributed nearest neighbor classification methods gather data at a central site before executing the classification algorithm. Li et al. use k-nearest neighbor classification in distributed sensor networks for collaborative signal processing [14]. The actual classification is performed in a central node which collects the data of the other nodes in its region. A monitoring algorithm for k-nearest neighbor queries over moving objects is proposed by Yu et al [15]. They propose two methods based on indexing objects or queries, in a two dimensional space. Song et al. propose methods for finding k nearest neighbors of a moving query point. Their algorithm uses R-tree structure [16].

A. *K-nearest Neighbor Classification*

Nearest neighbor classifiers [7] are based on learning by analogy. They compare a given query tuple with training data tuples that are similar to it. Each data tuple is described by m attributes. So each tuple represents a point in the m-dimensional space. Given a query tuple with an unknown class attribute, the KNN classifier searches the m-dimensional space for k data tuples that are closest to the query tuple according to a distance function. The query tuple is assigned the most common class among these k nearest data tuples. To determine the nearest tuples, a suitable distance metric such as Euclidean distance should be used.

B. *Content Addressable Network*

CAN [1] considers a virtual m-dimensional Cartesian logical coordinate space on a m-torus. The entire coordinate space is dynamically partitioned among all the peers in the network, such that each peer owns a distinct zone within the overall space. A hash function is used to map each (key, value) pair in the network to a point in the coordinate space. The pair is then stored at the node that owns the zone containing the corresponding point. Retrieval of any entry corresponding to a key is similar; after applying the hash function to the desired key, the related value is retrieved from the appropriate zone owner.

Whenever a new node joins the network, it chooses a random point in the coordinate space and joins the zone containing that point. The zone owner splits its zone along a dimension, chosen on a round robin fashion, and transfers the data belonging to half of the zone to the new node. Also when a node leaves the network, it assigns its zone to one of its neighbor nodes. After any join or leave, the owners of neighbor zones should be informed of the new situation. Also control messages are propagated when a new data point is added to the network. Some other control messages are discussed in [1].

For efficient routing of queries, any node keeps pointers to neighboring zones along each dimension. Any message is always forwarded to the neighbor node which is closer to the destination zone.

III. System Model

A network of size N is assumed, where each peer stores a portion of the available data in the network. Each data tuple, d, is represented by a m-dimensional attribute vector $(d_1, d_2, \ldots, d_m)$. The query is also described by a similar vector. The peers form an m-dimensional CAN overlay over the coordinate space. Attribute vectors are used, instead of the hash function, to map data tuples to points in the coordinate space. To handle dynamics in network topology and also changes in peers' data, the mechanisms introduced in the CAN proposal are employed [1].

Two zones are considered adjacent if they share at least one point in their boundary. Note that this definition defers from the definition of neighboring zones in the CAN proposal [1]. Regarding the latter definition, adjacent zones reside at most two hops away from each other. The distance of any data tuple to an arbitrary zone is the length of the shortest line between the data tuple and the zone boundary. The zone owned by a peer and the data tuples mapped to that zone, are called the local zone and local data of that peer respectively. When a query is initiated, the zone containing the query tuple is called the query local zone and the owner of this zone is referred to as query owner. Also the term region refers to a collection of at least one zone in the CAN overlay.

IV. The Distributed Classification Algorithm

As a consequence of using attribute vectors to map data tuples to the coordinate space, similar data tuples will reside in approximate zones. The basic idea of our algorithm is that the nearest neighbors of a query are discovered, with high probability, in the query local zone and its approximate zones. This probability is directly proportional to the density of data objects in the zones and inversely proportional to the value of k in the KNN algorithm.

The CAN overlay can be leveraged to find the nearest neighbors of a query in an iterative manner. Figure 1 shows the distributed k-nearest neighbors algorithm. The symbols used in the algorithm are summarized in table 1. When a query q is initiated by a peer, it is routed to the query local zone and the query owner initiates the KNN algorithm. In any iteration, a search region (SR) consisting of zones which may contain the k nearest neighbors of q is investigated. Also a candidate data set (CS) is maintained in different iterations. This set contains the data tuples that may be among the k nearest neighbors of q. The k nearest neighbors of q will eventually be excluded from this set and added to the final answer set.

The main part of the distributed algorithm for finding the k nearest neighbors is iteration on three tasks:

1) Searching the current search region for nearest data tuples (lines 7-16 of algorithm 1).

2) Updating the final answer set (lines 17-23 of algorithm 1).

3) Determining the next search region (lines 24-29 of algorithm 1)

The above tasks are repeated until the final answer set size is k or the search region for the next iteration is empty. Note that for correct behavior of the algorithm it is necessary that the structure of the zones containing k nearest neighbors of q remain unchanged. Also as query owner stores contact information of zone owners in the search region, if these information changes additional routing is required to complete the algorithm.

In the next subsections each task is explored in depth. In the following sections the subscript of the symbols in table 1, is a representative of the iteration number.

A. *Searching the current search region*

The first task of algorithm 1 consists of sending query messages by the query owner to all the zone owners in the current search region. The search region for the first iteration is the query local zone, thus in the first iteration the query owner will send a query message to itself. At the beginning of any iteration if the search region is empty, then $k - |KN|$ data tuples with minimum distance to q are extracted from CS and added to KN. The algorithm is then terminated. The candidate data set and the final answer set are empty at the beginning of the algorithm.

All of the nodes that receive a query message should search their local zones for candidate data tuples that may be among the k nearest neighbors of q. A subset of their local data satisfying the conditions of lemma 1 is sent back to the query local zone owner which will insert them in the

TABLE I. DESCRIPTION OF SYMBOLS

Symbol	Description
KN	The final answer set containing k nearest neighbors
CS	The set containing candidate data tuples
SR	The search region
SZ	Previously searched zones
BDS	The set containing distances to adjacent zones
RZ	The set containing received information of adjacent zones

candidate data set. They will also compute the distance between q and all of their adjacent zones, and send the set of adjacent zone owners' addresses along with the computed distances to the query local zone owner. This information will be utilized later in the algorithm.

Lemma 1. Let $D_{z'}$ denote the data of zone z'. The owner of zone z' which has received the query message in iteration i, will reply back with a subset $SD_{z'}$ of its local data, such that $|SD_{z'}| \le (k - |KN_{i-1}|)$. Also the following property holds:

$$\forall\, d \in SD_{z'}.\ \forall d' \in D_{z'} \backslash SD_{z'}.\ \mathrm{Dist}(q, d') \ge \mathrm{Dist}(q, d) \quad (1)$$

If $|CS_{i-1}| > (k - |KN_{i-1}|)$, then we also have:

$$\forall\, d \in SD_{z'}.\ \mathrm{Dist}(q, d) \le \max\{\mathrm{Dist}(q, d') | d' \in CS_{i-1}\} \ (2)$$

First note that size of $SD_{z'}$ is at most equal to number of desired data tuples, $k - |KN_{i-1}|$, so that no extra data is transmitted. Inequality (1) emphasizes that the data tuples in $SD_{z'}$ have minimum distance to q among all the data tuples in $D_{z'}$. Also if size of CS_{i-1} is greater than number of desired data tuples, no data tuple which has greater distance to q than all the data tuples in CS_{i-1}, can be among the k nearest neighbors of q, thus (2) should hold.

After receiving replies from all of the nodes in the search region, the query owner will set $CS_i = \bigcup_{z' \in RZ_i} SD_{z'} \cup CS_{i-1}$.

B. Updating the final answer set

After obtaining CS_i in the previous task, the query owner should now examine CS_i to extract suitable data tuples and add them to the final answer set.

Lemma 2. Let D_i be the set of data tuples mapped to region SR_i. If C is the maximum subset of CS_i such that

$$\forall\, d \in C.\ \mathrm{Dist}(q,\ d) < \min\{\mathrm{Dist}(q, z') | z' \text{is adjacent to } SR_i\ \wedge\ z' \notin\ \textstyle\bigcup_{j=1}^{i} SR_j\},$$

then $KN_i = KN_{i-1} \cup C$ and . $CS_i = CS_i \backslash C$. Note that if $|KN_{i-1} \cup C| > k$, then k points with minimum distance to q will be kept.

Using lemma 2, any candidate data tuple which is closer to q than to any adjacent zone of SR_i which is not investigated before, is extracted from CS_i and added to KN_i. We have the following statement:$\forall\, d \in \bigcup_{j=1}^{i} D_j\ .\ \forall\, d' \in CS_i\ \ \mathrm{Dist}(q, d) \ge \mathrm{Dist}(q, d')$. If this inequality does not hold for a pair of data tuples d and d′ such that $d \in \bigcup_{j=1}^{i} D_j$ and $d' \in CS_i$, then d can replace d′ in the set of candidate data tuples. Also the following property holds for any data tuple d belonging to an adjacent zone:

$$\forall d \in C.\ \forall d' \in z' | (z' \text{is adjacent to } SR_i \wedge z' \notin \textstyle\bigcup_{j=1}^{i} SR_j).\ (\mathrm{Dist}(q, d') \ge \mathrm{Dist}(q, z') \ge \mathrm{Dist}(q, d)\}. \quad (3)$$

So d is one of the k nearest neighbors of q.

C. Determining the next search region

If $|KN_i| < k$, the search area should be expanded. Lemma 3 is used to construct the search region for the next iteration.

Lemma 3. In any iteration we have:

```
Algorithm 1. Finding k nearest neighbors of q
z: query local zone
$O_z$: query owner
$O_z$. Find_k_nearest_neighbors(q,k)
 1: KN = $\emptyset$, SR = z, SZ = $\emptyset$, CS = $\emptyset$, BDS = $\emptyset$
 2: while $|KN| < k$ do
 3:   if |SR| is zero then
 4:     add $k - |KN|$ tuples from CS with minimum distance
        from q, to KN
 5:     terminate the algorithm
 6:   end if
 7:   RZ= $\emptyset$
 8:   send query message to owners of zones in SR with
      parameters ( q, k- |KN|, |CS|, $\max\{\mathrm{Dist}(q, d) | d \in CS\}$)
 9:   RZ = $\emptyset$   // the received candidate adjacent zones
10:   while not received reply from a zone owner in SR do
11:     receive message from a zone owner o in region SR
12:     CS= CS $\cup$ received data tuples
13:     RZ = RZ $\cup$ received adjacent zones
14:     BDS = BDS $\cup$ received boundary distances
15:   end while
16:   SZ = SZ $\cup$ SR
17:   minBDS = $\min\{\mathrm{Dist}(q, b) | b \in BDS\}$
18:   for i=1 to |CS| do
19:     if $\mathrm{Dist}(q, d_i \in CS) <$ minBDS then
20:       KN = KN $\cup$ $\{d_i\}$   //if |KN| = k replace one of the
                    tuples in $KN_q$ with $d_i$ if the overall sum of
                    distances is improved
21:       CS = CS$\backslash\{d_i\}$
22:     end if
23:   end for
24:   maxD = $\max\{\mathrm{Dist}(q,\ d_i) | d_i \in CS\}$
25:   if (|CS| < (k − |KN|)) then
26:     SR = RZ$\backslash\{z | z \in SZ\}$
27:   else
28:     SR = $\{z' \mid z' \in RZ \wedge \mathrm{Dist}(q, z') < \mathrm{maxD}\}\backslash\{z | z \in SZ\}$
29:   end if
end while
```

Figure 1. The distributed KNN algorithm

$$\forall \mathrm{i} > 1. \forall j < i. \mathrm{SR_i} \cap \mathrm{SR_j} = \emptyset \quad (4)$$

Also for any zone $\mathrm{z'}$ which is adjacent to $\mathrm{SR_i}$ and $\mathrm{z'} \notin \bigcup_{\mathrm{j}=1}^{\mathrm{i}} \mathrm{SR_j}$ we have:

$$\text{If } |\mathrm{CS_q}| < k - |\mathrm{KN_i}| \text{ then } \mathrm{z'} \in \mathrm{SR_{i+1}}$$

$$\text{else } (\exists\, \mathrm{d} \in \mathrm{CS_q}. \mathrm{Dist}(\mathrm{q},\mathrm{d}) > Dist(\mathrm{q},\mathrm{z'}) \Rightarrow \mathrm{z'} \in \mathrm{SR_{i+1}} \quad (5)$$

If less than $\mathrm{k} - |\mathrm{KN_i}|$ data tuples are available in the candidate data set, then any zone adjacent to the current search region which is not investigated before should be included in the next search region, as it may contain closer points to the query. Else, the distance between q and adjacent zones should be calculated. Only those zones are included in the next search region that the distance between them and q is less than the distance of q to at least one of the candidate data tuples. Consequently these zones may contain data tuples that are closer to the query. The distance between q and zones adjacent to the current search region is calculated by zones in $\mathrm{SR_i}$ and sent to the query owner in task 1. Also addresses of the adjacent zone owners, is sent to the query owner so that in subsequent iterations, it can contact the adjacent zones directly. Note that $\mathrm{SR_{i+1}}$ does not contain any zones searched in the previous iterations.

If the total number of data tuples in the network is greater than k, then the termination of the algorithm is guaranteed. Figure 2 illustrates examples of non expandable (a) and expandable (b) search regions in a 2-dimensional CAN overlay for k=5. In Figure 2 (b), the distance from q to the zones z2-z4 is less than the distance from q to data tuple 1. Therefore these zones should be investigated in the second iteration.

V. Analysis and Experimental results

In this section we present a general analysis on the message complexity of our algorithm in a static network.

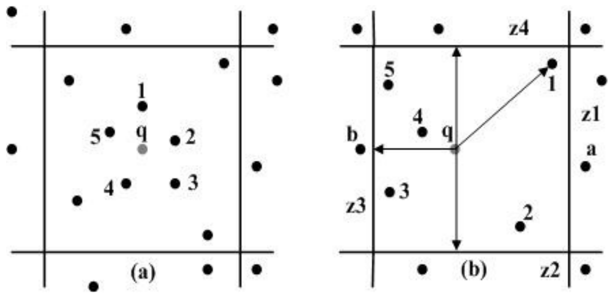


Figure 2. Examples of (a) nonexpandable and (b) expandable search regions for k=5.

Recall the three steps of the algorithm. Note that only step 1 imposes communication in the network. In any iteration the query owner sends a message to all zone owners in the search region. As the search regions in different iterations do not have any zones in common, no zone receives more than one message from the query owner. Also in other than the first iteration the query owner receives addresses of adjacent zones from the zones in the search region. So it can contact them directly without routing the message in the CAN overlay. So if we could determine the maximum number of zones investigated in the algorithm, the number of messages could be calculated.

Define the diameter of an m-dimensional hypercube as the largest distance between any two of its vertices. Assume that the minimal hypercube centered at the q, which contains at least k data tuples other than q, has diameter l. So the distance from q to any of these k data tuples is at most l. To determine the maximum number of zones searched by the algorithm, consider the minimal m-dimensional hypercube centered at q, which contains the union of all search regions in different iterations. By extending lemma 3, it is induced that the edge size of this hypercube should be at most $2\mathrm{l} + \varepsilon$, where $\varepsilon \to 0$, so that the algorithm investigates all the zones that may contain the k nearest neighbors. This observation shows that the communication overhead of our algorithm is independent of the network size and it is considered a local algorithm [5]. Having the average number of data tuples contained in any zone of the CAN overlay, the minimum number of zones that are investigated can be determined.

We conducted a simulation to evaluate our proposed algorithm in a CAN network. We used two different synthetically generated data sets containing 20000 data tuples in our experiments. Figure 3 shows the snapshots of the test data used to evaluate the algorithm, which are generated using the uniform and 5 Gaussian distributions with pre selected centers and standard deviation. Each simulation is executed 10 times for random queries and the average value is displayed. Similarity is measured using Euclidean distance.

We have also compared our algorithm with the KNN algorithm executed on P2PR-tree [17]. We extended the NN algorithm proposed in [18] for KNN by modifying the pruning rule for k data tuples, and implemented it on top of P2PR-tree. As each node in the P2PR-tree keeps information about the top level of the tree, it can initiate the KNN algorithm. Different parameters used in P2PR-tree are set as follows: number of blocks and groups in each block is set to 10, number of routers of each node is set to 1, $\mathrm{G_{Max}}$ and $\mathrm{SG_{Max}}$ are set to 50 and minimum number of nodes in a subgroup is set to 20. Tree vertices other than blocks and leaves have two children. The tree node splitting algorithm is introduced in [20]. We consider a power law distribution of data among the peers. Each peer is assigned a maximum of 100 data tuples based on the Sarioiu distribution [19].

As our algorithm uses CAN overlay network as its base, it is useful to consider the message complexity incurred by maintaining this overlay. To better reveal the efficiency of the algorithm, we have compared the message complexity of our algorithm with the simple distributed KNN algorithm in which queries are broadcasted in the network. In the latter algorithm an unstructured P2P network is considered. In such a network, a new node sends join requests to some

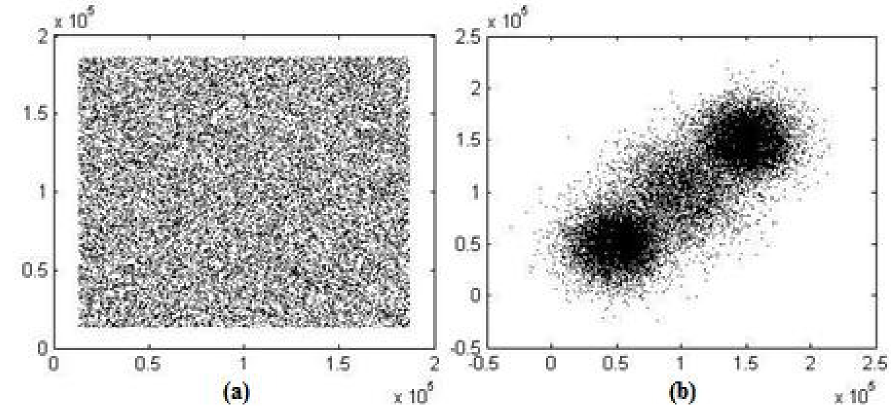


Figure 3. (a) Two dimensional uniform data, (b) Two dimensional mixture of Gaussian data sets

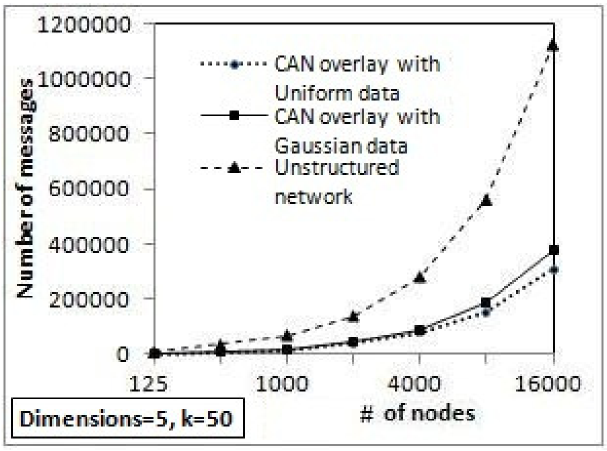


Figure 4. Number of messages incurred in the CAN network and unstructured network

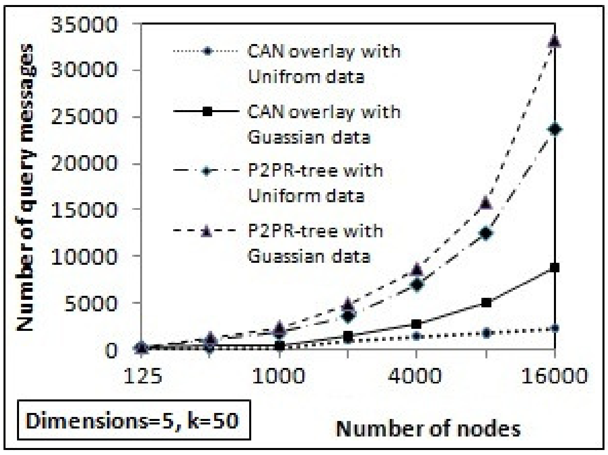


Figure 5. Query messages incurred by algorithms when number of nodes is increased

randomly chosen nodes. But when leaving the network no message overload is incurred.

Figure 4 shows the average number of messages per request in a dynamic CAN network compared to an unstructured network. A request can be a join, leave, or KNN query request and also addition of new data to the network. As seen in the figure, although our algorithm imposes high control traffic load in the network, it outperforms executing the KNN algorithm in the unstructured network, due to the low query traffic.

Figure 5 shows the message complexity of our algorithm and the KNN algorithm in P2PR-tree under different network sizes. As observed, the number of query messages for Gaussian data is larger than uniform data. In the former case, there are sparse areas in the network with few data tuples. If the query resides in these areas, more query messages should be propagated in the network. Also, Figure 5 exposes the efficiency of our algorithm compared to KNN in P2PR-tree.

The performance of our algorithm when the nodes form a regular m-dimensional mesh is shown in Figure 6 and Figure 7. In such networks, number of adjacent zones of a particular zone is proportional to number of dimensions. So as observed in Figure 6, when number of dimensions is 10, the query traffic is much less than the same configuration in

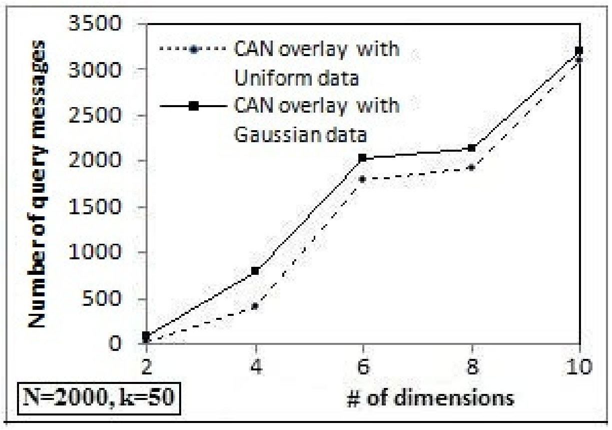


Figure 6. Effect of changing number of dimensions on number of query messages

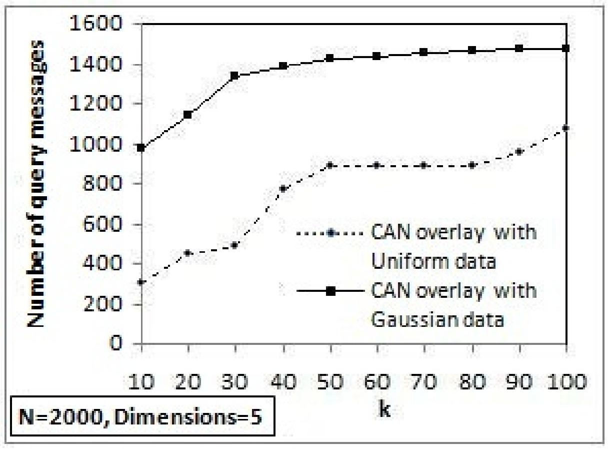


Figure 7. Effect of changing the value of parameter k on number of query messages

Figure 5. Obviously when number of dimensions increases, more query messages are propagated through the network. Figure 7 shows the effect of changing the parameter k in the KNN algorithm, on number of query messages. As observed by increasing this parameter, number of query messages increases. The parameter k has a more visible effect when the data tuples are distributed uniformly in the network. This is due to the fact that the average number of nodes' data is the same in the uniform configuration. Therefore by increasing the parameter k, eventually more zones should be investigated.

I. Conclusion

In this paper we studied classification of data objects in P2P networks. We presented a new distributed algorithm for finding the k nearest neighbors of a query in P2P networks. Our algorithm uses a content addressable network to organize data and moderate the number of query messages propagated in the network. Using pruning rules, the number of nodes that should be prompted to find the k nearest neighbors is decreased. Simulation results show the effectiveness of our algorithm in different configuration. We have presented comparisons with the KNN algorithm executed in an unstructured P2P network and in a network with P2PR-tree structure. Extending our algorithm to dynamically adapt the query answer, when nodes leave and join the network or k is updated, remains as future work. Also examining the effectiveness of our algorithm with real world non-numerical data sets can further expose its strengths and weaknesses.

References

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," Proc. 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Press, 2001, pp. 161-172, doi: 10.1145/383059.383072.

[2] M. Li, G. Lee, W. C. Lee, and A. Sivasubramaniam, "PENS: An algorithm for density-based clustering in peer-to-peer systems," Proc. 1st international conference on Scalable information systems, ACM Press, 2006, Article No. 39, doi: 10.1145/1146847.1146886.

[3] J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, "Distributed Data Mining and Agents," Eng. Applications of Artificial Intelligence, vol. 18 (7), 2005, pp. 791-807, doi: 10.1016/j.engappai.2005.06.004.

[4] E. Januzaj, H. P. Kriegel, and M. Pfeifle, "DBDC: Density Based Distributed Clustering," Proc. Ninth Int'l Conf. Extending Database Technology (EDBT '04), Springer Berlin / Heidelberg Press, LNCS, vol. 2992, 2004, pp. 88-105, doi: 10.1007/b95855.

[5] R. Wolff and A. Schuster, "Association Rule Mining in Peer-to-Peer Systems," IEEE Transactions on Systems, Man and Cybernetics - Part B, vol. 34 (6), 2004, pp. 2426-2438.

[6] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1(1), 1986, pp. 81-106, doi: 10.1023/A:1022643204877.

[7] B. V. Dasarathy, "Nearest Neighbor (NN) Norms: NN Pattern Classification techniques," IEEE Computer Society Press, 1991.

[8] T. M. Mitchell, "Machine learning," McGraw Hill Higher Education, 1997.

[9] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," Proc. Fifth Annual Workshop on Computational Learning Theory, ACM Press, 1992, pp. 144-152, doi: 10.1145/130385.130401.

[10] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, "Distributed Data Mining in Peer-to-Peer Networks," IEEE Internet Computing Special Issue on Distributed Data Mining, vol. 10 (4), 2006, pp. 18-26, doi: 10.1109/MIC.2006.74.

[11] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan, "JAM: Java Agents for Meta-Learning over Distributed Databases," Proc. Third International Conference on Knowledge Discovery and Data Mining, AAAI press, 1997, pp. 74–81, doi: 10.1.1.44.5728.

[12] T. Seidl and H. P. Kriegel, "Optimal Multi-Step k-Nearest Neighbor Search," Proc 1998 ACM SIGMOD international conference on Management of data, ACM Press, 1998, pp. 154-165, doi: 10.1145/276304.276319.

[13] N. Rousopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," Proc. 1995 ACM SIGMOD international conference on Management of data, ACM Press, 1995, pp. 71-79, doi: 10.1145/223784.223794.

[14] D. Li, K. D. Wong, Y. Hu, A. M. Sayeed, "Detection, Classification and Tracking of objects in Distributed Sensor Networks," IEEE Signal Processing Magazine, vol. 19 (2), 2002, pp. 17-29, doi: 10.1.1.58.2340.

[15] X. Yu, K. Pu, and N. Koudas, "Monitoring k-Nearest Neighbor Queries Over Moving Objects," Proc. 21st International Conference on Data Engineering, IEEE Press, 2005, pp. 631-642, doi: 10.1109/ICDE.2005.92.

[16] Z. Song and N. Roussopoulos, "K-Nearest Neighbor Search for Moving Query Point," Proc. 7th International Symposium on Advances in Spatial and Temporal Databases, Springer-Verlag Press, 2001, pp. 79-96, doi: 10.1.1.108.8564.

[17] A. Mondal, Y Lifu, and M Kitsuregawa, P2PR-tree: An R-tree-based Spatial Index for Peer-to-Peer Environments, Proc. Current Trends in Database Technology - EDBT 2004 Workshops, Springer Berlin / Heidelberg Press, LNCS, vol. 3268/2005 (516), 2005, DOI: 10.1007/978-3-540-30192-9_51

[18] S. Berchtold, C. Bohm, D. Keim, and H. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In Proc. ACM Symp. on Principles of Database Systems, ACM Press, 1997, pp. 78-86, doi: 10.1145/263661.263671.

[19] S. Saroiu, P. K. Gummad, SD. Gribble, A measurement study of peer-to-peer file sharing systems, Proc of Multimedia Computing and Networking, 2002.

[20] A. Guttman, R-trees: a dynamic index structure for spatial searching, Proc of the ACM SIGMOD Conference on Management of Data, ACM Press, 1984, pp. 47-57, doi: 10.1145/602259.602266.

Optimization of Flows in Level-Constrained Multiple Trees for P2P Multicast System

Michal Kucharzak and Krzysztof Walkowiak
Department of Systems and Computer Networks
Wroclaw University of Technology, Poland
michal.kucharzak@pwr.wroc.pl, krzysztof.walkowiak@pwr.wroc.pl

***Abstract*—Peer-to-Peer multicast defined for overlay networks, has been taking an advantage over IP Multicast during recent years. It derives from the fact that the overlay architecture for P2P streaming provides potential scalability and easy deployment of new protocols independent of the network layer solutions at relatively low costs. In this paper, we focus on modelling and optimization of multiple trees for flow assignment in P2P multicast systems. The optimization covers multicast flow arrangement on multiple paths in order to minimize the overall streaming cost. Due to quality of service requirements in such kind of systems, we apply a basic hop-constrained spanning tree and capacitated spanning tree problems and we define level-constrained multiple trees problem with bandwidth capacity constraints for multicast flow assignment in overlay system. We propose and compare two Mixed Integer Programming formulations for the problem. In addition, we examine multicast flows in relation to various fragmentation of the content.**

***Keywords*-P2P; Multicast; Flows; Network Optimization**

I. Introduction

Multicast network technique delivers information to a group of destinations simultaneously. Over the years, a lot of research, development, and testing efforts have been devoted to multicast support [1][3][12][17][18][21][22][23]. Multicasting implemented by network-aware approaches using Internet Protocol (IP-Multicast) is afflicted with problems derived form scalability, addressing scheme management, flow or congestion control but in contrast, features of overlay architecture provides potential scalability or ease of deployment new, network-layer-independent protocols at relatively low costs. Overlay network strategy expands end-system multicast [4][11][16] and using overlay based systems has become an increasingly popular approach for multicast and streaming, where participating peers actively contribute their upload bandwidth capacities to serve other peers in the same streaming session by forwarding their available content. Overlay multicast flows are realized as multiple unicast flows at level of a network layer.

In this paper, we consider an overlay P2P system and flow assignment problem for multicast application based on multiple delivery trees. We assume constant bit rate of the multicast stream, which can be divided into separate fractional flows. Due to the quality of service requirements for mutlicast trees structure we employ the hop constrained spanning tree problem [5] and we formulate level-constrained multiple trees problem. The main goal of the problem is to minimize the total cost of delivery trees without considering issues related to dynamics of P2P systems and individual algorithm's tree creation. Cost can be treated as a distance between pair of overlay nodes and can refer to delay, physical flow delivery expenses and network maintenance or cross-ISP's payments and depends on bit rate of transferred stream. To solve the problem in optimal way, we formulate two different mixed integer models. First one, based on directed multicommodity flow and second, derived from level-based formulation.

Although the model based on directed multicommodity flows (DMFM) [7][14] has been applied in past works on hop constrained trees for single tree creation, the last one is in general novel. Level-based formulation (LTM) for creating trees was used in our previous works [19][20] but in this paper we improve its first version and apply it for the level-constrained multiple trees flow problem. These formulations provide with offline optimization, which can be used to find lower bound of other solution approaches or for constructing initial topology. Moreover some P2P multicasting systems are generally static (e.g., data distribution in CDN), and results of offline optimization can be applied in such systems to improve the system performance.

Furthermore we compare and contrast these models in quantitative and time-consumption meaning. To compare the performance and effectiveness of the formulations in terms of the execution time we apply the models in Gurobi Optimizer [10]. Our results show that the models can be more efficient and can enable solving instances in shorter time for different cases. We also employ the models for examining in various ISP (Internet Service Provider) topologies the effect of the tree depth (hop limit) and multicast stream fragmentation on the optimal delivery cost.

The remainder of this paper is organized as follows. In Section II we define Level-Constrained Multiple Trees Problem and in Section III we formulate it as a integer optimization problem. In Section IV, we compare and contrast basic features of the models and present time-consuming evaluation of building and solving models with Gurobi Optimizer [10]. Investigations on the impact of stream fragmentation on the total cost are covered in Section V. Section VI concludes the paper and presents ideas for future work.

II. Level-Constrained Multiple Trees Problem

In this paper we consider an overlay multicast session with single source and multiple participating receivers. We assume that, in the overlay system all nodes except of the root node are receivers. The general objective is to stream the content as multicast session, which is divided into T delivery spanning trees representing fractional flows.

For live media applications, a minimized delays or end-to-end latency at each receiver are significant to guarantee the high liveness and quality of service of the streaming media. On the other hand multicast system features should include reliability, availability or even ease of creation of delivery trees. One of the concept for designing of network-based systems with overall quality of service constraints is the hop-constrained minimum spanning tree problem (HMST) [5][6], which is defined as follows: Given a graph $G = (N, E)$ with node set N and edge set E as well as a cost c_e associated with each edge e of E and a natural number H, we wish to find a spanning tree $t \in G$ with minimum total cost and such that the unique path from a specified root node, node r, to any other node has no more than H edges (hops). Service quality can refer to availability and reliability, redundancy at network layer, quality or ease of creation and managing of the tree.

In accordance to flow problems more useful is arc-based (directed edges) formulation of the HMST. The directed formulation replaces every edge $e = \{i, j\}$ in the graph by the two arcs (i, j) and (j, i) and associates to each of these two arcs the cost of the original edge. Analytically, costs can be defined in asymmetric way, that is, cost of arc c_{ij} can vary from c_{ji}. Let A denote set of directed arcs in the directed model. Notice also that - according to overlay network structure - we practically consider a complete directed graph. We assume that there is a bi-directed edge between any pair of overlay nodes except of the root node. In case of the root node r we "remove" directed arcs to the root from any other node, edge $e = \{i, r\}$ is only replaced by one single arc (r, i).

Second, every peer is connected to the overlay network with link, which has a limited upload and download capacity, hence peer's number of all children in all fractional trees is constrained. Speaking in more precise way, the limitation of child nodes for peer depends on number of fractional trees and their streaming rates. The assumptions stated for single tree can be expressed as the capacitated minimum spanning tree (CMST) [15]. In this paper we extend this capacity constraint to multicast system with multiple trees.

The Level-Constrained Multiple Trees Problem (LCMT)

The main aim of the problem is to construct an overlay multicast topology and assign flows on multiple trees, which are limited with maximum hops. We consider stream S of constant bit rate and define constant number of fractional flows as T with fractional streaming rates s_t for each tree t where $S = \sum_t s_t$. This concept can be easily deployed in real systems with existing codecs, i.e., we can steer or manage the number of packets or frames as a bit rate in each tree t. Upload capacity (u_i) and download capacity (d_i) refer to peer's i available upstream and downstream bandwidth, respectively.

Note that, the LCMT contains as particular cases (the case with $T = 1$, $H = N - 1$) a degree constrained spanning tree or capacitated minimum spanning tree, which are NP-Complete problems [15] or (the case with $T = 1$, $SN <= u_i$) a NP-Hard version of the hop-constrained minimum spanning tree [6].

III. Mixed Integer Programming Formulations

The general formulation derives from the well discussed, based on the multicommodity flow model [7][14], which was successfully implemented for the HMST problem. Several previously known and developed formulations for the HMST (single tree) were mainly derived from the multicommodity flow model and were used for problems with limited sizes of either hops or arcs [5][6] or for optimization at network-aware level [2][8][13].

A. Directed Multicommodity Flow Model (DMFM)

The general multicommodity flow model, denoted DMFM or MCF, uses two sets of binary variables. Variables x_{ijt} indicate whether the spanning tree for fractional flow t contains the arc (i, j) and additional set of directed flow variables f_{ijkt} specify if the unique path from the root node r to node k traverses the arc (i, j) in tree t.

In the following IP formulation we consider an overlay system, which can be represented as a complete directed sub-graph, which consists of $N\backslash\{r\}$ nodes and in addition node r with arcs to any i. Hence $(i, j) \in A \equiv (i, j) : i \in N; j \in N\backslash\{i, r\}$. Further notation replaces set N by indexing scheme $1, ..., N$.

indices

$i, j, k = 1, 2, ..., N$	vertices (peers, application layer nodes)
$t = 1, 2, ..., T$	trees (fractional flows)

constants

r	root node ($r \in 1, 2, ..., N$)
H	maximum hops
d_i	download capacity limit in kbps
u_i	upload capacity limit in kbps
s_t	streaming rate of tree t in kbps
c_{ij}	defines cost of 1 kbps transferred from i to j

variables

x_{ijt}	= 1 if the spanning tree t contains arc (i, j); 0 otherwise (binary variable)
f_{ijkt}	= 1 if path from root r to k contains arc (i, j) in tree t; 0 otherwise (binary, auxiliary variable)

objective

$$\min \quad F = \sum_{i} \sum_{j \neq \{i,r\}} \sum_{t} c_{ij} s_t x_{ijt} \tag{1}$$

constraints

$$\sum_{i \neq j} x_{ijt} = 1 \quad \forall j \neq r \quad \forall t \tag{2}$$

$$\sum_{j \neq \{i,r\}} f_{ijkt} - \sum_{j \neq \{i,k\}} f_{jikt} = \begin{cases} 1 & i = r, \forall k \neq \{i,r\}, \forall t \\ -1 & i = k, \forall k \neq r, \forall t \\ 0 & \forall i \neq \{k,r\}, \forall k \neq r, \forall t \end{cases} \tag{3}$$

$$\sum_{i \neq k} \sum_{j \neq \{i,r\}} f_{ijkt} \leq H \quad \forall k \neq r \quad \forall t \tag{4}$$

$$\sum_{i \neq j} \sum_{t} s_t x_{ijt} \leq d_j \quad \forall j \neq r \tag{5}$$

$$\sum_{j \neq \{i,r\}} \sum_{t} s_t x_{ijt} \leq u_i \quad \forall i \tag{6}$$

$$f_{ijkt} \leq x_{ijt} \quad \forall i, \forall j \neq \{i,r\}, \forall k \neq \{i,r\}, \forall t \tag{7}$$

The main goal of the problem (1) is to find T level-constrained spanning trees, which minimizes the total cost of all flows in the system. Formula (2) refers to the completion constraint and assures each node except of the root node has exactly one parent node in each tree t. Constraint (3) derives from the flow conservation concept and guarantees that the solution of fractional flow t is a directed spanning tree rooted at node r. Constraints (4) state that no more than H arcs are in the path from the root node r to any other node k in tree t. Limitation of downloads in the systems is constrained by formula (5). By analogy, we introduce the upload capacity constraint, which must be satisfied with regards to physical outgoing bandwidth and available capacity of any node i. To bound flow variable f and tree variable x, constraints (7) are introduced. These constraints satisfy that every arc (i, j), which transports flow to any node k in tree t $f_{ijkt} = 1$ exists in tree $x_{ijt} = 1$ simultaneously. Equivalently, if arc (i, j) is not belonging to the tree t, $x_{ijt} = 0$, there cannot be any flow $f_{ijkt} = 0$.

B. Level-based Tree Model (LTM)

To model the multicast trees, the LTM exploits a single set of binary variables x_{ijlt}, which equal to 1 if the spanning tree t contains the arc (i, j) and node i is located at level l. We assume that the root r of each tree t is located at the first level ($l = 1$). All children of the root are located at level 2, etc. The proposed notation enables us to set the value of L as a limit on the maximal depth of the tree. The LTM is based on a precedence relation between two adjacent nodes and is defined by analogy to hop-constrained walk [5] or Steiner Tree Problem in a Layered Graph [9].

indices

$i, j = 1, 2, ..., N$ vertices (peers, application layer nodes)
$l = 1, 2, ..., L$ level of the node
$t = 1, 2, ..., T$ trees (fractional flows)

constants

r	root node ($r \in 1, 2, ..., N$)
d_i	download capacity limit in kbps
u_i	upload capacity limit in kbps
s_t	streaming rate of tree t in kbps
c_{ij}	defines cost of 1 kbps transferred from i to j

variable

x_{ijlt} = 1 if arc (i, j) belongs to spanning tree t and i is located at level l; 0 otherwise (binary variable)

objective

$$\min \quad F = \sum_{i} \sum_{j \neq \{i,r\}} \sum_{l>1} \sum_{t} c_{ij} s_t x_{ijlt} + \sum_{j \neq r} \sum_{t} c_{rj} s_t x_{rj1t} \tag{8}$$

constraints

$$x_{rj1t} + \sum_{i \neq \{j,r\}} \sum_{l>1} x_{ijlt} = 1 \quad \forall j \neq r, \forall t \tag{9}$$

$$\sum_{j \neq r} \sum_{t} s_t x_{rj1t} \leq u_r \tag{10}$$

$$\sum_{j \neq \{i,r\}} \sum_{l>1} \sum_{t} s_t x_{ijlt} \leq u_i \quad \forall i \neq r \tag{11}$$

$$\sum_{i \neq \{j,r\}} \sum_{l>1} \sum_{t} s_t x_{ijlt} + \sum_{t} s_t x_{rj1t} \leq d_i \quad \forall j \neq r \tag{12}$$

$$x_{ij(l+1)t} \leq \begin{cases} x_{ri1t} & \forall i \neq r, \forall j \backslash \{i,r\}, l = 1, \forall t \\ \sum_{k \neq \{i,r\}} x_{kilt} & \forall i \neq r, \forall j \backslash \{i,r\}, \forall l \backslash \{1, L\}, \forall t \end{cases} \tag{13}$$

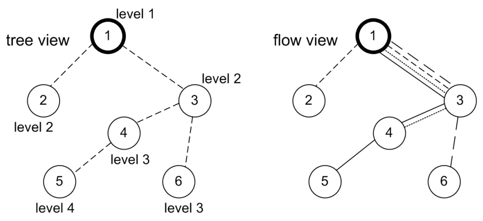


Figure 1. An example of fractional delivery tree for all models (left) and visualization of flow variables in DMFM (right).

The objective function (8) minimizes the spanning trees cost and includes two elements: cost of flows from any $i \neq r$ and arcs, which are starting in the root r. Constraints (9) satisfy that each node is connected to any of tree t, either j's parent is root ($x_{rj1t} = 1$) or any other node i ($\sum_{i \neq \{j,r\}} \sum_{l>1} x_{ijlt} = 1$). Node $j \neq r$ has exactly one and only one parent node situated at exactly one level in each tree t. To clarify upload capacity limitations we introduce (10) and (11) for outgoing streams from root and any other node, respectively. Download capacity constraints include all receiving nodes j (without the root r) and limit the sum of incoming flows in all trees. Due to the fact, that every parent node is situated at exactly one level in each tree t thus to avoid loops, the multiple tree variables x_{ijlt} are created under constraints (13). In this case, every node i can be a parent of any node j at level $l+1$ ($x_{ij(l+1)t} = 1$) if and if only i's parent node k is located at level l ($\sum_{k \neq i} x_{kilt} = 1$).

IV. Models Comparison

Figure 1 presents an example of single, fractional delivery tree t. The flow view refers to the set of flow variables in the DMFM formulation. In Tab. I binary variables x equal to 1 are shown. For the DMFM and the LTM there is only such straightforward representation of the tree from example in Fig. 1

The main advantage of the LTM over directed multi-commodity flow formulation is that we can limit models' sizes (number of variables and constraints) according to the selected hop limit. Table II presents sizes of the models. Variables of the DMFM are x_{ijt}, f_{ijkt}; and the LTM are x_{ijlt}. Note that the maximum usable value for L is $L = N-1$ (in extreme case, the tree is a path bounded with $N-1$ levels for locating parents). In regard to this reason, the number of DMFM's variables exceed variables of the LTM by at least $2N^2 - 2N$ for each fractional tree t. 'Constr.' in Tab. II refers to the number of formulation's constraints in relation to network sizes and represents constraints of (2)-(7) for the DMFM, (9)-(13) for the LTM.

The computational results were obtained on a PC, Intel Core2 Duo, 2.13 GHz, 4GB RAM, Windows 7 Professional. We used the C++ libraries of *Gurobi 2.0.2* with default parameters to obtain the optimal integer solutions of the models tested. We first compare executable build time of

Table I
Equivalent solution for various models

	Example of solution (variables equal to 1)				
Model	1	2	3	4	5
DMFM	$x_{1,2,t}$	$x_{1,3,t}$	$x_{3,4,t}$	$x_{3,6,t}$	$x_{4,5,t}$
LTM	$x_{1,2,1,t}$	$x_{1,3,1,t}$	$x_{3,4,2,t}$	$x_{3,6,2,t}$	$x_{4,5,3,t}$

Table II
Size of the models

	Element	Size
DMFM	Variables	$(N^3 - 3N^2 + 4N - 2)T$
DMFM	Constr.	$(N^3 - 3N^2 + 7N - 5)T + 2N - 1$
LTM	Variables	$((L-1)N^2 + (4-3L)N + 2L - 3)T$
LTM	Constr.	$((L-1)N^2 + (4-3L)N + 2L - 3)T + 2N - 1$

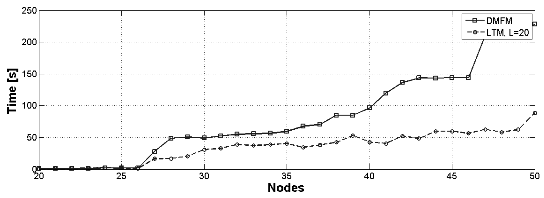


Figure 2. Average time of building models DMFM and LTM

the models. It refers to time, which is needed for dynamic creation of variables and merging all constraints of the problem. Fig. 2 presents average time required for building the models. Average building time of LTM with its upper bound of $L = N-1$ is comparable to DMFM but for any value L less than $N-1$ the LTM is expected to be built faster in contrast to the DMFM, which need, in general, the same number of operations irrespective of the hop limit.

A. Experimentation on Networks

We propose to select the arcs' costs to represent various topologies at level of inter-ISP connections. Fig. 3 shows hypothetical scenarios with different topologies at cross-ISP level. The root node is always in ISP 1 and remaining nodes are distributed among available providers in proportional way. To model these topologies we define cost tables with the following simple algorithm: if any pair of peers belong to the same ISP the cost between them is randomly chosen from 3 to 10. If the shortest path (referred as the number of inter-ISP hops) between peers equals to 1, the cost is set from 20 to 40. Distance of 2 ISP-hops introduces cost in the range of 50-90. We assume symmetric costs ($c_{ij} = c_{ji}$) for all instances but note that costs do not satisfy irregularity of the triangle. In the remaining simulations we assume all peers have the same capacity parameters $512kbps$ for upload and $1024kbps$ for download, streaming rate of the session is $S = 252$ kbps and streaming rate of each fractional tree s_t is derived from the proportional division S into T trees, i.e., if only one tree $T = 1$ is used for the flow allocation its streaming rate $s_1 = 252kbps$, if $T = 3$ then $s_1 = s_2 = s_3 = 84kbps$. We chose $S = 252kbps$ in regards to provide fractional flows with rational values. Note that, for $T = 5$, $s_t = 50.4kbps$, which is still rational number expressed in bps.

Problem's solving times presented in Figs. 4 and 5 were obtained for set of 50 instances with 20 nodes and $T = 1$ for topologies of $S1$ and $S2$ respectively. Results shown there indicate that, the LTM is in general better than DMFM especially in cases of strongly limited number of levels. Average solving time by DMFM and can be approximately treated as constant. Note that time required for solving MIP problems depends not only on the problem size but also on values of input parameters (e.g., costs or root location) and the execution time relation presented in this paper refer to rather overall rough comparison without delving into solver implementation and operating system performance. Moreover, results shown in Fig. 7 presents that, for the much more complicated instances it is worth using DMFM with regards to solve problem in shorter time. This can suggest a general high stability of the DMFM in contrast to the LTM in computational time meaning.

Finally, all of the presented models can be applied for determining the number of required trees for fractional flows, for which total cost of flows assignment can be minimized. Fig. 8 presents optimization costs in relation to number of trees and allowed levels. Note that, the solution is infeasible in cases of hop limit $H = 1$, which means that, the root node is parent of all nodes in all trees and in case $H = 2$ and $T = 1$. The overall conclusion, illustrated in these figures, can be stated as follows: the flow delivery cost for multicast system based on multiple trees can be decreased if more trees with more allowed hops are employed, nevertheless, the most critical impact on delivery cost has limitation of levels.

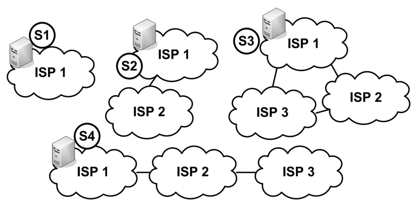


Figure 3. Proposed scenarios of inter ISP connections.

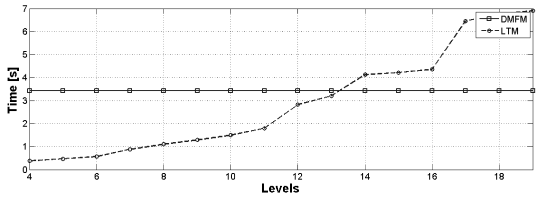


Figure 4. Scenario S1: average time of obtaining optimal result (N=20, T=1).

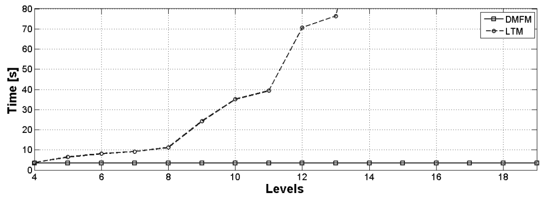


Figure 5. Scenario S2: average time of obtaining optimal result (N=20, T=1).

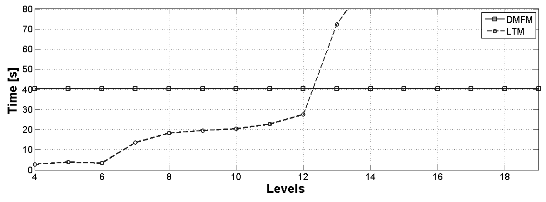


Figure 6. Scenario S1: average time of obtaining optimal result (N=20, T=5).

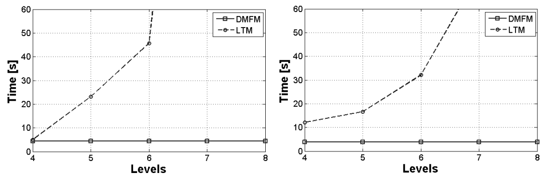


Figure 7. Scenario S3 (left) and S4 (right): average time of obtaining optimal result (N=20, T=1)

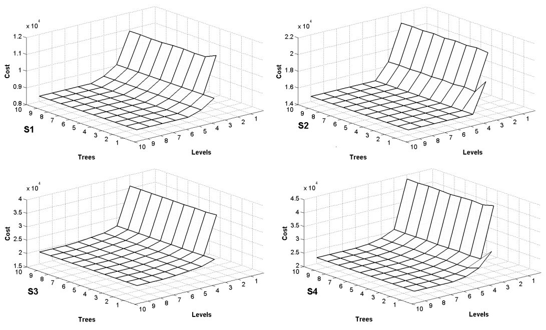


Figure 8. General relation of level and trees number to objective value.

V. Conclusion and Further Work

In this paper, we defined the level-constrained multiple trees problem for multicast streaming purposes. We assumed that the multicast streaming session is divided into separate fractional flows, which spread in the overlay network on multiple spanning trees. These spanning trees are additionally created under the depth (hop, level) constraint. We proposed two mixed integer formulations for the problem and compared as well as contrasted quantitative and time consuming performance of them. The LTM applies a precedence relation between two adjacent nodes, eliminate recording paths to receiving peers and create the spanning tree as short as possible, what requires less number of variables and constraints. Experimentation results with *Gurobi Optimizer* show that the LTM can improve solving the problem in computational time, especially for either limited hops number or small difference between arcs' costs. However, generally most 'deterministic' and predictable results in computational time meaning are provided with DMFM formulation (using *Gurobi*). Moreover, we showed how to manipulate the table cost for simulating various structures of inter-ISP connections and the next step is to analyse more complicated ISP-based topologies with various root locations, different distribution of peers and numerous streaming rates for different tree. Finally, results presented in the paper indicate that, it is worth dividing media stream into separate fractional streams and constructing trees with level limitation greater than 4 provides with relatively short trees without extortionate expenses. Further work includes examination of various overlay networks with much more heterogeneous nodes (i.e., upload and download capacities), more complicated inter-ISP topologies and investigations on different streaming rates of fractional flows. With regards to solve the problem for instances of larger sizes, we plan to design and implement heuristics and metaheuristic algorithms.

Acknowledgment

This work is partially supported by The Polish Ministry of Science and Higher Education under the grant, which is being realized in years 2008-2011.

References

[1] B. Akbari, H. R. Rabiee, and M. Ghanbari, "An optimal discrete rate allocation for overlay video multicasting," *Computer Communications*, vol. 31, no. 3, pp. 551–562, 2008.

[2] A. Balakrishnan and K. Altinkemer, "Using a hop-constrained model to generate alternative communication network design," *ORSA Journal of Computing*, vol. 4, no. 2, pp. 192–205, 1992.

[3] A. Benslimane, Ed., *Multimedia Multicast on the Internet*. ISTE, 2007.

[4] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal resource allocation in overlay multicast," in *in Proc. of 11th International Conference on Network Protocols, ICNP 2003*, 2003.

[5] G. Dahl, L. Gouveia, and C. Requejo, "On formulations and methods for the hop-constrained minimum spanning tree problem," in *Handbooks of Telecommunications*, P. Pardalos and M. G. C. Resende, Eds. New York: Springer, 2006, pp. 493–515.

[6] G. Dahl, "The 2-hop spanning tree problem," *Operations Research Letters*, vol. 23, pp. 21–26, 1997.

[7] L. Gouveia, "Multicommodity flow models for spanning trees with hop constraints," *European Journal of Operational Research*, vol. 95, no. 1, pp. 178–190, Nov 1996.

[8] L. Gouveia and P. Patficio, "Mpls over wdm network design with packet level qos constraints based on ilp models," in *Proc. IEEE Infocom*, 2003.

[9] L. Gouveia, L. Simonetti, and E. Uchoa, "Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs," *Mathematical Programming*, 2009.

[10] Gurobi Optimization, "Gurobi optimizer 2.0," Online, available at http://www.gurobi.com, 2008.

[11] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *in Proceedings of ACM Sigmetrics*, 2000, pp. 1–12.

[12] L. Lao, J. hong Cui, and M. Gerla, "Multicast service overlay design," in *In Proc. of Second International Symposium on Wireless Communication Systems, ISWCS'05*, Philadelphia, Pennsylvania, USA, 2005.

[13] L. J. Leblanc, J. Chifflet, and P. Mahey, "Packet routing in telecommunication networks with path and flow restrictions," *INFORMS J. on Computing*, vol. 11, no. 2, pp. 188–197, 1999.

[14] T. L. Magnanti and L. A. Wolsey, *Handbooks in Operations Research and Management Science*. Elsevier, 1995, vol. 7, ch. 9 Optimal trees, pp. 503–615.

[15] C. H. Papadimitriou, "The complexity of the capacitated tree problem," *Networks*, vol. 8, pp. 217–230, 1978.

[16] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will iptv ride the peer-to-peer stream? [peer-to-peer multimedia streaming]," *Communications Magazine, IEEE*, vol. 45, no. 6, pp. 86–92, 2007.

[17] S. Shi and J. S. Turner, "Multicast routing and bandwidth dimensioning in overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1444–1455, 2002.

[18] T. Small, B. Li, S. Member, and B. Liang, "Outreach: Peer-to-peer topology construction towards minimized server bandwidth costs," in *in IEEE Journal on Selected Areas in Communications, Special Issue on Peer-to-Peer Communications and Applications, First Quarter*, 2007, pp. 35–45.

[19] K. Walkowiak, "Network Design Problem for P2P Multicasting," *International Network Optimization Conference INOC 2009*, Apr 2009.

[20] ——, "Survivability of P2P Multicasting," *7th International Workshop on the Design of Reliable Communication Networks, DRCN 2009*, pp. 92 – 99, Oct 2009.

[21] C. Wu and B. Li, "Optimal rate allocation in overlay content distribution," in *Networking*, 2007, pp. 678–690.

[22] ——, "On meeting p2p streaming bandwidth demand with limited supplies," in *In Proc. of the Fifteenth Annual SPIE/ACM International Conference on Multimedia Computing and Networking*, 2008.

[23] Y. Zhu and B. Li, "Overlay networks with linear capacity constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 2, pp. 159–173, 2008.

Design an Integration of Bee Hive into a Multi-Agent-based Resource Discovery Method in P2P Systems

Jun Yamasaki, Yoshikuni Harada, and Yasushi Kambayashi
Department of Computer and Information Engineering, Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun, Saitama 345-8501 Japan
{c1065434, c1015367}@cstu.nit.ac.jp and yasushi@nit.ac.jp

***Abstract*—One of the most important functions in P2P systems is the location of resources; it is generally hard to achieve due to the intrinsic nature of P2P, i.e., dynamic re-configuration of the network. We have proposed and implemented an efficient resource locating method in a pure P2P system based on a multiple agent system. All the resources as well as resource information are managed by cooperative multiple agents. In order to optimize the behaviors of cooperative multiple agents, we utilized the ant colony optimization (ACO) algorithm that assists mobile agents to migrate toward relatively resource-rich nodes. Even tough the ACO algorithm helped multiple search agents to find desired resources effectively while reducing communication traffic in the network, some research scientists have claimed that non-pheromone-based algorithm, such as honey bee algorithm, is significantly more efficient to find resources. Then efficient migration should be achieved through direct communication between bee agents in the dance floor instead of pheromone-based indirect communications. In this paper, we propose and discuss the possibility of integration of the bee-like agents into our resource discovery method to optimize the behaviors of the mobile multiple agents.**

Keywords-multi-agent system; mobile agents; resource discovery; swarm intelligence; honey bee algorithms

I. INTRODUCTION

As the Internet spreads throughout the world, it is used for a variety of human interactions. User interactions across various applications require software that exchanges resources and information within the network community. The traditional client-server model on computer networks barely accommodates the real-world situations such as the advent of video-streaming services. Intensive accesses on a server can easily create a bottleneck in a network. Peer-to-peer (P2P) systems can provide a solution to this problem. A P2P system consists of a number of decentralized distributed network nodes that are capable of sharing resources without central servers. Many applications such as IP-phone, contents delivery networks (CDN) and distributed computing adopt P2P technology into their base communication systems. A P2P system includes an overlay network where the nodes can interact and share resources with one another. Here, 'resources' means the variety of services that are provided by the network nodes.

One of the most important problems in P2P systems is the location of resources; it is one of the hardest mechanisms to implement. Napster avoids this problem by using a central server that provides indexing service [1]. Such a server, however, can be the most vulnerable point, where a failure can paralyzes the entire network. Therefore, P2P systems without any central server (pure P2P) are the area of active research in current P2P system developments. We have proposed and implemented an efficient resource location method based on a multi-agent system for a pure P2P system [2].

In this paper, we report our multi-agent system with an enhanced resource location mechanism. The efficiency is gained through a technique inspired by social insects like ants and honey bees. In order to optimize the behavior of cooperative multiple agents, we integrate the bee behaviors as well as DHT into mobile agents to migrate toward the desired nodes. In the previous paper, we discussed that the ant colony optimization algorithm provided quasi-optimally guidance for multiple search agents toward resource rich nodes and to assists them to find desired resources effectively while reducing communication traffic in the network. Efficient migration is achieved through an indirect communication that is typical of social insects, called *stigmergy*. When an agent finds a resource-rich node, it strengthens the path toward the node to gain efficiency. Strengthening of the route to a desirable node is achieved by pheromone applied by preceding agents that guides succeeding agents so that they can easily reach that node. This pheromone-based indirect guidance takes a certain amount of time to emerge the paths toward resource rich nodes, and even such paths are established, the guidance did not guarantee the suggested nodes have resources that the user really desired.

In order to ameliorate this problem, we are proposing and implementing a new type of resource discovery method that can be used in pure P2P systems. The method is based on yet another biologically inspired algorithm namely BeeHive [3]. BeeHive is a fault-tolerant routing algorithm and we have found it can be useful for P2P systems.

The structure of the balance of this paper is as follows. The second section describes the background. The third section describes the P2P system we are proposing. Static and mobile multiple agents work together to find network resources in the P2P system. We also discuss the honey bee behaviors for foraging that contribute the resource discovery. The fourth section describes the resource discovery algorithm that uses the multiple agents and a honey bee algorithm to find network resources in the P2P system. Finally, we sketch how the system is implemented using an overlay construction tool kit and a mobile agent construction framework and conclude our discussion in the fifth section.

II. BACKGROUND

The famous pure P2P system, Gnutella, employs message flooding for locating resources [1]. The advantage of such a system is its simplicity, but it is impractical for a large-scale network system, because flooding resource discovery messages alone can easily saturate entire networks. In order to solve this problem, the use of a distributed hash table (DHT) is proposed and used [4]. Even though DHT is one of the most promising methods and it certainly provides fast resource lookup (*O(log n)* computational complexity) for pure P2P systems, it is too rigid to process flexible and intelligent queries.

In order to ameliorate the problems in DHT-based systems, we have proposed a multiple-agent-based approach. One of the authors has engaged in a project where autonomous agents play major roles in an intelligent robot control system [5] [6]. The mobile agents in the project can bring the necessary functionalities and perform their tasks autonomously, and they have achieved reduction of communications as well as flexible behaviors. Thus, it is natural for us to employ not only static agents but also mobile agents in our P2P system in order to provide flexible search. The mobile agents are expected to reduce the quantity of query messages.

On the other hand, algorithms that are inspired by behaviors of social insects such as ants that communicate each other by an indirect communication called *stigmergy* are becoming popular [7]. Upon observing real ants' behaviors, Dorigo et al found that ants exchanged information by laying down a trail of a chemical substance (called *pheromone*) that is followed by other ants. They adopted this ant strategy, known as ant colony optimization (ACO), to solve various optimization problems such as the traveling salesman problem (TSP) [7].

Even tough ACO is effective; it is known that it takes some amount of time for the shortest path to emerge, and some doubts about its efficiency are posed [8]. It has been well known that honey bees have a remarkable sophistication of the communication capabilities those are comparative to ants. Von Frisch deciphered the enigma of bee's behaviors [9].

Honey bees' foraging behavior consists of two types that are recruitment and navigation. For navigation, bees use a strategy called path integration. They can compute their present location from their past trajectory continuously; hence they always know the direct route to their hive rather than retracing their outbound route, as ants do. For navigation, bees use a strategy called dance. Upon returning from a foraging trip, a bee communicates the distance, the direction and quality of flower site with its fellow foragers by performing waggle dances on the dance floor in its hive. The more zealously they dance, the more foraging bees are recruited to exploit the high quality flower site. Honey bees evaluate the quality of each discovered flower site and perform the waggle dance for the most promising flower site on the dance floor. As a result, high quality flower site are exploited quite intensively. Hence we abstract the dance floor into node management table by which the bee agent exchange resource information as BeeHive algorithm abstract the dance floor into routing tables [3].

BeeHive is a novel routing algorithm. For the algorithm, the network is organized into fixed parturitions called *foraging regions*, and two types of agents, *short distance bee agents* and *long distance bee agents* collect and disseminate routing information. Short distance bee agents only migrate in the foraging regions, thus they propagate routing information in the neighbors, while long distance bee agents can migrate to all the nodes in the network, thus they propagate routing information in the entire network [3]. In our approach, we model bee agents in resource discovery algorithm

III. THE P2P SYSTEM

The model of our system is a multi-agent system that consists of a set of cooperative static and mobile agents. All the resources as well as resource information are managed by cooperative multiple agents. They are: 1) information agents (IA), 2) two types of bee agents, namely long distance bee agents (LBA), and short distance bee agents (SBA), 3) node management agents (NA), and 4) DHT agents (DA). These five agents are the minimum configuration. LBAs and SBAs are the only mobile agents. IA's encapsulate all the interfaces between users and applications that utilize this P2P system so that all the other parts (agents) can be independent from any applications. We separate DA from NA, because only high performance nodes construct DHT. A Node with DA is called *super node*, and a super node dominates and administrates surrounding non-super nodes. We call this surrounding region a *foraging zone* and use it in our honey bee algorithm.

The P2P system is loosely partitioned into a number of foraging zones. Each foraging zone is constructed around one super node. Therefore the number of foraging zone is the same as the number of super nodes. Each node also has its specific foraging zone that consists of all nodes to which its short distance bee agents can reach. The union of these specific foraging zones is the foraging zone constructed around the super node. Each non-super node periodically dispatches a short distance bee agent in order to disseminate the resource information in its node management table. In order to prevent communication congestion, short distance bee agents only move in its specific foraging zone and have short lives. Therefore the foraging zone that partitions the P2P system (super-node's foraging zone) is not homogeneous; it is just an aggregate of many non-super nodes' foraging zones.

Only the super nodes can dispatch the long distance bee agents that can migrate to other super nodes, and propagate the node management table information as short distance bee agents do as well as collect node information in other foraging zones. The followings are the descriptions of each agent.

1) *Information Agent (IA):* Each node has a static information agent (IA) that manages resource information. IA periodically launches a short distance bee agent in order to collect and carries resource information in its specific foraging zone. IA also interacts with users. Each non-super node has resource information in the specific foraging zone. If the requested resource is in its foraging zone, IA can reply to the user immediately by consultation with the fellow node management agent (NA).
2) *Short Distance Bee Agent (SBA):* A Short distance bee agent circulates in its specific foraging zone to disseminate resource information. Each non-super node dispatches one short distance bee agent and it restricts its scout area in its specific forager zone so that it does not cause communication congestion. They are trying to uniform the node management tables in its specific foraging zone. Since the foraging zones for non-super nodes are overlapping each other, ultimately the resource information in the super node's foraging zone becomes relatively homogeneous.
3) *Long Distance Bee Agent (LBA):* If the user's requesting resource is not in the specific foraging zone of the non-super node that accepts the query, the non-super node's IA delegates the request to the super node, and then the super node creates a long distance bee agent to search the entire P2P system. There is small possibility that the requested resource is in a specific foraging zone of another non-super node in the same super node's foraging zone due to non-homogeneity. We are trying to measure empirically the rate of such anomaly. LBA also plays the role of messenger to convey DHT query messages.
4) *Node Management Agent (NA):* Each node has a static node management agent (NA) that has neighbor information collected dispatched short distance bee agent. NA has a table that contains the IP addresses of neighbors and resources that are hold by the neighbors.
5) *DHT agent (DA):* DHT agents (DA) construct DHT through cooperation with other DA's that reside on other super nodes. Only high-performance nodes have DA's so that we can construct pure P2P systems in a heterogeneous environment. Nodes with DA are called super nodes, and they construct a kind of super highway for the long distance bee agents (LBA). Though current implementation integrates the Chord [10] as the DHT algorithm, we can replace it with other algorithms just through replacing DA's. Due to the high churn property of pure P2P network, we can not expect DHT is ever complete. Therefore arriving LBA has incomplete information for the hash value, and DA can only provide incomplete information. Still the super node has nearly complete information about the non-super node under its dominance; it is therefore straightforward for LBA to find the requested resource in the foraging zone in which it arrives.

The majority of foragers exploit the food source in the closer vicinity of their hive while minority visit flower sites faraway from their hive [3] [9]. This observation inspires us to transform the search agents in the previous system into two groups namely short distance bee agents, and long distance bee agents. Short distance bee agents are not search agent in the current system. They collect and disseminate resource information in the neighborhood of source node, while long distance bee agents actually search for entire network to find the requested resource.

IV. Resource Discovery Algorithm

Our resource discovery method is based on mobile multiple agents based on BeeHive [3]. The network is loosely partitioned into several foraging zone. The reason why it is *loosely* partitioned is that the nature of P2P system is high frequency of joining and leaving of participating nodes (churning). Therefore rigid partitioning is impossible and unnecessary. On the other hand this property of loose provides robustness in the network. The foraging zone has one super node with DHT agent and surrounding non-super nodes that are reachable in some limited number hops. Each participating node launches one short distance bee agent (SBA) to collect and to carry resource information in the foraging zone. Upon arriving neighboring nodes, a SBA communicate with the residing node management agent (NA) to check the node management table, and if there are some missing information, the agent gives it to NA, and if it lacks some information, it receives it from NA. Therefore the NA's in one foraging zone share relatively uniform information of the resources the participating nodes in that foraging zone. Figure 1 shows the activity of SBA.

When a user requests that the information agent (IA) in the current node locate a resource, the user has to specify the lookup keywords and search terminating conditions such as the number of hops, duration time, and the number of found nodes. The user is also required to specify how IA should

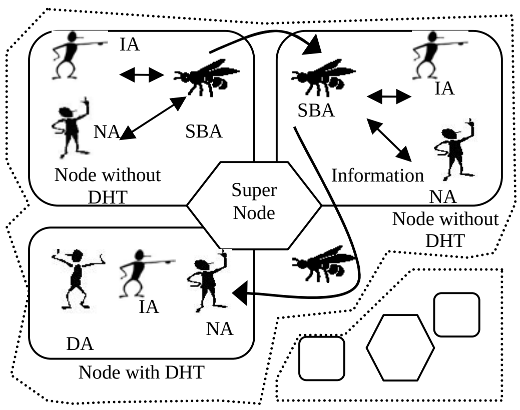


Figure 1. The activities of short distance bee agents to disseminate the resource information among the participating nodes.

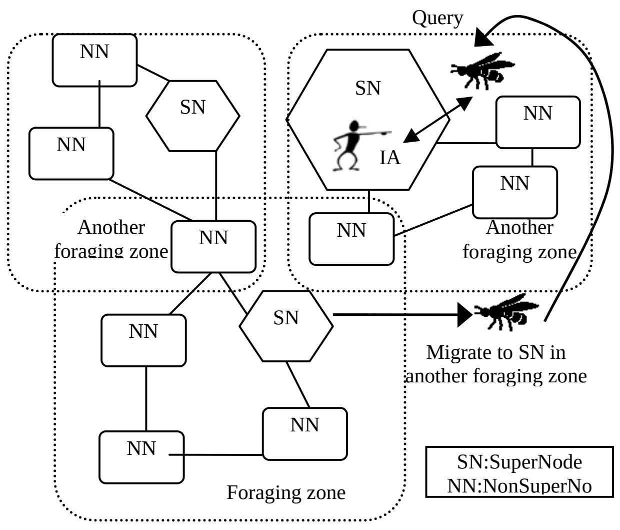


Figure 2. The activities of long distance bee agents to locvate a resorce. They communicate with super nodes.

behave when the dispatched LBA does not return due to some accidents. Figure 2 shows the activities of LBA that interact with other cooperative agents to locate desired resources. The resource discovery algorithm that the coordinated multiple agents perform is as follows:

1. IA consults NA whether the requested resource is in the local foraging zone. If it exists, IA reply so with the IP address stored in the node management table.
2. If the requested resource is not in the local foraging zone, IA delegates the request to the super node in the foraging zone.
3. The super node creates a long distance bee agent (LBA) for specific search.
4. The LBA requests the DHT agent (DA) to receive the IP address to which it migrates. The given address by DA may be inaccurate due to ambiguity of given information by the user. LBA's flexibility covers such incompleteness.
5. The LBA migrates to the selected super node in a different foraging zone.
6. The LBA then interacts with the IA in the arriving super node to find whether it has the requested resource. Then IA consults its NA.
7. If the foraging zone has the resource, the IA gives the IP address of the node to LBA, then goes to step 8, otherwise repeats at step 4.
8. The LBA checks the terminating condition, and if it is satisfied, migrates back to the original node where the user query was created.

V. Conclusion and Discussion

We are proposing and implementing a new type of resource discovery method that can be used in pure P2P systems. The method is based on yet another biologically inspired algorithm namely BeeHive that is a fault-tolerant routing algorithm. We have found it can be useful for P2P systems.

We are implementing our resource discovery methods by using Overlay Weaver [11] and Agent Space [12]. Overlay Weaver is an overlay construction tool kit for the Java language, and provides common API's for high-level services such as DHT and Multicast. This emulator can handle several thousand (virtual) nodes and records the number of produced messages and their duration time. Agent Space is a framework for constructing mobile agents. By using its library, the user can implement a mobile agent environment with the Java language. Since the integration of ACO into P2P system had provided a significant reduction of messages, we are expecting a similar or better effect on the system by the integration of honey bee behaviors.

References

[1] S. Saroiu, K. P. Gummadi, and S. D.Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," Multimedia Systems, vol. 9, 2003, pp. 170-184.

[2] Y. Kambayashi and Y. Harada, "A resource discovery method based on multi-agents in P2P systems," in Intelligent Agents in the Evolution of Web and Applications', N. T. Nguyen and L. C. Jain Eds. Studies in Computational Intelligence 167, Springer-Verlag, 2009, pp.113-135.

[3] H. F. Wedde, M. Farooq, and Y. Zhang, "BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior," Proc. International Workshop, ANTS, Ant Colony, Optimization and Swarm Intelligence, Lecture Notes in Computer Science 3172, 2004 pp. 83–94.

[4] A. S. Tanenbaum, Computer Networks, 4th ed., Prentice Hall, Upper Saddle River, 2002.

[5] Y. Kambayashi and M. Takimoto, "Higher-order mobile agents for controlling intelligent robots," International Journal of Intelligent Information Technologies, Vol. 1, No. 2, 2005, pp.28-42.

[6] M. Takimoto, M. Mizuno, M. Kurio, and Y. Kambayashi, "Saving Energy Consumption of Multi-Robots Using Higher-Order Mobile Agents," Proc. KES International Symp. on Agent and Multi-Agent Systems: Technologies and Applications, Lecture Notes in Artificial Intelligence 4496, Springer-Verlag, 2007, pp. 549-558.

[7] M. Dorigo and L.M.Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman," IEEE Transaction on Evolutionary Computation, vol. 1, no.1, 1996, pp. 53-66

[8] N. Lemmens, S. de Jong, K. Tuyls, and A. Nowé, "Bee behaviour in multi-agent systems: a bee foraging algorithm," in Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning, Lecture Notes in Computer Science 4865, Springer, 2008, pp. 145-156.

[9] K. von Frisch, The Dance Language and Orientation of Bees," Harvard University Press, Cambridge, Massachusetts, 1967.

[10] I. Stoica, R. Morris, and D. Karger, "Chord: a scalable peer-to-peer lookup service for Internet applications," Proc. ACM SIGCOMM Conference, 2001, pp. 149-160.

[11] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay weaver: an overlay construction toolkit," Proc. Symposium on Advanced Computing Systems and Infrastructures, 2006, 183-191, In Japanese.

[12] I. Satoh, "A mobile agent-based framework for active networks," Proc. IEEE System, Man and Cybernetics Conference, 1999, pp. 71-76