AP2PS 2012

The Fourth International Conference on Advances

in P2P Systems

ISBN: 978-1-61208-238-7

September 23-28, 2012

Barcelona, Spain

AP2PS 2012 Editors

Nick Antonopoulos, University of Derby, UK

Antonio Liotta, Eindhoven University of Technology, The Netherlands

Giuseppe Di Fatta, The University of Reading, UK

AP2PS 2012

Forward

The Fourth International Conference on Advances in Peer-to-Peer Systems (AP2PS 2012), held on September 23-28, 2012 in Barcelona, Spain, 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 2012 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 2012. 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 2012 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 2012 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 Barcelona provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

AP2PS 2012 Chairs:

Nick Antonopoulos, University of Derby, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Giuseppe Di Fatta, The University of Reading, UK

AP2PS 2012

Committee

AP2PS General Chairs

Nick Antonopoulos, University of Derby, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Giuseppe Di Fatta, The University of Reading, UK

AP2PS 2012 Technical Program Committee

Jemal H. Abawajy, Deakin University - Geelong, Australia
Nick Antonopoulos, University of Derby, UK
Ataul Bari, University of Western Ontario, Canada
Ahmad Tajuddin Bin Samsudin, TMR&D/ Telekom Malaysia Bhd., Malaysia
Ahmet Burak Can, Hacettepe University, Turkey
Dumitru Dan Burdescu, University of Craiova, Romania
Juan-Carlos Cano, Universidad Politécnica de Valencia, Spain
Charalampos Chelmis, University of Southern California - Los Angeles, USA
Chou Cheng-Fu, National Taiwan University, Taiwan
Carmela Comito, University of Calabria & ICAR-CNR, Italy
Noël Crespi, Telecom SudParis, France
Antonio Cuadra-Sanchez, Indra - Valladolid, Spain
Rosario De Chiara, Università degli Studi di Salerno - Fisciano, Italy
Carl James Debono, University of Malta, Malta
Giuseppe Di Fatta, The University of Reading, UK
Anne Doucet, Universite Pierre et Marie Curie - Paris, France
Nashwa Mamdouh El-Bendary, Arab Academy for Science, Technology, and Maritime Transport - Giza, Egypt
Mário Freire, University of Beira Interior, Portugal
Marco Furini, University of Modena and Reggio Emilia, Ital
Alex Galis, University College London, UK
Katja Gilly de la Sierra-Llamazares, Miguel Hernandez University, Spain
Takahiro Hara, University of Osaka, Japan
Kenneth Hawick, Massey University - Albany, New Zealand
Pilar Herrero, Universiad Politécnica de Madrid, Spain
Nicolas Hidalgo, Yahoo! Research Latin America, France
Eva Hladka, Masaryk University, Czech Republic
Eduardo Huedo Cuesta, Universidad Complutense de Madrid, Spain
Fabrice Huet, University of Nice, INRIA-CNRS - Sophia Antipolis, France
Hai Jiang, Arkansas State University, USA
Katerina Kabassi, TEI of Ionian Islands - Zakynthos Island, Greece
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Georgios Kambourakis, University of the Aegean - Samos, Greece

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Subscription Awareness Meets Rendezvous Routing

Fatemeh Rahimian, Sarunas Girdzijauskas, Amir H. Payberah, Seif Haridi

Swedish Institute of Computer Science
Stockholm, Sweden
email: {fatemeh,sarunas,amir,seif}@sics.se

***Abstract*—Publish/subscribe communication model has become an indispensable part of the Web 2.0 applications, such as social networks and news syndication. Although there exist a few systems that provide a genuinely scalable service for *topic-based* publish/subscribe model, the *content-based* solutions are still suffering from restricted subscription schemes, heavy and unbalanced load on the participating nodes, or excessively high matching complexity. We address these problems by constructing a distributed content-based publish/subscribe system by using only those components that are proven to be scalable and can withstand the workloads of massive sizes. Our publish/subscribe solution, *Vinifera*, requires only a bounded node degree and as we show, through simulations, it scales well to large network sizes and remains efficient under various subscription patterns and loads.**

***Keywords*-content-based pub/sub; load balancing; P2P;**

I. Introduction

The amount of data in the digital world that surrounds us is increasing very rapidly, thus, finding the relevant pieces of information becomes even more challenging. Publish/subscribe systems, which are now pivotal to many Web 2.0 applications, especially On-line Social Networks (OSNs) like Twitter, Facebook and Google+, leverage this problem by providing users with only the information they are actually interested in, e.g., a friend's status update, sport news, or a music playlist. As a result, when some new data is generated, the interested subscribers are notified. All these examples, which classify data into coarse-grained predefined categories or topics, are known as *topic-based* subscriptions. On the other hand, subscriptions that define a more fine-grained filter over the content of the generated data, are called *content-based*. These subscriptions usually consist of continuous ranges of values over several attributes that describe the content. For example, a user may be interested to get notified if the local temperature is below zero between 6am and 6pm.

The traditional model to provide a publish/subscribe service uses a central server or broker to maintain node subscriptions [2][6][14]. The published data is also sent to this central point and is matched against the existing subscriptions. Since subscription maintenance and matching are done centrally in this approach, the server could become a bottleneck as the number of users and subscriptions grow. Consequently, researchers have studied distributed publish/subscribe systems, in particular peer-to-peer (P2P) solutions, as an alternative design paradigm to the centralized model. Currently, a wide range of solutions are proposed for topic-based publish/subscribe over P2P overlays [4][11][16][32]. The topic-based solutions, however, can not be readily reused for the content-based model, due to the conceptual differences, i.e., discrete independent topics versus multiple continuous ranges over various attributes. Hence, a number of solutions have been proposed particularly for P2P content-based publish/subscribe [19][31][36][39], spanning from the designs based on unstructured gossip driven overlays to highly structured overlays with rigid event dissemination structures.

In particular, at one end of the design space we find a family of solutions that are subscription aware [19][36]. These solutions, partition the data space into subspaces that include each and every subscriber that is interested in that subspace. Published data is routed to the subspace that it belongs to, and is delivered to the subscribers in that subspace. As we describe in Section II, these systems perform well under simple workloads, but fail to deliver an efficient service to massive number of users with multi-dimensional subscriptions, mainly because they require unbounded number of connections per node. Moreover, despite having to maintain potentially numerous connections, these solutions can not provide an upper bound guarantee for average delivery path length.

At the other end of the design space lie DHT-based solutions, such as [31][37][39], that exploit a technique, known as *rendezvous routing* [5]. To enable rendezvous routing all the nodes and attributes are embedded in an identifier space, by taking a random identifier. Also, a distance function is introduced to make a greedy routing possible. The node with the closest, but higher, identifier to the attribute identifier is selected as the responsible node, i.e., *the rendezvous node*, for that attribute. Every subscriber node that performs a greedy routing (lookup) for an attribute identifier, therefore, ends up at the rendezvous node for that attribute. Next, the reverse routing path, i.e., from the rendezvous node to the subscriber node, is used for data dissemination. Consequently, the dissemination structure consists of a single tree per attribute, thus, often consisting of a handful of dissemination trees for the whole node population. Note, this is different from the multiple rendezvous based trees for topic-based subscription model (e.g., as in Scribe [10]), because, as opposed to the topic-based model, where a subscriber to a topic wants to receive all the events relevant to that topic, in the content-based model nodes that join an attribute tree, are often subscribed to only a subset of the possible values for that attribute. Hence, while this approach does not require an unbounded node degree, the constructed dissemination trees, which blindly deliver all the events to every node on the tree, are very inefficient.

 ISBN: 978-1-61208-238-7

In fact, the aforementioned state-of-the-art solutions are forced to choose a trade-off between scalability (bounded node degree and efficient routing) and overhead (both in volume and distribution), thus, fail to provide a genuinely distributed publish/subscribe service for Internet-scale applications. Such state of the design space inspired us to work on a solution which would not require unsuitable trade-offs and could retain all the desirable properties of a scalable system under any scenarios. As a result, we propose *Vinifera*, which to the best of our knowledge, is the first P2P content-based system that simultaneously fulfills all the fundamental requirements of a scalable distributed service.

Vinifera is empowered by a gossip-based topology management service and clusters the nodes with similar subscriptions. These clusters are later exploited to create efficient data dissemination structures. The same gossiping process, also, embeds a navigable small-world structure into the overlay after each node is assigned an identifier, selected from a globally known identifier space. This enables a distributed greedy distance minimizing routing algorithm to find short paths between any two nodes, which in turn, allows us to utilize the aforementioned rendezvous routing technique [5]. However, in contrast to other content-based publish/subscribe systems that construct a single delivery tree per attribute, thus, suffer from an unbalanced load and large traffic overhead, Vinifera constructs a forest per attribute, where the roots of the trees in the forest are the rendezvous nodes for the attribute values, thus, the load is distributed over all the participating nodes. These trees are dynamically constructed based on the user subscriptions.

Vinifera forest is constructed by utilizing an order preserving hash function [13], that maps each and every attribute domain to the node identifier space. For example, if an attribute has a domain [a, z], this range is mapped to the whole identifier space (say between 0 and 1) and every node in the overlay takes the responsibility for a part of this range that falls between itself and its predecessor in the identifier space. For example, let us assume nodes X, Y, and Z are responsible for ranges [a, d], [d, f] and [f, g] in the attribute domain, which are in turn, mapped to [0, 0.1], [0.1, 0.3], and [0.3, 0.4] in the identifier space. Then, nodes that subscribe to any value in the first range, route towards node X, while nodes that subscribe to the values in the second or third ranges, route towards nodes Y or Z, respectively (Note, a node can subscribe to a range, which contains multiple rendezvous nodes, for example, a subscription for the range [b, e] will be routed to both nodes X and Y, each being responsible for a part of the subscription). Hence, multiple small trees are constructed for event delivery for this attribute. We further enhance the load balancing in Vinifera, by a novel technique that enables it to deal with non-uniform subscriptions and publications. Thus, we ensure an evenly distributed load, even in case the data in some regions of the identifier space is more popular or is published more frequently than the other regions.

The resulting balanced load in Vinifera is of critical importance, not only because it implies fairness and a higher resource utilization, but also, and most importantly, because it enables the system to function under massive data publications and tolerate failures.

We run extensive simulations to evaluate multiple aspects of the performance, namely scalability, fault tolerance, load balancing and congestion control. We compare Vinifera to a baseline system, constructed based on a state-of-the-art solution, eFerry [39], which is a purely small-world overlay, oblivious to node subscriptions. Section II shows that this baseline solution is also conceptually equivalent to Ferry [39] and CAPS [31]. We show that, compared to the baseline system, Vinifera generates only one third of the traffic overhead, while the load is evenly distributed among the nodes and the delivery paths are up to four times shorter. We also show that Vinifera outperforms the baseline solution in the presence of churn, derived from real-world traces, and under intensive publications.

To summarize, our contribution is a genuinely scalable fault-tolerant multi-dimensional content-based publish/subscribe system with a bounded node degree requirement, a logarithmic worst-case bound on the delivery path length, and small and balanced load on the nodes. We achieve these properties by utilizing (i) an overlay topology that adapts to user subscriptions and exploits the similarity of subscriptions, in order to reduce the amount of traffic overhead that is generated in the network, (ii) constructing multiple efficient dissemination paths, instead of a rigid single tree structure, and (iii) a load balancing mechanism that enables the system to work under massive workloads.

II. Related work

As we briefly discussed in the introduction, there exist a number of solutions for building distributed content-based publish/subscribe systems [19][31][36][39]. In this section, we will have a closer look at these systems.

Meghdoot [19] exploits the idea of mapping each node subscription to a point in a $2d$ dimensional space, where d is the number of attributes/dimensions in the subscription scheme. Then, a CAN [33] overlay is utilized for routing the messages. Although matching events against subscriptions can be nicely done in Meghdoot, the routing is not efficient, due to the inherent inefficiencies in CAN overlay. Moreover, node degree could grow linearly with the number of attributes. The load on the nodes is also very unbalanced, depending on where in the CAN overlay the node is positioned.

Sub-2-Sub [36] takes a completely different approach. It clusters the subscription space into multiple subspaces, where each subspace includes all and only the nodes that are subscribed to the whole subspace. From then on, each subspace is treated like a topic in a topic-based model. A ring is constructed over each subspace for disseminating the events inside that subspace. The problems are two fold: firstly, it is difficult to construct the subspaces, if subscriptions are complex. In Hyper [38], which is a non P2P solution for content-based publish/subscribe, it is proved that solving such a problem is NP-complete. The existence of churn in the

 ISBN: 978-1-61208-238-7

P2P networks makes this problem even more challenging. Secondly, maintaining a ring per subspace implies that if a subscription is split into many subspaces, then the node has to join many overlays at the same time. Therefore, the node degree and maintenance cost could grow very large.

Ferry [39] is yet another approach to enable subscriptions over multiple attributes by employing a structured overlay network. Every node hashes the attribute names and sends its subscription to a rendezvous (RV) node, which is responsible for one of the generated hash values, preferably to the closest one. All the subscriptions are then maintained at the RV nodes. Upon an event publication, the event is delivered to all the RV nodes and will be routed towards the subscribers, accordingly. The strong point in Ferry is that the node degree is bounded regardless of the number of attributes in the subscription scheme. However, since the nodes subscribe for the hash of the attribute names, the routing structure solely depends on the subscription scheme in the system. For example, if there is only one attribute in the model, then one RV node and one delivery tree will exist. Therefore, the load on the nodes will be extremely unbalanced. The RV node not only receives all the published events in the system, but also has to match each and every event against all the existing subscriptions, before relaying the received events. An effort to solve the problems in Ferry is presented in eFerry [37]. The approach is to use different combinations of several attributes, for subscription registration. The proposed mechanisms exhibits desirable load balancing properties only for the publish/subscribe system with extremely large number of attributes, while is still inefficient for the usual systems with one or few attributes.

Another solution, that also requires a bounded node degree, is CAPS [31]. Similar to Ferry, CAPS uses the rendezvous model for subscription installation and event delivery. The main difference is that instead of a single key per attribute, it generates a set of hash values for each subscription, and installs a node subscription in multiple RV nodes in the overlay. The matching is then performed at those RV nodes and events are forwarded along the overlay links from the RV nodes to the subscribers. The problem in CAPS is that a subscription may be translated into too many keys to be installed, and could potentially result in a high traffic in the network. Moreover, the matching is performed centrally at the RV nodes and there is no mechanism for load balancing.

Pyracanthus [1] is another solution, which uses order preserving hashing to enable range queries for content-based publish/subscribe. However, it has a high maintenance cost as it stores a node subscription in all the rendezvous nodes across all the attributes. Moreover, event publication is very costly in Pyracanthus, since the publisher requires to collect all node subscriptions from the rendezvous nodes for all the attributes. It then selects the matching subscribers among the collected subscriptions, and forwards the event to them.

BlueDove [28] is yet another solution for multi-dimensional content-based subscriptions, which is particularly designed for well-engineered environments like clouds. In such environments, data center servers are connected by high speed local networks, packet loss rate is very low, and servers stay on-line for long periods of time.

There are also some related work on enabling range queries over P2P networks [3][8]. Mercury [8], for example, supports multi-dimensional rang-based searches, while it guarantees efficient routing and load balancing. Nevertheless, it does not provide all the necessary propertied of a fully-fledged publish/subscribe system, since it lacks the mechanisms for installing user subscriptions and event delivery. Moreover, the construction of the overlay in Mercury is oblivious to user interests.

Another set of related work is focused on how to filter data content in the overlay networks [12][17][27]. However, these works are orthogonal to our work and can be complementary to Vinifera. In particular, we can utilize [12] on top of our dissemination trees in order to better filter out the published content. The focus in Vinifera is on building a topology that exploits user subscriptions to enable efficient data dissemination structures.

III. Architectural Model

Vinifera is a multi-layer solution, where each lower layer provides a service to its upper layers. The architectural model of Vinifera (Figure 1(a)) consists of three main layers:

Random overlay. On the bottom layer we have a random network, which we construct by a gossip-based peer sampling service [22], similar to Cyclon [35], NewsCast [21], or Gozar [30]. This service is periodically executed by all the nodes and provides each node with a random sample of the existing nodes in the overlay. This layer also enables nodes to propagate control information that are required by the upper layer. In particular, every node piggybacks its subscription information on the gossip messages that it sends out.

Vinifera overlay. The topology of this layer is constructed by capturing the existing subscription correlation in the system and clustering similar nodes together, using the same gossiping protocol. Moreover, we make this topology navigable by embedding it into an identifier space and enriching it by Small-World links following Kleinberg's model [24]. The resulting topology allows efficient routing while preserving interest locality.

Vinifera Content management layer. This layer consists of several components that work together to manage the efficient delivery of the content. It exploits the navigability and the interest locality of the underlying layer by constructing a forest of dissemination structures based on RV routing. Because of the inherent interest locality property of the underlying overlay, each and every dissemination tree is expected to be small and efficient, involving mostly the interested nodes in the dissemination process. At the same time, Vinifera trees are expected to be shorter as compared to the state-of-the-art.

IV. Vinifera

A. Preliminaries

As we mentioned in the previous section, Vinifera is a gossip-based protocol, i.e., each nodes periodically exchanges

 ISBN: 978-1-61208-238-7

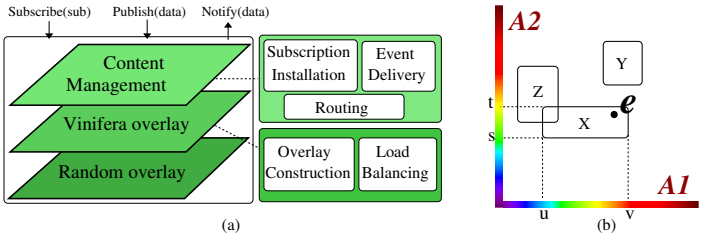


Fig. 1. (a) Vinifera architectural model (b) Vinifera Subscription model

some information with some other nodes in the overlay. This information includes the node *profile*, which contains the *node identifier* and the node *subscription*. The node identifier is selected uniformly at random from a globally known identifier space. The subscription scheme includes n attributes from A_1 to A_n, of any type, where attribute A_i could take values between $v_{i_{min}}$ and $v_{i_{max}}$. We map the range $[v_{i_{min}}, v_{i_{max}}]$ to the entire node identifier space, by applying an order preserving hash function (OPHF) [13] over the values that are valid for each attribute. An OPHF guarantees that *if* $v > u$ *then* *OPHF(v)* $>$ *OPHF(u)*. For the sake of simplicity, from now on we refer to the hashed value $OPHF(v)$, only as v. Each node subscribes in the system by introducing a number of constraints over a subset of attributes. A constraint specifies either an exact value (equality) or a range of values for an attribute, and a subscription S is the conjunction of all such constraints. Figure 1(b) shows a system with two attributes A_1 and A_2 and three subscriptions: X, Y, and Z. Subscriptions are shown by rectangles that specify the ranges over the two attributes. For example, subscription X is modeled as:

$$S_x : A_1 \in [u, v] \wedge A_2 \in [s, t]$$

A data item, or *event*, is a point in the attribute space, with exact values for all the attributes. An event *matches* a subscription, if each and every attribute value satisfies the corresponding constraint over that attribute. For example event e in Figure 1(b) matches subscription X, but it does not match subscriptions Y and Z.

B. Components

1) Overlay Construction: To enable nodes to select their *neighbors* (i.e., links or connections), based on their interest, identifier, or load, we employ a topology management protocol, inspired by T-Man [20], on top of the peer sampling service provided by the underlying random overlay. Each node p, maintains a *routing table*, i.e., a list of its neighbors, which it periodically exchanges with a neighbor, q, chosen uniformly at random among the existing neighbors in the routing table. Node p, then, merges its current routing table with the routing table of q, together with a fresh list of the nodes, provided by the underlying peer sampling service. The resulting list becomes the candidate neighbors list for p. Next, p selects a number of neighbors among the candidate neighbors and refreshes its current routing table. The same process will take place at node q.

Every vinifera node selects three types of links. First, each node maintains two links to connect to the nodes that are closest to it in the node identifier space, one in each direction. These links are called *ring links*, because eventually these links shape up a ring structure in the overlay. The ring topology guarantees the existence of a path between any two nodes, and ensures the lookup consistency in the overlay, which is later required.

Algorithm 1 Select Primary Attribute

```
1:  procedure SELECTPRIMARYATTRIBUTE
2:      for all A_i in self.S do                    ▷ S represents node subscription
3:          rank(A_i) ← 0                           ▷ initialize the rankings
4:      end for
5:      for all n in self.neighbors do
6:          for all A_i in self.S do
7:              if n.S.contains(A_i) then
8:                  selfC_i ← self.S.getC(A_i)      ▷ self constraint over A_i
9:                  nC_i ← n.S.getC(A_i)            ▷ neighbor constraint over A_i
10:                 if overlapping(selfC_i, nC_i) ≠ ∅ then
11:                     rank(A_i) =rank(A_i) + 1
12:                 end if                          ▷ increment the rank of the attribute
13:             end if
14:         end for
15:     end for
16:     A_p ← A_i where rank(A_i) is highest for all A_i ∈ self.S
17: end procedure
```

Next, to boost the routing efficiency, each node also selects some *small-world links*, based on the idea introduced by Kleinberg [24]. More precisely, node p selects node q as a small-world link, with a probability inversely proportional to the distance from p to q in the identifier space. It is proved that, having K_{sw} such neighbors enables a poly-logarithmic routing cost in the overlay ($O(\frac{1}{K_{sw}} log^2 N)$) [26].

Finally, links that are selected based on similarity of subscriptions are referred to as *friend links*. Every Vinifera node selects K_f friend links. These links connect together nodes with similar subscriptions. In a system with one attribute the similarity between two nodes p and q is captured by

$$Utility(p, q) = \frac{S_p \cap S_q}{S_p \cup S_q} \qquad \text{(Function I)} \qquad (1)$$

where, S_i contains the range(s) that node i has subscribed to. However, when there are more attributes, this approach is not readily applied. Instead, each node first selects one of the attributes, and then uses the mentioned utility function along that attribute only. As we will explain in Section IV-B4, when an event is published in a system with multiple attributes, multiple copies of the event are propagated in the overlay, one for each attribute. Therefore, to guarantee the event delivery, it is enough if a node is efficiently located in a cluster associated with only one of the attributes. The clusterization, i.e., the friend links selection, is completed in two steps:

- A node first examines the subscriptions of its candidate neighbors to select an attribute, across which it has more neighbors with overlapping ranges. We refer to this attribute as the *primary attribute* for the node. Algorithm 1 illustrates how the primary attribute is selected. The basic idea is that a node assigns a rank to each of the attributes in its own subscription. The rank of an attribute is calculated by counting the number of neighbors with an overlapping interest on that attribute (Lines 10, 11).

 ISBN: 978-1-61208-238-7

Algorithm 2 Range Query

```
1:  procedure LOOKUP(requester, lookupRequest)
2:      if requester ≠ self then
3:          installNeighborSubscription(requester, lookupRequest)
4:      end if
5:      RVNodes ← NULL
6:      if overlap(lookupRequest.range, [self, successor]) ≠ ∅ then
7:          RVNodes.add(successor)
8:      end if
9:      for all neighbor in self.neighbors do
10:         if lookupRequest.range.includes(neighbor) then
11:             RVNodes.add(neighbor)
12:         end if
13:     end for
14:     if RVNodes ≠ NULL then                                  ▷ Shower
15:         for all RV in RVNodes do
16:             send lookup(self, lookupRequest) to RV
17:         end for
18:     else                    ▷ Proceed with a lookup for the beginning of the range
19:         nextHop ← findNextHop(lookupRequest.range.from)
20:         send lookup(self, lookupRequest) to nextHop
21:     end if
22: end procedure
```

Finally, the attribute with the highest rank is selected as the primary attribute (Line 16).

- Next, the node uses the utility function (Function I) on the primary attribute and biases its neighbor selection towards selecting those nodes with higher rank as its friend links.

The combination of ring, small-world, and friend links results in a hybrid overlay, on top of which we build the data dissemination paths. We show, in the experiment section, that such a hybrid topology performs significantly better than a pure small-world overlay, as it not only reduces the unnecessary traffic in the network, but also improves the routing efficiency.

2) Routing: The basic routing or lookup in Vinifera is a greedy distance minimizing algorithm, i.e., at each step the lookup request is routed to a node which is closer to the destination. The destination of a lookup for value v is a node with the closest, but higher, identifier to v, which we refer to as the *rendezvous node (RV)* for v.

In Vinifera, nodes can not only route towards exact values, e.g., v, but also towards ranges of values, e.g., $[u, v]$ (Algorithm 2). To perform a range query, a node first applies an OPHF on the range boundary values. Then, a showering routing protocol [15] is executed, i.e., every node forwards the lookup request to as many nodes as it knows that fall into the range of the hashed values. Lines 15 to 17 in Algorithm 2 describe how the showering mechanism is performed. In case the node does not know any node in the requested range (line 18), it performs a simple greedy routing, i.e., it forwards the request to a node with closer, but not higher, identifier to the beginning of the range. The lookup ends at one or more consecutive RV nodes, each responsible for a part of the range.

3) Subscription Installation: Every node installs its subscription along the routing path to the rendezvous node(s) for the range over its primary attribute. We refer to this path (from the node itself to the rendezvous node(s)) as the *installation path*. Note that, we take advantage of the Vinifera overlay topology, by installing node subscriptions along the attribute for which they clustered more effectively, i.e., their

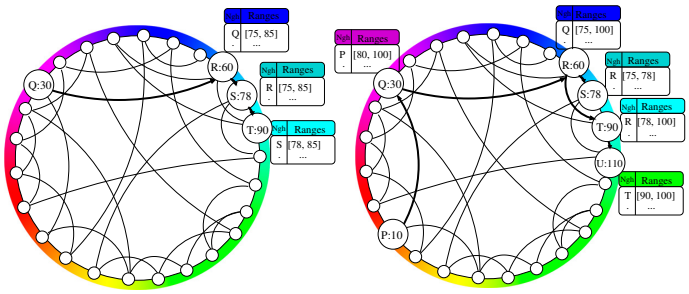


Fig. 2. (a) Subscription [75, 85] for node Q is installed. (b) Subscription [80, 100] for node P is installed (and aggregated with the previously installed subscription).

primary attribute. As a result, nodes that have the same primary attribute, and a similar range of interest over that attribute, will lookup for the similar rendezvous nodes. Since these nodes are very likely to be directly connected through friend links, they share most of their routing path towards the rendezvous node(s), with high probability. This is important, as it reduces the amount of traffic overhead transferred on the delivery paths.

Figure 2(a) illustrates the lookup process with a single attribute. Assume node Q wants to subscribe for the hashed values from 75 to 85. Among its neighbors, it selects node R, which has the closest, but not higher, node identifier to the requested range (Line 19 in Algorithm 2). The request is, therefore, sent from Q to R. Node R forwards the request to its neighbor S, which falls into the requested range (Line 10 in Algorithm 2). Node S takes the responsibility for the sub-range [75, 78], and also forwards a request for the remaining sub-range to node T (Line 6 in Algorithm 2). Consequently, the installation path from the subscriber node to the RV nodes is constructed (Path $Q \rightarrow R \rightarrow S \rightarrow T$ in Figure 2(a)).

Every node on the installation path maintains a table, called a *Content Routing Table* or *CRT* for short. The CRTs are populated when the queries are forwarded in the overlay. In our example, node R adds an entry to its CRT, for node Q requesting range [75-85], while node S registers range [75, 85] for node R. Finally node T registers a request from node S for the range [78-85]. In this example, we only have one attribute, and therefore CRTs, only include the requested range for that single attribute. If there are more attributes, each entry associates the requested range(s) with an attribute. Moreover, each entry of CRT contains the complete subscription(s) of the requesting node over all the attributes. This field will be used during event delivery process, described in Section IV-B4.
Putting all these together, an entry of a CRT is a tuple in form of $< Ngh, Attr, Ranges, Subscriptions >$, where *Ngh* indicates the requesting neighbor, *Attr* is the requested attribute, *Ranges* are the interested ranges over the requested attribute, and *Subscriptions* are the subscription requests, containing all the attributes, which are received through the requesting neighbor.

The subscription installation process is further equipped with an aggregation technique. That is, whenever possible, a node aggregates all the requests it receives from the same neighbor on the same attribute. This is usually referred to as subscription subsumption or covering in the literature. For example, in Figure 2(b) node P appears in the system and

 ISBN: 978-1-61208-238-7

subscribes for the range [80-100]. The closest neighbor of P to the requested range is node Q. So P sends a request to Q. As a result, Q installs this request in its CRT, and forwards it further to node R. Since node R previously had an entry for Q in its CRT (for the range [75-85]), it aggregates the two requests and modifies the entry for Q to range [75-100]. Now R knows two nodes, S and T, in the requested range. So it showers the request to both of them, by sending a request for range [75-78] to S and the remaining part of the range to T. When this new request is further forwarded in the overlay, nodes S and T similarly update their CRTs with the range [75, 78] and the aggregated range [78-100], respectively. Then, node T forwards the request further to node U, which is also a rendezvous node for a part of the requested range.

As mentioned previously, thanks to the employed clustering technique, nodes with similar subscriptions on the same primary attribute are grouped together. When these nodes install their subscriptions, they initiate a routing towards the same or close-by rendezvous nodes. Therefore, the installation tree is mostly shared between such nodes, thus, the requests can be effectively aggregated. This results in smaller CRTs, as well as less traffic overhead in the overlay. Smaller CRTs not only reduce the maintenance cost of the trees, but also simplify the matching process.

Note that, the subscription installation is a periodic process, and therefore, if a node fails or changes its subscription, it does not send any more request for the previously requested range, and therefore, the already installed rows in CRTs further ahead, can de-aggregate or be removed completely. This ensures that the quality of CRTs are constantly maintained.

4) Event Delivery: Any node in Vinifera can publish events. As mentioned previously, an event is a piece of data that has an exact value for each attribute. Therefore, in a system with n attributes, an event is associated with n rendezvous nodes, one for each attribute. When a node publishes an event $e\{v_1, v_2, .., v_n\}$, it sends one copy of the event to each of the n relevant rendezvous nodes, i.e., $RV(v_1)$, $RV(v_2)$, .., $RV(v_n)$, which are responsible for the values assigned to each of the attributes. This is done by initiating a simple rendezvous routing for each attribute. Then, n *delivery trees* for the event are constructed on the fly, by following the links on the reverse installation paths from the rendezvous nodes to the subscriber nodes, using the node CRTs. Note that each matching subscriber is registered in only one of the delivery trees, i.e., the one that corresponds to its primary attribute. So, it does not receive redundant messages from multiple trees.

The delivery trees are constructed as follows. Each rendezvous node, matches the event against the subscriptions that are registered in its CRT, and sends the event only to the neighbors with matching subscriptions. Note that, the matching is performed on the whole registered subscriptions, that is, if the event does not match the registered subscription of a node on any of the attributes (primary or not), the event will not be forwarded further to that node. Likewise, every node on the path performs such a matching process and forwards the event further if it matches the registered request, until it reaches the interested subscribers. By this approach, the matching process will be carried out as the event goes down to the subscribers, while every node maintains partial information about the other nodes. In essence, we distribute the load of matching events against subscriptions between the nodes that are on the installation path. At every step, those branches that are not interested in the event are pruned and the event is forwarded only down the paths that hold some interested node(s).

5) Load Balancing: Due to the prevalent skewed subscription patterns in the real world, the use of an OPHF inevitably results in a non-uniform identifier distribution and thus, an unbalanced load on the nodes. More precisely, some regions in the identifier space might be very popular with huge number of corresponding events, while some other regions might not be popular at all. So, the nodes who happen to fall into the popular regions may have to deal with huge number of requests. For example, if an attribute in the subscription scheme is temperature in a city, then the published events are most likely in the range $[-10, +30]$, probably a few around this range, and almost no event in less than -30 or over $+50$. Hence, node that have an identifier between OPHF(-10) and OPHF$(+30)$ are likely to be highly loaded, while the rest of the nodes are under loaded.

To alleviate this problem, we utilize an idea, inspired by [23], for adapting the node identifiers to the existing load in the network. The idea is that Vinifera nodes may change their identifiers along the ring, while their connections remains intact. In other word, nodes change their identifier to an identifier between themselves and their successor. The new identifier is assigned to the node to halve the load on the successor. Since nodes do not change their neighbors upon change of the identifier, they can easily inform their neighbors of the new identifier, when they send the next gossip message to the neighbors.

For our system, we define a measure of load as follows. Every node counts the number of events that it receives as a rendezvous node, without having any interest in them. Whenever the node sends its gossip message to its predecessor, it piggybacks its current load on the message. In order to prevent perturbation of the node identifiers, we define a threshold β for load imbalance between a node and its successor. When the difference of load between the two nodes exceeds the threshold β, the proceeding node changes its identifier to a value closer to its successor. Then the two nodes update their load to the average of their current loads. We show in the evaluation section that by employing this mechanism, Vinifera nodes can adapt to even very skewed user subscriptions.

C. Maintenance

In general, P2P networks are subject to churn, i.e., nodes join or leave the system continuously and concurrently, and network capacity changes. Therefore, it is essential to design P2P systems that tolerate failures. When a node fails in Vinifera, all the layers take the required actions to deal with that failure. The random overlay at the bottom, which is

 ISBN: 978-1-61208-238-7

a gossip-based peer sampling service, is inherently failure tolerant. More details on fault tolerance in such protocols can be found in [22].

In the Vinifera overlay layer, failure handling is done similarly to the random overlay. As we described in Section IV-A, nodes periodically send their profile to their neighbors. This profile message, therefore, serves as a heartbeat message and enables the nodes to detect the failure of their direct neighbors. When a node fails, its direct neighbors that detect the failure, remove the entries for the failed neighbor from their partial view. When these nodes exchange their views with other nodes in the system, the contacted nodes will also receive the updated information and remove the stale entries, accordingly. Therefore, the information about the failed nodes, is propagated in the overlay by taking advantage of the ongoing gossiping protocol.

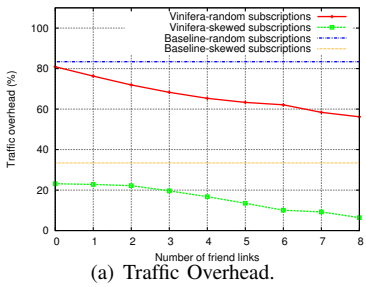
In the content management overlay, we need to ensure that CRTs are always updated and do not contain stale entries, i.e., when a node fails, we should remove its subscription from all the CRTs along the installation path. Note that, every entry in the CRT has an expiry timestamp. If a node does not receive a new subscription request from its neighbor, it will automatically remove the request from its CRT. Normally, in each gossip round requests are resent and therefore maintained in the CRTs. When a node fails, however, the first node met on the installation path, which has been directly connected to the failed node, detects the failure and removes the subscription of the failed node from its CRT. Therefore, it never again forwards the requests for that subscription, i.e., the next node on the relay path, will not receive the request for that subscription, thus, removes the entry from its CRT. Note that, if an entry in the CRT is the result of an aggregation, i.e., a node A received two requests from the same neighbor B for some overlapping ranges, and part of this aggregated range concerned a node C that is failed now, only the part that corresponded to C will be de-aggregated, as node B will send only one of the requests to node A, thereafter.

V. Evaluation

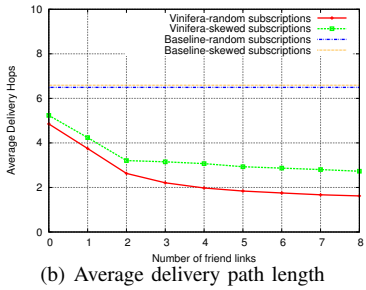
We implemented Vinifera in Peersim [29], a discrete event simulator for P2P applications. Through extensive simulations with a hybrid of cycle based and event based models, we evaluated the performance of Vinifera, while comparing it against a baseline system, inspired by the state-of-the-art techniques such as Ferry [39], eFerry [37], and CAPS [31]. That is, the baseline system is a pure small-world overlay, thus, requires a bounded node degree and guarantees a bounded routing time, but it is oblivious to node subscription, with no load balancing mechanism. In the lack of real-world traces, we synthetically generated user subscriptions, using a Zipf-like distribution over the attribute space [9]. Unless otherwise mentioned, the network size is 1000.

A. Topology Configuration

In this experiment, we investigate the choice of values for parameters K_{sw}, and K_f, which define the number of small-world links and friend links, respectively. The total number of links per node is set to 10, among which two are dedicated to ring links and the rest are used for small-world and friend links, i.e., $K_{sw} + K_f = 8$. Figures 3(a) and 3(b) show the traffic overhead and average delivery path length of Vinifera and the baseline system for two subscription models. Since the baseline system does not have any friend links, its topology does not change across the X-axis. Zero friend link in Vinifera generates a pure small-world topology, which is oblivious to node subscriptions, just like the baseline system. Hence, at this point both systems have the same topology and that is why with random subscriptions, the improvement in the traffic overhead in Vinifera is negligible. However at the same point, the average path length in Vinifera is decreased. This is due to the existence of many short delivery trees in Vinifera, as opposed to a single long delivery tree in Ferry. With skewed subscriptions, we can also improve the traffic overhead from over 32% to 22%. By adding more friend links, we take advantage of the subscription similarities and significantly improve both metrics at the same time. With 8 friends and skewed subscriptions, for example, the traffic overhead drops from 32% to less than 10%, while the average path length is decreased from around 7 to 3. This proves the huge potential of exploiting user subscription correlations, common in real-world scenarios. In the rest of our experiments, we set K_{sw} to 0 and K_f to $log(N) - 2$, where N is the number of nodes in the network. However, for the applications that require an upper bound guarantee on the number of delivery hops, we can add more small-world links.



(a) Traffic Overhead.



(b) Average delivery path length

Fig. 3. Performance with variable number of friend links

B. Scalability

To measure the scalability of Vinifera, we performed experiments with different number of nodes, as well as, different number of attributes. Figure 4(a) shows the traffic overhead of both systems. The performance of both systems is almost the same for different network sizes. However, the traffic overhead in the baseline system is more than 80% for random subscrip-

 ISBN: 978-1-61208-238-7

tions, whereas it is reduced to 60% in Vinifera. Note that random subscriptions bring up the worst case scenario, for nodes can not effectively benefit from our clustering technique, due to the lack of correlation between user subscriptions. However, it is shown that a significant subscription correlation exists in real-world application [25][34]. When the user subscriptions are skewed, the traffic overhead in the baseline system drops to nearly half, while Vinifera reduces the traffic to almost one sixth of that of the random subscriptions. This shows that our data dissemination overlay is remarkably benefiting from the utilized clustering technique. Figure 4(b) shows the average delivery path length of both systems in terms of hop counts. Here again, the number of hops in Vinifera is nearly one third of that of the baseline system. However, the path length is slightly bigger with skewed subscriptions, because the overlay topology is clustered. With the random subscription model, however, the overlay better resembles a random network, thus, we observe a reduced path length.

Next, we observe the behavior of both systems when the subscription model includes more attributes. We have designed Vinifera to work with any dimensionality. Because of the lack of real-world traces we had to decide on the number of dimensions ourselves while carrying out the experiments. To be consistent (as well as being able to easily compare) with the existing state-of-the-art solutions we used most of the parameters (including dimensionality) from the related work papers. As we observed, most of the related work had reported results with 2, 3, or 5 attributes. We also show the results with up to 5 attributes. In higher dimensions, Vinifera still exhibits consistent results. Nevertheless, with our randomly generated events, the fraction of matching events drops sharply, thus, the measured values become insignificant, making the system hard to evaluate. This is due to a phenomenon, known as *the curse of dimensionality* in the literature [7]. However, since in real life the generated data is not uniformly spread in the data space and there exists a correlation between user subscriptions and the generated data [9][25][34], Vinifera is expected to function effectively in higher dimensions under realistic workloads.

Figure 4(c) shows when the subscription model is random, the traffic overhead of the baseline system remains at around 80%. This overhead starts from around 58% in Vinifera, but increases in higher dimensions, because in a random subscription scheme there exists very little similarities to be exploited, and therefore, the primary attribute is practically a random attribute for each node. As a result, the installation paths, and thus, the delivery trees are scattered in the overlay and nodes can not effectively cooperate in data dissemination. However, this overhead is still less than that of the baseline system. With the skewed subscription model, both systems behave significantly better. The overall improvement is again due to the skewed event publication in the system. However, the improvement in the baseline system with more attributes is because instead of having one single tree, more delivery trees are constructed, one for each attribute. Each node joins one of these trees, as a subscriber, and does not receive the events that are forwarded on other trees, unless it is a relay node in those trees. However, in Vinifera nodes with similar subscriptions are grouped together, and the delivery tree is shared between these nodes. Thus, the overall number of uninterested node on the delivery trees is reduced, thus, the traffic overhead drops to nearly one third of that of the baseline system.

The average delivery path length of the two systems are shown in Figure 4(d). The baseline system delivers the events slightly faster when there are more attributes. However, the average delivery path length in Vinifera is still by far better than the baseline system, even with 5 attributes in the subscription model, thanks to the utilized clustering technique. As soon as an event reaches a cluster of nodes with matching subscriptions, it is propagated inside that cluster very quickly.

We conclude that although both systems can accommodate any number of attributes in the subscription scheme, Vinifera exhibits a significantly better performance than the baseline system, specially in the presence of skewed subscriptions.

C. Load Balancing

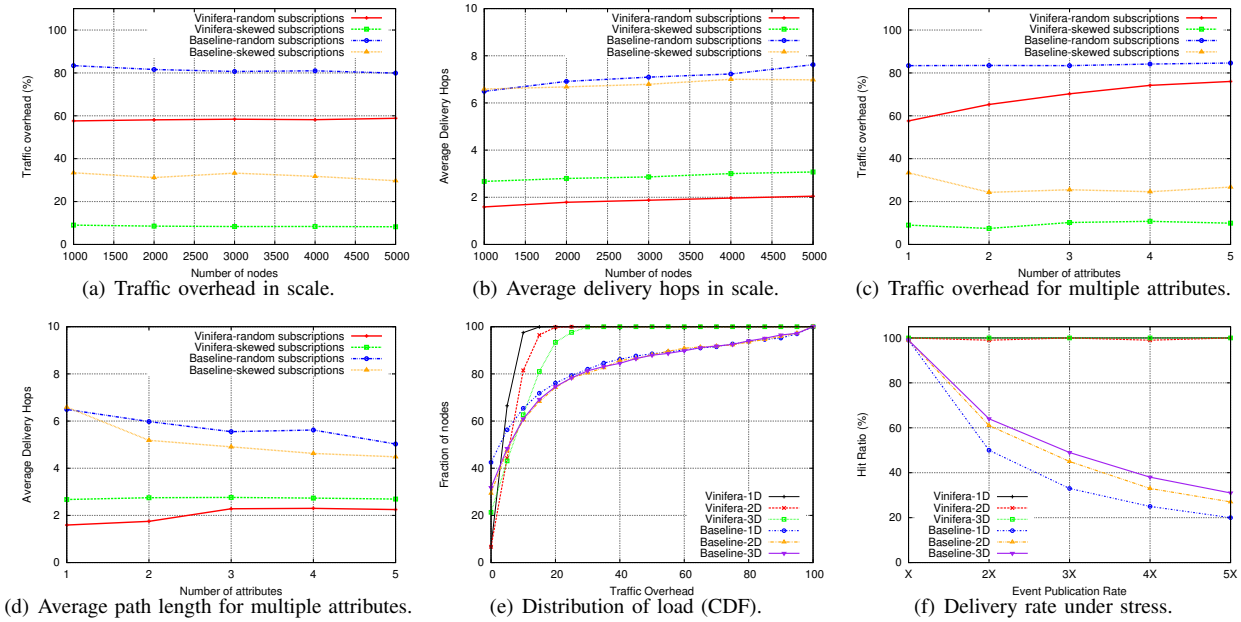
To explore how the load is distributed among the node in Vinifera versus the baseline system, we plot the cumulative distribution of load on the nodes, for 1, 2, and 3 attributes, and report the results in Figure 4(e). Although with more attributes, the load distribution is slightly degraded in Vinifera, it is still significantly better than the baseline system and the load on any Vinifera node never exceeds 30%. More precisely, over 95% of the nodes has a load less than 20%, even with 3 attributes, whereas, 10% of the baseline system nodes suffer from over 60% load in the system, while nearly 40% of the nodes have zero load. Figure 4(e) shows that the load in the baseline system is extremely unbalanced. This is because nodes up on the delivery tree are highly stressed, while leaf nodes are just receiving the service for free. There are nodes with even nearly 100% load (the rendezvous nodes), which can significantly harm the performance of the system as soon as they stop functioning correctly, due to congestion or failure. This problem prevents the baseline system to work under real-life scenarios where node and network failures are inevitable, while Vinifera can still function without having any imminent bottleneck.

D. Workload

In this section, we examine if the systems can function under different workloads, i.e., under different event publication rates. To model the congestion, we assume that every node in the system can handle a bounded number of messages, X, in every round, and if it receives more events it will simply drop them. Then, we increase the number of events that are published in the system to up to five times X.

Figure 4(f) shows that the hit ratio in the baseline system significantly drops as soon as the publication rate passes the X threshold. Whereas, Vinifera survives even under high event publication rates. This is due to the fact that the baseline system relies on very few nodes to propagate the event in the system (only the intermediary nodes in the delivery tree). Therefore, under a high publication rate, those nodes become

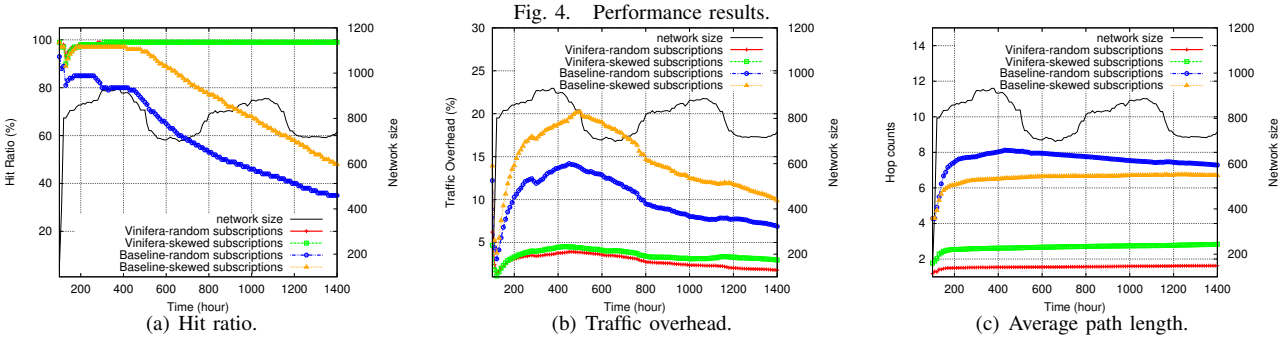
 ISBN: 978-1-61208-238-7



(a) Traffic overhead in scale. (b) Average delivery hops in scale. (c) Traffic overhead for multiple attributes.

(d) Average path length for multiple attributes. (e) Distribution of load (CDF). (f) Delivery rate under stress.

Fig. 4. Performance results.



(a) Hit ratio. (b) Traffic overhead. (c) Average path length.

Fig. 5. Performance under Skype churn trace.

highly overloaded and start dropping the messages. When a node in the tree drops a message, all its descendant nodes fail to receive the message. On the other hand, in Vinifera load is almost evenly distributed among the nodes. Thus, the nodes do not have to drop the messages due to the excessive load.

E. Fault Tolerance

To evaluate the performance of Vinifera in the presence of failures, we used real-world churn traces [18], that were obtained by monitoring a set of 4000 nodes participating in the Skype superpeer network for one month beginning September 12, 2005. In Figures 5(a) to 5(c) the x-axis shows the time, while the y-axis on the right shows the network size. The black solid line in the three graphs shows how the network size changes over time. Figure 5(a) shows the hit ratio of the two systems with random and skewed subscriptions. Although the hit ratio of Vinifera slightly decreases in the flash crowds, i.e., around time 100 h., when a large number of nodes join the system concurrently, the system recovers quickly and the hit ratio goes back to and remain at 100%, even in the presence of further joins and failures. In contrast, the hit ratio in the baseline system is highly affected by churn, due to the fragile structure of a single delivery tree. When this tree is broken, the baseline system can not repair it quickly enough to catch up with further event deliveries. When no more node joins or fails, the baseline system is potentially able to repair the dissemination tree. However, as we see in this real-world trace, this hardly happens.

Figure 5(b) shows the traffic overhead in both systems. The traffic overhead in Vinifera is one forth compared to the baseline system for both random and skewed subscriptions. Note that, the reduced traffic overhead in the baseline system is because it fails to deliver the events to all the nodes. We also observe, in Figure 5(c), that Vinifera is takes a four times shorter delivery path compared to the baseline system, in the presence of churn. Here again, we should take into account that in the baseline system some nodes are not receiving the events, and the measured values for the baseline system only include the nodes that received the events.

VI. Conclusion

We introduced Vinifera, a P2P content-based publish/subscribe system that enables users to subscribe for the information they are willing to receive, without having to

 ISBN: 978-1-61208-238-7

rely on any single authority or central server. We employed a gossip-based technique to construct a topology that not only resembles a small-world network, but also connects the nodes with similar subscriptions together. On top of this hybrid overlay, we utilized a rendezvous routing mechanism to propagate node subscriptions in the overlay. Together with an order preserving hashing technique and an efficient showering algorithm we enabled range queries, and at the same time, we employed a load balancing technique to deal with the potential non-uniform user subscriptions. The combination of all these techniques are seamlessly integrated within a single gossiping layer, thus keeping Vinifera simple, lightweight and robust.

Our hybrid publish/subscribe system exhibited superior performance against the state-of-the-art techniques, effectively without the need to trade-off or degrade any important properties of the system. The overlay topology autonomously adapts to user subscriptions and is highly resilient to the dynamism in the network. The generated traffic overhead and the average delivery path length are simultaneously kept low, while only a bounded node degree is required and no global knowledge at any point is assumed.

REFERENCES

[1] I. Aekaterinidis and P. Triantafillou. Pyracanthus: A scalable solution for dht-independent content-based publish/subscribe data networks. *Information Systems*, 36(3):655–674, 2011.

[2] M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra. Matching events in a content-based subscription system. In *Proc. of PODC'99*, pages 53–61. ACM, 1999.

[3] A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *Proc. of P2P'02*, pages 33–40. IEEE, 2002.

[4] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni. Tera: topic-based event routing for peer-to-peer architectures. In *Proc. of DEBS'07*, pages 2–13. ACM, 2007.

[5] R. Baldoni and A. Virgillito. Distributed event routing in publish/subscribe communication systems: a survey, 2005.

[6] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom, and D. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proc. of ICDCS'99*, pages 262–272. IEEE, 1999.

[7] R. Bellman and R. Kalaba. On adaptive control processes. *IRE Trans. Auto. Cont.*, 4(2):1–9, 1959.

[8] A. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *Proc. of SIGCOMM'04*, pages 353–366. ACM, 2004.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of INFOCOM'99*, volume 1, pages 126–134. IEEE, 1999.

[10] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, 2002.

[11] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proc. of DEBS'07*, pages 14–25. ACM, 2007.

[12] Y. Diao, S. Rizvi, and M. Franklin. Towards an internet-scale xml dissemination service. In *Proc. of VLDB'04*, pages 612–623, 2004.

[13] E. Fox, Q. Chen, A. Daoud, and L. Heath. Order-preserving minimal perfect hash functions and information retrieval. *ACM Trans. Info. Syst.*, 9(3):281–308, 1991.

[14] G. Fox and S. Pallickara. An event service to support grid computational environments. *Conc. and Comput.: Prac. and Exp.*, 14(13-15):1097–1127, 2002.

[15] Š. Girdzijauskas. *Designing peer-to-peer overlays: a small-world perspective*. PhD thesis, École Polytechnique Fédérale De Lausanne, 2009.

[16] S. Girdzijauskas, G. Chockler, Y. Vigfusson, Y. Tock, and R. Melamed. Magnet: practical subscription clustering for internet-scale publish/subscribe. In *Proc. of DEBS'10*, pages 172–183. ACM, 2010.

[17] E. Grummt. Fine-grained parallel xml filtering for content-based publish/subscribe systems. In *Proc. of DEBS'11*, pages 219–228. ACM, 2011.

[18] S. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer voip system. In *Proc. of IPTPS'06*, 2006.

[19] A. Gupta, O. Sahin, D. Agrawal, and A. Abbadi. Meghdoot: content-based publish/subscribe over p2p networks. In *Proc. of Middleware'04*, pages 254–273. Springer, 2004.

[20] M. Jelasity and O. Babaoglu. T-man: gossip-based overlay topology management. In *Proc. of the ESOA'05*, pages 1–15. Springer, 2006.

[21] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proc. of ICDCS'04*, pages 102–109. IEEE, 2004.

[22] M. Jelasity, S. Voulgaris, R. Guerraoui, A. Kermarrec, and M. Van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3), 2007.

[23] D. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proc. of SPAA'04*, pages 36–43. ACM, 2004.

[24] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proc. of STOC'00*, pages 163–170. ACM, 2000.

[25] H. Liu, V. Ramasubramanian, and E. Sirer. Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews. In *Proc. of IMC'05*, pages 3–3. USENIX, 2005.

[26] G. Manku, M. Bawa, and P. Raghavan. Symphony: distributed hashing in a small world. In *Proc. of USITS'03*, pages 10–10. USENIX, 2003.

[27] I. Miliaraki, Z. Kaoudi, and M. Koubarakis. Xml data dissemination using automata on top of structured overlay networks. In *Proc. of WWW'08*, pages 865–874. ACM, 2008.

[28] L. Ming, Y. Fan, K. Minkyong, H. C., and H. L. A scalable and elastic publish/subscribe service. In *Proc. of IPDPS'11*, pages 1254–1265. IEEE, 2011.

[29] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *Proc. of P2P'09*, pages 99–100. IEEE, 2009.

[30] A. Payberah, J. Dowling, and S. Haridi. Gozar: Nat-friendly peer sampling with one-hop distributed nat traversal. In *Proc. of DAIS'11*, pages 1–14. Springer, 2011.

[31] J. Pujol-Ahullo, P. Garcia-Lopez, and A. Gomez-Skarmeta. Towards a lightweight content-based publish/subscribe services for peer-to-peer systems. *Int. Grid Util. Comput.*, 1(3):239–251, 2009.

[32] F. Rahimian, S. Girdzijauskas, A. Payberah, and S. Haridi. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe. In *Proc. of IPDPS'11*, pages 746–757. IEEE, 2011.

[33] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of SIGCOMM'01*, pages 161–172. ACM, 2001.

[34] Y. Tock, N. N., H. A., and G. G. Hierarchical clustering of message flows in a multicast data dissemination system. In *Proc. of PDCS'05*, pages 320–326, 2005.

[35] S. Voulgaris, D. Gavidia, and M. Van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Net. and Syst. Manag.*, 13(2):197–217, 2005.

[36] S. Voulgaris, E. Riviere, A. Kermarrec, and M. Van Steen. Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In *Proc. of IPTPS'06*, 2006.

[37] X. Yang, Y. Zhu, and Y. Hu. Scalable content-based publish/subscribe services over structured peer-to-peer networks. In *Proc. of PDP'07*, pages 171–178. IEEE, 2007.

[38] R. Zhang and Y. Hu. Hyper: A hybrid approach to efficient content-based publish/subscribe. In *Proc. of ICDCS'05*, pages 427–436. IEEE, 2005.

[39] Y. Zhu and Y. Hu. Ferry: An architecture for content-based publish/subscribe services on p2p networks. In *Proc. of ICPP'05*, pages 427–434. IEEE, 2005.

 ISBN: 978-1-61208-238-7

What are the Services of an Information-centric Network, and Who Provides Them?

Anders Eriksson, Börje Ohlman, Karl-Åke Persson
Ericsson Research
Ericsson
Stockholm, Sweden
{*Anders.E.Eriksson, Borje.Ohlman, Karl-Ake.Persson*}*@ericsson.com*

***Abstract*—Various Information-centric Network (ICN) services have been proposed in the literature, such as content distribution, publish-subscribe, event notification, and search. Such services have traditionally been described as separate from one another, and little attention has been given to the issue of a common ICN architecture within which they all can interact efficiently. In this paper, we describe how information-centric services can interact within one architectural framework to provide a rich ICN service offering related to Information Objects, such as content and data objects. We give examples of how the architecture can support various application domains, for example content distribution, machine-to-machine communication, and interactive and live streaming applications. We also propose the business role of an ICN Service Provider, which adds value by composing a service offering of a variety of ICN services. The contribution of the paper is to highlight the need for efficient interaction between multiple ICN services in order to provide an attractive and complete ICN service offering, to describe an architecture for such interaction, and to identify the business role of an ICN Service Provider.**

***Keywords*-Information-centric; architecture; service model.**

I. INTRODUCTION

Information-Centric Networks (ICN) is a relatively new research field. There are currently a number of approaches being developed, e.g., NetInf [1], CCN/NDN [7], PSIRP [8]. There have also been some attempts to define common mechanisms and characteristics that should be generic to ICN [2]. In this paper, we make an attempt to define ICN at a mechanism-independent service level. The core idea of ICN is to move the focus from devices and network entities to the Information Objects (IOs) themselves, i.e., content, data files, RFID tags, web pages, sensor data, application instances, etc. ICN should thus be defined by how we can interact with these IOs, i.e., by defining the services needed to interact with them.

The service offered by an IP network is primarily store-and-forwarding of IP packets. What are, then, the services offered by an Information-Centric Network? Judging from the papers published on this topic summarized in [2], an ICN primarily offers an optimized transfer service for IOs, similar to that of http. Using this service, clients can retrieve named IOs from storage or streaming servers in an anycast fashion. Some network architectures also offer a subscribe and event notification service. Other services needed for a full-blown ICN, like an advertisement service, are not, as far as we have seen, described in the proposed approaches.

Much of the ICN research focus on an http-like transfer service which results in a research agenda with topics and mechanisms well-known from TCP/IP research, such as routing, congestion control, mobility management, etc., although with an information-centric touch. In this paper, we argue that there is more to Information-Centric Networks than this. An ICN allows for a significantly richer service offering in addition to that of transfer of IOs. A key ICN service is search for IOs and metadata associated with IOs. Traditional Internet search engines offer unpredictable performance in terms of when or whether a published resource will be reachable via search. An ICN search service should offer timely reachability of published IOs. One of the things we discuss in this paper is how the interaction between an ICN Search Service and other ICN services, such as Event Notification, Advertisement, and Storage allows for a more predictable Search Service.

When discussing ICN services, it is of course relevant to describe business roles, service providers, and interdomain interfaces. The paper outlines a business model for ICN using open and non-proprietary interdomain interfaces.

The remainder of this paper is organized as follows: Section II describes a horizontal application-independent ICN infrastructure. Section III describes the semantics of the interaction between ICN services on the one hand, and the interaction between these services and the ICN Publishers and Subscribers on the other hand. Section IV describes how this interaction model applies to different use cases. Section V describes business roles and interdomain interfaces between ICN services and service providers, as well as ICN metadata relevant for these interfaces. Section VI discusses related work. Section VII draws conclusions and proposes next steps in the development of an ICN service model.

 ISBN: 978-1-61208-238-7

II. Application-Independent Horizontal ICN Services

As mentioned in the introduction, a number of ICN approaches are currently being proposed. Which of them will be successful only the future can tell. Potentially they could prove to be successful in different parts of the network, e.g. the core network, highly dynamic edge networks or sensor networks. Irrespective how this will turn out we see that they will offer a very similar set of services. By defining these services, including standardized Application Programming Interfaces (APIs) and interdomain interfaces we can provide a stable environment for application developers. Likewise, an interdomain interface which hides domain-internal ICN mechanisms will enable interoperation between ICN domains employing different domain-internal mechanisms.

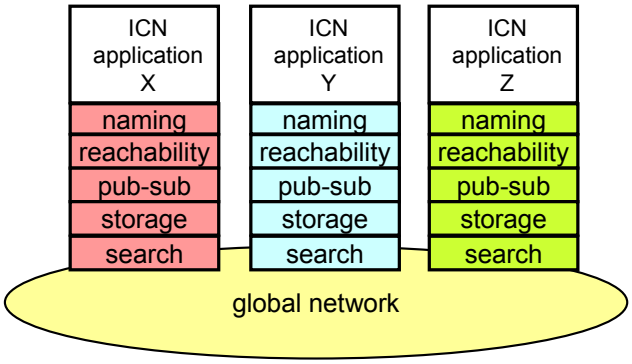


Figure 1. Information Object lock-in per vertical application.

Figure 1 shows several vertically integrated ICNs, each dedicated for a single application and operating as an overlay on a global network. Such vertically integrated ICNs have been employed for content distribution, social networks, peer-to-peer communication, and machine-to-machine communication, e.g. Skype, Spotify and BitTorrent. Each of the overlay ICNs has a separate name space, and a separate name resolution system. This has several implications:

- Each vertically integrated ICN requires dedicated systems for management of information object names, name resolution, search, and publish-subscribe.
- The IOs are locked-in within one vertical ICN and cannot be reached from other verticals. As a consequence, it is hard to develop an application that communicates with IOs belonging to different verticals. The limited access to such IOs diminishes their usefulness.
- The barrier of entry for a new vertical ICN with global coverage is rather high, since it requires deployment of new systems for management of IO names, name resolution, reachability, search, and publish-subscribe.

The lock-in effect on IOs ultimately results in a lock-in effect on end-users, which become tied to specific applications, since only those applications can communicate with the favourite IOs of the end-users. Likewise, the renaming effort needed to port a large number of IOs from one vertical to another may be prohibitive.

The issues associated with vertically integrated ICNs can be overcome by means of a horizontal infrastructure for management of IO names, name resolution, reachability, publish-subscribe, storage, and search as shown in Figure 2. The infrastructure is horizontal in the sense that it can serve as a global and open platform for a multitude of applications.

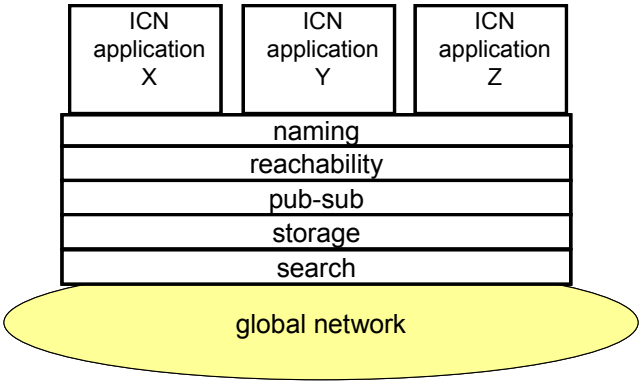


Figure 2. Horizontal infrastructure allowing multiple applications to communicate with Information Objects in order to avoid object lock-in.

With the horizontal approach, each IO has an application-independent name which can be resolved into a globally unique locator by a Name Resolution Service (NRS). Any application can thereby reach any IO. This allows for a variety of applications communicating with a specific IO, which adds to the usefulness of the IO. In addition, an application can communicate with a variety of IOs of different types. This facilitates innovation of novel applications which cannot be supported by rigid vertical ICNs dedicated for a specific application and set of IO types.

To allow for internetworking across a variety of access technologies, the horizontal infrastructure should not only be independent of applications, but also of access technologies. Moreover, the application-independent horizontal ICN infrastructure should be designed so that it can handle arbitrary types of networked objects, such as data files, RFID tags, and persons. This will enable novel applications that can interact seamlessly with such objects.

III. Interaction model for ICN services

We claim that the emergence of a monolithic ICN service, in analogy with the best-effort store-and-forward service of an IP network, is unlikely due to the multitude of ICN use cases. Instead, we propose that an ICN will provide a set of services. Figure 3 shows a set of services provided by an ICN, and the semantics of the interaction between these services, as well as with the Publishers and Subscribers of the ICN. Interaction between the services allows for a more

 ISBN: 978-1-61208-238-7

powerful ICN than a set of non-interacating services. The set of services amd their interactions are selected to handle a range of use cases, see section IV.

In the model shown in Figure 3, the Publisher and the Subscriber are end users of the ICN. A Publisher publishes IOs, which can subsequently be accessed by a Subscriber. The various services shown in the figure support different aspects of this interaction between the Publisher and the Subscriber as will be described below.

The IO has a URI and can be accessed over the ICN. The Publisher assigns a URI to the IO. To inform a potential Subscriber about the URI and the metadata associated with the IO, the Publisher uses an Advertisement Service. Subscribers use the Advertisement Service to find information about an IO before it is eventually published via the Storage Service or via streaming. The Advertisement Service can use any ad-hoc mechanism external to the ICN, such as random web pages or e-mail lists. This is currently a rather common way of advertising IOs. Alternatively, the Advertisement Service can be included in the ICN, and be accessed via an ICN API.

When a Publisher assigns an URI to an IO, this URI may not include any information about the location of the IO. For example, if the IO is nomadic or mobile, location information may not be included in the URI. In this case, there is a need for a NRS to resolve a location-independent URI into a URL, which includes information about the locator of the IO. The Publisher registers the binding between the URI and the URL in the NRS. The Publisher may also register metadata associated with an IO in the NRS.

When a Publisher advertises an IO with its URI and metadata via the Advertisement Service, it also registers the IO with the Event Notification Service. The Subscriber can then subscribe to the IO via the Event Notification Service to receive notifications about events related to the IO, for example when it is published via the Storage Service or via streaming.

A Storage Service allows for access to stored IOs. When a Publisher publishes an IO, it also stores the IO in the Storage Service. The Storage Service returns a URL for the stored IO to the Publisher, which can use this URL when registering a URI-URL binding in the NRS. The Storage Service may be associated with the Publisher, or it may be a third-party service.

There is a Processing Service associated with the Storage Service. The Processing Service processes stored IOs, and thus produce new IOs, which in turn are stored by the Storage Service. The Processing Service notifies the Publisher when a new IO has been stored, and the Publisher can decide to publish this IO and register it in the Name Resolution Service in order to make it available for Subscribers.

A Search Service can keep track of all advertisements, and decide to subscribe to all subscription names learnt via the Advertisement Service. The Search Service will then be notified about all events related to all URIs registered with the Event Notification Service.

In addition, the Search Service can index information stored in the Storage Service, and also index URI-metadata bindings in the Name Resolution Service. This type of Search Service will be able to immediately detect when IOs are published, or when they change state. This allows for shorter and more deterministic response times compared to traditional Internet search systems based on web crawling.

A key aspect of ICN is the use of caching. Caches in the ICN store IOs which are retrieved from the Storage Service. This reduces access times and network load.

The abstract service model described above can be instantiated in various fashions, with different sets of protocols and mechanisms to invoke the services. Of course, the implementation details of an instantiation, including the selection of protocols, depend on the concrete set of use cases at hand.

IV. ICN Use Case Examples

In this section, we present some examples of how our service model for ICN can be applied to specific use cases.

A. Sensor networks

An example of an information flow in an information-centric sensor network is shown in Figure 4. Sensors collect data from Entities of Interest (EoI), for example the temperature at a specific location. Each Machine Device (MD) has a Publisher function, a network interface, and hosts one or many sensors. The sensor data are stored and processed by a Storage and Processing (S&P) function in the MD. As a result, the sensor data is transformed into an IO which is stored in the MD, and also published by the Publisher function in the MD.

Dedicated S&P nodes in the information-centric network subscribe to IOs published by MDs, or published by other S&P nodes. The S&P nodes process these IOs and produce new IOs. For data sets, e.g. sensor data, S&P nodes can create new IOs with data for, e.g., average, sum, or time series. Finally, application nodes subscribe to IOs published by S&P nodes or an MD.

A motivation for separate S&P nodes is that sufficient processing capacity may not be available in an MD, so processing and storage needs to be off-loaded. It is also possible that the application nodes at the top of Figure 4 are lightweight devices which off-load processing and storage to separate S&P nodes.

B. Content distribution

In the interaction model shown in Figure 3, content distribution starts with the Publisher storing a content IO in the Storage Service, and then registering the IO and its metadata with the Advertisement Service and the Name Resolution

 ISBN: 978-1-61208-238-7

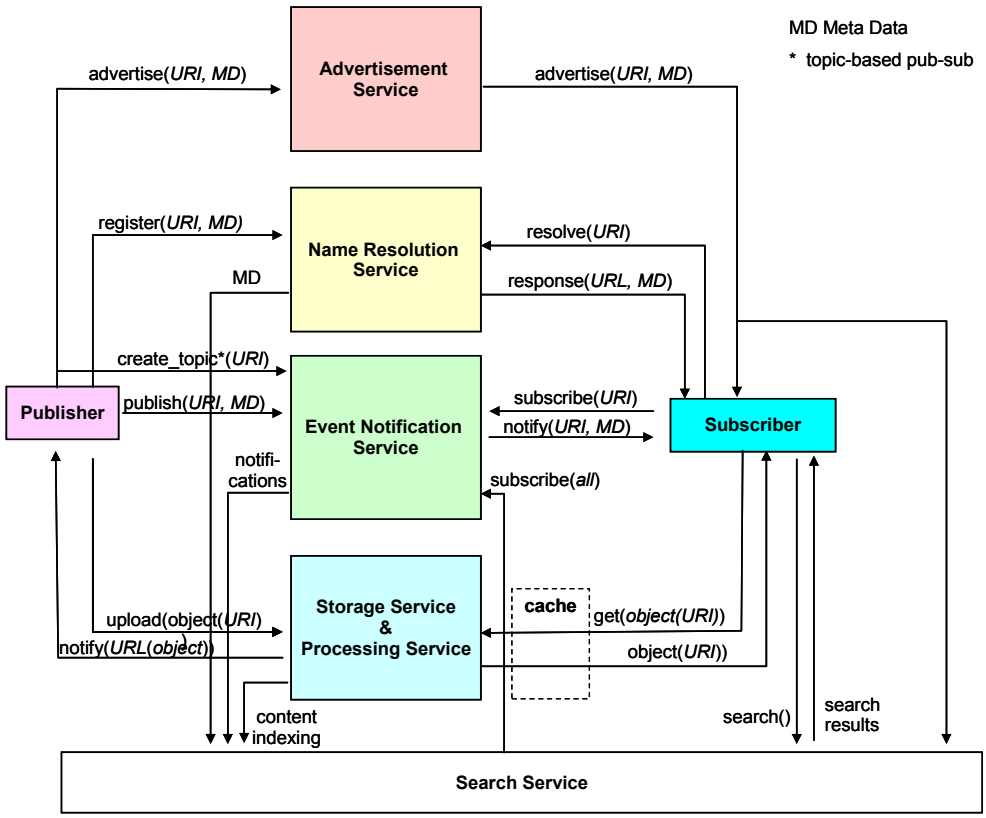


Figure 3. Service semantics of an ICN and interaction between services.

Service. This will trigger the Advertisement Service or the Storage Service to register the IO and its metadata with the Search Service; see the figure. The Subscriber can now search for the IO using the Search Service, which returns a URL for the IO. This URL enables the Subscriber to either retrieve the IO directly from the Storage Service, or make further queries for the IO using the Advertisement Service.

Alternatively, the Publisher can advertise the IO with the Advertisement Service and the Event Notification Service well before it is stored in the Storage Service. This will trigger the Advertisement Service to register the IO and its metadata with the Search Service. The Subscriber can now find the Advertisement for the IO using the Search Service, and then subscribe to the IO via the Event Notification Service. When the Publisher eventually stores the IO in the Storage Service, this will trigger the Event Notification Service to send event notifications to inform all Subscribers of the IO about the URL of the stored IO.

C. Interactive and live streaming use case

The first thing to happen in the interaction model is that the Publisher advertises the upcoming service (e.g. telephony call or live football game) through the Advertisement Service. Next, interested Subscribers will subscribe through the Event Notification Service. As content becomes available, the Publisher will make it available by uploading it to the Storage Service, register it with the NRS, and publish it through the Event Notification Service.

Many interactive services have more of a push than a pull character. This is in conflict with one of the key features of ICN, the receiver-oriented operation, which gives the receivers better control of the traffic flows and helps to curb DoS attacks. In the interaction model shown in Figure 3, this can be dealt with in a receiver-oriented fashion by handling a media stream as an IO which is requested by making a subscription for the IO ID that represents the stream. The sender can then push the stream by sending updates to that subscription, which will result in an event notification reaching the Subscriber.

V. Interdomain Operation and Federation of ICN Services

In ICN, what constitutes a domain will differ very much from context to context. The minimal ICN interdomain operation might only consist of two ICN nodes that belong to different domains. At the other end of the spectrum, very large ICN providers could offer services like name resolution or storage on a global scale in a federated fashion. The service model in Figure 3 can be applied regardless of the size or number of ICN providers.

 ISBN: 978-1-61208-238-7

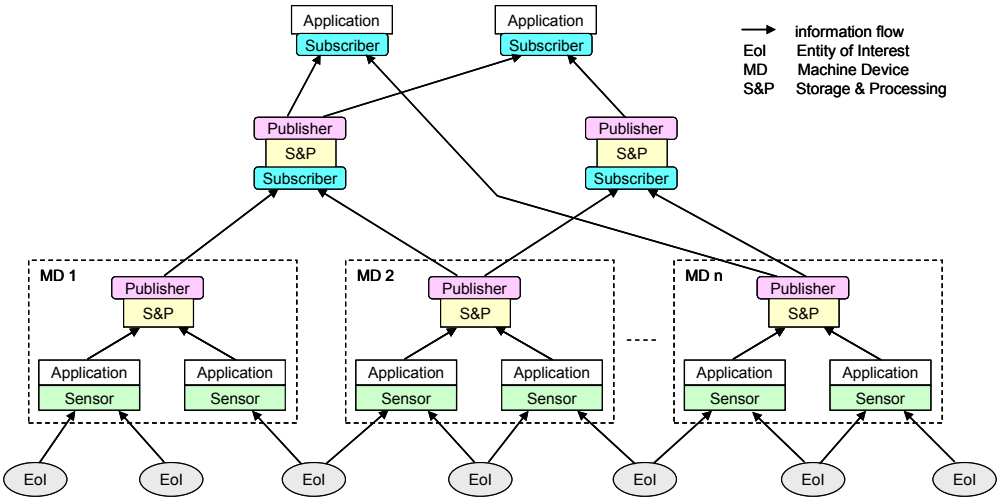


Figure 4. Information flow in an ICN for sensor data

A. Business Roles and Interdomain Interfaces

Figure 3 describes a set of services and the interactions between these. The figure is agnostic with regard to the business roles of the actors implementing these services. In this section we discuss the issue of mapping these services to business roles and actors, as well as describing interdomain interfaces between different business roles and actors. We focus the discussion on open and non-proprietary interfaces between the different ICN services shown in the figure, as well as between the actors implementing the services. The intention of such interfaces is to foster competition in the evolution of various ICN services, and to enable an efficient horizontal infrastructure with open interfaces as shown in Figure 2.

The interdomain interfaces between actors implementing a Name Resolution Service is strongly dependent on the architecture of the NRS. For example, a global and federated NRS based on a distributed hash table architecture will result in each actor managing resource records belonging to other actors. By contrast, a global NRS based on a DNS-like architecture will result in each actor managing its own resource records. This will lead to very different interdomain interfaces between the actors for the two types of architectures with regard to signaling semantics for update and retrieval of resource records, as well as for authentication and authorization.

An ICN Search Service can be implemented by traditional search providers, each offering a separate service having a proprietary internal architecture, but offering open and non-proprietary interdomain interfaces to allow for interaction with other ICN services as shown in Figure 3. Alternatively, a federated Search Service can be considered, where different search providers join in a search federation and use open and non-proprietary interdomain interfaces between each other, as well as between the federated Search Service and other ICN Services.

As illustrated in Figure 3, an ICN Search Service has interdomain interfaces to the NRS, the Advertisement Service, the Event Notification Service, and the Storage & Processing Service.

Contrary to Name Resolution Services or Search Services, there are no well established global Event Notification Services. There are several reasons for this, among which are scalability and real-time performance issues. While waiting for the ultimate event notification service, there may be a need to use a variety of Event Notification Service types, depending on the use case at hand.

There is a possibility for new business roles which provide the ICN services in the model shown in Figure 3. An Access ICN Service Provider adds value by composing a service offering of a variety of ICN services by acting as a broker between end users and Interdomain ICN Service Providers.

B. Metadata

Metadata is in ICN used for a number of purposes including access and security policies. Information contained in metadata is an important part of the interdomain interface. This can be compared with the metadata for Content Delivery Networks Interconnection [4]. In traditional host-centric networking, this type of information is normally stored on the same node-complex that stores the data the policies relate to. In ICN this type of information needs to be tied to the individual IOs (including all copies). How metadata is best stored is still a research issue under investigation. Alternatives include having metadata stored as records in the NRS, or stored together with the IO itself in the IO storage.

Metadata can include security data such as signatures, delegation of rights, and the public key of the publisher. When it comes to ensuring the confidentiality of an IO,

 ISBN: 978-1-61208-238-7

ICN can offer a wide range of alternative security levels, starting with fully public IOs which anyone can retrieve if they only know the IO ID, or if they just search for IOs that are matching a set of attributes. A bit more of confidentiality can be achieved by only storing objects in private NRSs. The highest levels of confidentiality is achieved by cryptographic means. By using strong crypto mechanisms and advanced key distribution schemes, very high level of confidentiality can be achieved. There are two main drawbacks of these cryptographic methods. Firstly, the complexity of administrating key distribution schemes, which is well-known. Secondly, when the highest level of access control with individual keys are used for each user and each object, one of the main advantaged of ICN is forfeited, since caching will be useless due to the individually encrypted IOs.

VI. Related work

As mentioned in section I, Information-centric Networking puts retrieval of information objects in focus instead of traditional packet routing and forwarding between nodes. A key idea in ICN is the naming of the information objects with globally unique names, in contrast to traditional networking, where the focus is on naming of nodes and interfaces. ICN can also, in general, be said to be receiver driven, i.e., to initiate the transfer of an information object the receiver requests an object by name from the network. In traditional networks the sender pushes data into the network and the network forwards it towards the receiver.

The ICN approach allows for a network architecture that can exploit a multitude of alternative infrastructures and mechanisms to retrieve the desired information objects. These include, but are not limited to, locally cached copies at nearby nodes, use of multiple access networks in parallel, and information retrieval over pure broadcast networks such as FM, TV, or satellite networks. Also, networks with intermittent connectivity and data mule networks can be utilized in this information-centric paradigm.

A number of information-centric architectures have been described in the literature, such as DONA [9] , NetInf [1], CCN/NDN [7] and PSIRP [8]. These architectures are based on a set of concepts and mechanisms for the retrieval of information objects, such as secure naming of IOs, routing and forwarding of IOs, transport of IOs, and an API between the application layer and the Information-centric Network layer. Key differentiators between these approaches, are if they use hierarchical names or flat names; if they route directly on names or if they rely on some type of name resolution service to map names of information objects into network locations. For a more extensive overview of different ICN architectures and their detailed mechanisms we refer the reader to the IEEE ICN survey paper [2]. That paper as well as [6] are trying to decompose and find the basic components of a generic ICN architecture with the focus on one specific network service, i.e., retrieval of information objects. By contrast, in this paper we propose a set of services and interactions between services to enable a more powerful ICN.

Another important aspect of ICN is the service model of the API, from the application perspective ICN looks more like a database than a traditional end-to-end communication network. In ICN the application makes a request for an information object to the network, not a specific host, by name. The requested information object is then delivered to the application through the API, without any information from which host it was retrieved. This makes ICN resemble Remote Invocation Method (RMI) systems. It can therefore be interesting to discuss to what extent distributed system technologies such as CORBA [11], Jini [3], or web services could support the ICN services and interactions shown in Figure 3. Both CORBA and Jini natively support name resolution and event notification mechanisms, which are key in the ICN architecture. However, these mechanisms do not necessarily scale to the number of IOs that are envisaged for an ICN, i.e. at least a few orders of magnitude above the number of objects accessible in the current Internet.

In addition, the ICN architecture should support highly dynamic network topologies where IOs, hosts, and networks can be mobile. CORBA and current web service technologies do not have native support for such dynamic topologies. Even though Jini is designed to handle object mobility, this support is not sufficient to handle highly dynamic network topologies with strict requirements on short handover latencies [10]. By contrast, mobility and multihoming mechanisms have been developed to handle such mobility use cases for ICN [5].

VII. Conclusions

In this paper, we argued that a key feature of ICN is the support of a variety of use cases by providing a set of services such as name resolution, reachability, storage, event notification, search, etc. These services make minimal assumptions about the applications, and can therefore be used by a multitude of application types. To describe these services and the interaction between them, there is a need for a model that is independent of the mechanisms used to implement the services. We have made a first attempt to describe such a service model. Having such a service model will make it possible to define APIs and interdomain interfaces to allow interoperation between ICN approaches with different domain-internal mechanisms, as well as providing a mechanism-independent environment for application developers. There is a possibility for a new business role which provides the ICN services in the model, i.e., an ICN Service Provider.

References

[1] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, M. Marchisio, I. Marsh, B. Ohlman, K. Pentikousis, R. Rembarz, O. Strandberg, and V. Vercellone. Design considerations for a network

 ISBN: 978-1-61208-238-7

of information. In *Proceedings of ReArch'08: Re-Architecting the Internet*, Madrid, Spain, Dec. 9, 2008.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7):26–36, July 2012.

[3] Apache. Jini Specification. Available online at http://river.apache.org/about.html, retrieved: August, 2012.

[4] M. Caulfield and K. Leung. Content Distribution Network Interconnection (CDNI) Core Metadata. draft-caulfield-cdni-metadata-core-00, October 2011.

[5] A. Eriksson and B. Ohlman. Scalable object-to-object communication over a dynamic global network. In *Proceedings of Future Network and MobileSummit 2010*, June 2010.

[6] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, and J. Wilcox. Information-Centric Networking: Seeing the Forest for the Trees. In *Tenth ACM workshop on Hot Topics in Networks (HotNets-X)*, Cambridge, MA, USA, Nov. 2011.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[8] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: Line Speed Publish/Subscribe Internetworking. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 195–206, New York, NY, USA, 2009. ACM.

[9] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of SIGCOMM'07*, Kyoto, Japan, Aug. 27-31, 2007.

[10] J. Newmarch. *Jan Newmarch's Guide to Jini Technologies*. 2006. Available online at http://jan.newmarch.name/java/jini/tutorial/Jini.xml, retrieved: August, 2012.

[11] OMG. CORBA Specification. Available online at http://www.omg.org/spec/CORBA/index.htm, retrieved: August, 2012.

 ISBN: 978-1-61208-238-7

Using Real-Time Backward Traffic Difference Estimation for Energy Conservation in Wireless Devices

Constandinos X. Mavromoustakis and Christos D. Dimitriou
Department of Computer Science,
University of Nicosia
46 Makedonitissas Avenue, P.O.Box 24005
1700 Nicosia, Cyprus
mavromoustakis.c@unic.ac.cy; dimitriou.cd@gmail.com

George Mastorakis
Department of Commerce and Marketing Technological Educational Institute of Crete, Ierapetra,
Crete, Greece
g.mastorakis@emark.teicrete.gr

***Abstract*—This work proposes a scheme for sharing resources using the opportunistic networking paradigm whereas, it enables Energy Conservation (EC) by allocating Real-Time Traffic-based dissimilar Sleep/Wake schedules to wireless devices. The scheme considers the resource sharing process which, according to the duration of the traffic through the associated channel, it impacts the Sleep-time duration of the node. The paper examines the traffic's backward difference in order to define the next Sleep-time duration for each node. The proposed scheme is being evaluated through Real-Time implementation by using dynamically moving MICA2dot wireless nodes which, are exchanging resources in a Mobile Peer-to-Peer manner. Various performance metrics were considered for the thorough evaluation of the proposed scheme. Results have shown the scheme's efficiency for enabling EC and provide a schematic way for minimizing the Energy Consumption in Real-Time, in contrast to the delay variations between packets; whereas the proposed scheme aims at maximizing the efficiency of resource exchange between mobile peers.**

Keywords-Energy Conservation Scheme; Lifespan Extensibility Metrics; Resource Exchange for Energy Conservation Scheme; Opportunistic Communication Performance; Traffic-oriented Energy Conservation.

I. INTRODUCTION

As wireless nodes communicate over error-prone wireless channels with limited battery power, reliable and energy-efficient data delivery is crucial. These characteristics of wireless nodes make the design of resource exchange schemes challenging [1]. Due to the fact that wireless devices in order to conserve energy switch their states between Sleep mode, Wake mode and Idle mode, the responsiveness of these devices is reduced significantly. These devices, while being in the process of sharing resources, face temporary and unannounced loss of network connectivity as they move, whereas, they are usually engaged in rather short connection sessions since they need to discover other hosts in an ad-hoc manner. In most cases the requested resources claimed by these devices, may not be available. Therefore, a mechanism that faces the intermittent connectivity problem and enables the devices to react to frequent changes in the environment, while it enables energy conservation in regards to the requested traversed traffic, is of great need. This mechanism will positively affect the end-to-end reliability, facing the unavailability and the scarceness of wireless resources.

This work proposes a backward estimation model for extracting the time-oriented differential traffic in contrast to the resource capacity characteristics in order to offer Energy Conservation and availability of the requested resources. The proposed scheme uses the cached mechanism for guaranteeing the requested resources and the monitored traffic that traverses the nodes, both as input (basis for estimating the next sleep time duration of each node). This mechanism follows the introduced Backward Traffic Difference (BTD) scheme. The designed model guarantees the end-to-end availability of requested resources while it reduces significantly the Energy Consumption and maintains the requested scheduled transfers, in a mobility-enabled cluster-based communication. The innovative aspect of this work is that each node uses different assignment(s) of sleep-wake schedule estimation, based on the traffic difference through time. The Sleep-time duration is assigned according to the scheme in a dissimilar form to enhance node prolonged hibernation (where needed), whereas it avoids mutation which, will result in network partitioning and resource sharing losses. Real-Time experiments using various newly introduced metrics' estimations, were carried-out for the energy conservation and the evaluation of the proposed model. The scheme takes into account a number of metrics hosted by the proposed scheme, as well as estimation of the effects of incrementing the sleep time duration to conserve energy. Likewise, the Real-Time experiments show that different types of traffic can be supported where the adaptability and the robustness that is exhibited, is mitigated according to the proposed scheme's Sleep-time estimations and assignments.

The structure of this work is as follows: Section II describes the related work done and the need in adopting a Traffic-based scheme, and then Section III follows presenting the proposed Backward Traffic Difference Estimation for Energy Conservation for Mobile opportunistic resource sharing. Section IV presents the real time performance evaluation results focusing on the behavioral characteristics of the scheme and the Backward Traffic Difference along with the system's response, followed by Section V with the conclusions and foundations, as well as potential future directions.

II. RELATED WORK

Many recent high-quality measurement studies [1-5] have convincingly demonstrated the impact of Traffic on the end-to-end connectivity [6], and thus the impact on the Sleep-time duration and the EC. The realistic traffic in Real-Time communication networks and multimedia systems, including wired local-area networks, wide-area networks, wireless and

 ISBN: 978-1-61208-238-7

mobile networks, exhibits noticeable burstiness over a number of time scales [8] [9] and [10]. This fractal-like behavior of network traffic can be much better modeled using statistically self-similar or Long Range Dependent (LRD) processes, which, have significantly different theoretical properties from those of the conventional Short Range Dependent (SRD) processes. There are many Sleep-time scheduling strategies that model the node transition between ON and OFF states. Existing scheduling strategies for wireless networks could be classified into three categories: the coordinated sleeping [11] [12], where nodes adjust their sleeping schedule, the random sleeping [13] and [14], where there is no certain adjustment mechanism between the nodes in the sleeping schedule with all the pros and cons [15], and on-demand adaptive mechanisms [16], where nodes enter into Sleep-state depending on the environment requirements whereas an out-band signaling is used to notify a specific node to go to sleep in an on-demand manner.

In addition to the existing architectures, a fertile ground of the development of new approaches has been the association of different parameters with communication mechanisms in order to reduce the Energy Consumption. These mechanisms can be classified into two categories: Active and passive schemes. Active techniques conserve energy by performing energy conscious operations, such as transmission scheduling and energy-aware routing. Mavromoustakis in [2] considers the association of EC problem with different parameterized aspects of the traffic (like traffic prioritization) and enable a mechanism that tunes the interfaces' scheduler to sprawl in the sleep state according to the activity of the traffic of a certain node in the end-to-end path.

The main goal of the proposed scheme is to minimize the energy consumption using the incoming Traffic that is destined onto each one of the nodes, taking into consideration the repetition pattern of the Traffic. The scheme then estimates the Backward Difference for extracting the time duration for which the node is allowed to Sleep during the next time slot *T*. This mechanism, in order to enable further recoverability and availability of the requested resources, proposes an efficient way to cache the packets destined for the node with turned-off interfaces (sleep state) onto intermediate nodes and enables, through the Backward Traffic Difference estimation, the next Sleep-time duration of the recipient node to be adjusted accordingly.

III. BACKWARD DIFFERENCE TRAFFIC ESTIMATION FOR ENERGY CONSERVATION FOR MOBILE PEER-TO-PEER OPPORTUNISTIC RESOURCE SHARING

The input nodal traffic is being considered in this work and estimated according to the Backward Traffic Difference (BTD). Wireless nodes, even if they are acting in the network as intermediate forwarding nodal points or as destinations, they have to be self-aware in terms of power and processing as well as in terms of accurate participation in the transmission activity. There are many techniques such as the dynamic caching-oriented methods. The present work utilizes a hybridized version of the proposed adaptive dynamic caching [2], which is considered to behave satisfactorily and enables simplicity in real time implementation [3]. On the contrary with [3][4], in this work a different real-time mobility scenario is modeled and hosted in the scheme, which, enables an adaptive tuning of the Sleep-time duration according to the activity of the Traffic on each node.

The following section presents the estimations performed on each node in order to evaluate the next Sleep-time duration according to the node's incoming activity by using the BTD mechanism.

A. Backward Difference Traffic Estimation for Energy Conservation

The proactive activity scheduling may increase the network lifetime, contrarily with periodic Sleep-Wake schedules, as it enables dissimilar active-time. The nodes are set in the active state for a period of time according to the incoming traffic. The activity period(s) of a node is primarily dependent on the nature and the spikes of the incoming traffic destined for this node [6]. If the transmissions are performed on a periodic basis then the nodes' lifetime can be forecasted and according to a model can be predicted and estimated [7]. This work introduces the Backward Traffic Difference (BTD) estimation in order to associate the traversed traffic of a node with the previous moments and, in real-time, reduce the redundant Activity-periods of the node in order to conserve energy. Figure 1 shows in Real-time the Incoming traffic that a node experiences with the associated traffic capacity and activity duration of the node. The traffic can be seen as a renewal process [7] that has aggregation characteristics [9] from different sources.

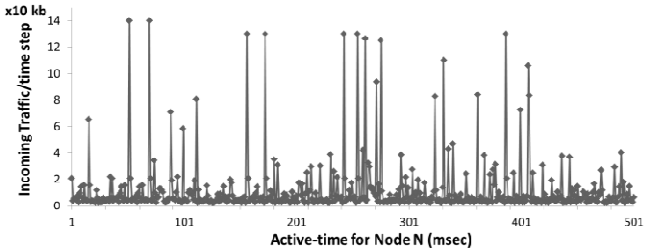


Figure 1. Real-time Incoming traffic that a node experiences with the associated traffic capacity and activity duration of the node with mean $E(A_{t_f})$.

This work assigns a dissimilar sleep and wake time for each node, based on traffic that is destined for each node which, is cached onto 1-hop intermediate node(s), during the sleep time duration. Figure 2 shows that, in a pre-scheduled periodic basis, nodes can be in the Sleep-state. Likewise, the packets that are destined for the certain node can be cached for a specified amount of time (as long as the Node (i) is in the Sleep-state) in the 1-hop neighbor node (Node(i-1)) in order to be recoverable when node enters the Active state.

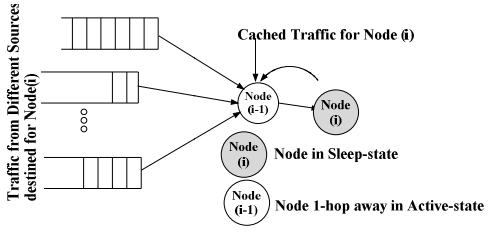


Figure 2. A schematic diagram of the caching mechanism addressed in this work.

 ISBN: 978-1-61208-238-7

The 1-hop neighbor node (*Node(i-1)*) is selected to cache the packets destined for the node with turned off interfaces (sleep state). The principle illustrated in Figure 2 denotes that, when incoming traffic is in action for a specific node then the node remains active for prolonged time. As a showcase this work takes the specifications of the IEEE 802.11x that are recommending the duration of the forwarding mechanism that takes place in a non-power saving mode lays in the interval *1 nsec < τ <1 psec*. This means that every ~0.125μsec (8 times in a msec) the communication triggering action between nodes may result a problematic end-to-end accuracy. Adaptive Dynamic Caching [2] takes place and enables the packets to be "cached" in the 1-hop neighboring nodes. Correspondingly, if node is no-longer available due to sleep-state in order to conserve energy (in the interval slot *T=0.125μsec*), then the packets are cached into an intermediate node with adequate capacity equals to: $C_{t_f,k(s)}(t) > C_{t_f,i}(t)$, where $C_{t_f} > \alpha \cdot C_i$; where α_i is the capacity adaptation degree based on the time duration of the capacity that is reserved on node N of C_k ; where $C_{t_f,k(s)}(t)$ is the needed capacity where i is the destination node and k is the buffering node (a hop before the destination via different paths).

As this scheme is entirely based on the aggregated self-similarity nature of the incoming traffic with reference to a certain node, there should be an evaluation scheme in order to enable the node to Sleep, less or more according to the previous activity moments. This means that, as more as the cached traffic is, there is an increase in the sleep-time duration of the next moment for the destination node. This is indicated in the following scheme which, takes into account the Self-Similarity to estimate the potential spikes of the Sleep-time duration. The Sleep-time in turn decreases or increases accordingly, based on the active Traffic destined for *Node(i)* while being in the Sleep-state.

1) Backward Difference Traffic Moments and Sleep-time duration estimation

Let $C(t)$ be the capacity of the traffic that is destined for the Node *i* in the time slot (duration) *t*, and $C_{N_i(t)}$ is the traffic capacity that is cached onto *Node (i-1)* for time *t*. Then, the one-level Backward Difference of the Traffic is evaluated by estimating the difference of the traffic while the *Node(i)* is set in the Sleep-state for a period, as follows:

$$\begin{aligned} \nabla C_{N_i(1)} &= T_2(\tau) - T_1(\tau-1) \\ \nabla C_{N_i(2)} &= T_3(\tau-1) - T_2(\tau-2) \\ &\vdots \\ \nabla C_{N_i(n+1)} &= T_n(\tau-(n-1)) - T_2(\tau-(n-2)) \end{aligned} \quad (1)$$

where $\nabla C_{N_i(1)}$ denotes the first moment traffic/capacity difference that is destined for *Node(i)* and it is cached onto Node (i-1) for time τ , $T_2(\tau) - T_1(\tau-1)$ is the estimated traffic difference while packets are being cached onto *(i-1)* hop for recoverability. Equation (1) depicts the BTD estimation for one-level comparisons, which means that the moments are only being estimated for one-level ($T_2(\tau) - T_1(\tau-1)$). The Traffic Difference is estimated so that the next Sleep-time duration can be directly affected according to the following:

$$\delta(C(T)) = C_{total} - C_1, \forall C_{total} > C_1, T \in \{\tau-1, \tau\} \quad (2)$$

where the Traffic that is destined for *Node(i)*, urges the Node to remain active for $\frac{\delta(C(T))}{C_{total}} \cdot T_{prev} > 0$.

The load generated by one source is mean size of a packet train divided over mean size of packet train and mean size of inter-train gap or it is the mean size of ON period over mean size of ON and OFF periods as follows:

$$L_i = \frac{\overline{ON_i}}{\overline{ON_i} + \overline{OFF_i}} \quad (3)$$

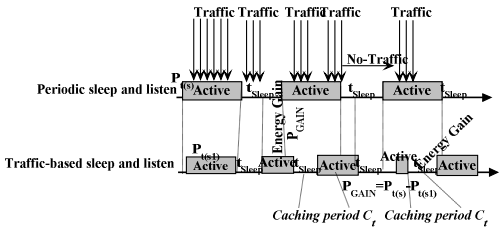


Figure 3. ON and OFF periodic durations of a Node with the associated cached periods.

When a node admits traffic, the traffic flow t_f , can be modeled as a stochastic process [17] and denoted in a cumulative arrival form as $A_{t_f} = \{A_{t_f}(T)\}_{T \in N}$, where $A_{t_f}(T)$ represents the cumulative amount of traffic arrivals in the time space [0..T]. Then, the $A_{t_f}(s,T) = A_{t_f}(T) - A_{t_f}(s)$ (4), denotes the amount of traffic arriving in time interval (*s*, *t*]. Hence the next Sleep-time duration for *Node (i)* can be evaluated as:

$$L_i(n+1) = \frac{\delta(C(T) \mid A_{t_f}(s,T))}{C_{total}} \cdot T_{prev}, \forall \delta(C(T)) > 0 \quad (5)$$

For the case that the $\delta(C(T)) < 0$ it stands that:

$\delta(C(T)) = C_{total} - C_1, \forall C_{total} < C_1, T \in \{\tau-1, \tau\}$, and $\frac{\delta(C(T))}{C_{total}} \cdot T_{prev} < 0, \forall T_{prev} > T_{prev}(\tau-1)$, the $C_{N_i} < 0$ and the total active time increases gradually according to the following estimation:

$$T_{sleep} = T(\tau - t_1) - (-C_{N_i}) = T(\tau - t_1) + T_{C_{N_i}} \quad (6)$$

where the $T_{C_{N_i}}$ is the estimated duration for the capacity difference for $C_{N_i} < 0$, whereas the Sleep-time duration decreases accordingly with Equations (5) and (6), iff the $C_{N_i} < 0$. Considering the above estimations the Traffic flow can be expressed as in [19] as

$$A_{t_f}(T) = m_{t_f}(T) + \hat{Z}_{t_f}(T) \quad (7)$$

where $m_{t_f}(T)$ is the mean arrival rate and $\hat{Z}_{t_f}(T) = \sqrt{a_{t_f} m_{t_f}(T)} \cdot \hat{Z}_{t_f}(T) \cdot a_{t_f}$. The coefficient a_{t_f} is the variance coefficient of $A_{t_f}(T)$. $\hat{Z}_{t_f}(T) \cdot$ is the smoothed mean as in [17], and with $\mathrm{E}(\hat{Z}_{t_f}(T)) = 0$ satisfying the following variance and covariance functions:

$$v_{t_f} = a_{t_f} m_{t_f} \cdot T^{2H_{t_f}}$$
$$\sigma_{t_f}(s,T) = \frac{1}{2} a_{t_f} m_{t_f} \cdot (T^{2H_{t_f}} + s^{2H_{t_f}} - (T-s)^{2H_{t_f}}) \quad (8)$$

where $H_{t_f} \in \left[\frac{1}{2}, 1\right]$ is the *Hurst* parameter, indicating the degree of self-similarity. Estimations in (8) can only be valid if the capacity of the *Node (i-1)* can host the aggregated traffic destined for *Node (i)* satisfying the

$\sup_{s \le T} \left\{ \sum_{t_f=1}^{N} A_{t_f}(s,T) - C_{t_f}(T) \right\}$, for traffic flow t_f at time T and $C_{t_f}(T)$ represents the service capacity of the *Node(i-1)* for this time duration.

The basic steps of the proposed scheme can be summarized in the pseudocode of the Table 1.

```
for Node(i) that there is C(t)>0 {
  while ( C_{N_i(t)}>0) { //cached Traffic measurement
    Evaluate ( ∇C_{N_i(1)} );
Calc( δ(C(T)) = C_total − C_1, ∀C_total > C_1, T ∈ {τ−1, τ} )
if (Activity_Period= δ(C(T))/C_total · T_prev >0)
//Measure Sleep-time duration
L_i(n+1) = δ(C(T) | A_{t_f}(s,T))/C_total · T_prev, ∀δ(C(T)) > 0
else if ( δ(C(T)) < 0 )
    T_sleep = T(τ − t_1) − (−C_{N_i}) = T(τ − t_1) + T_{C_{N_i}} ;
    Sleep ( T_sleep );
    } //for
}//while
```

Table 1. Basic steps of the proposed BTD scheme.

Taking into consideration the above stochastic estimations, the Energy Efficiency EE_{t_f} can be defined as a measure of the capacity of the *Node(i)* over the *Total Power consumed* by the *Node*, as:

$$EE_{t_f}(T) = \frac{C_{t_f}(T)}{TotalPower} \quad (9)$$

Equation 9 above can be defined as the primary metric for the lifespan extensibility of the wireless node in the system.

IV. REAL TIME PERFORMANCE EVALUATION ANALYSIS, EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we demonstrate the effectiveness of the proposed BTD approach by using the MICA2 sensors nodes [20]. Nodes are configured to be manipulated as Peer devices hosting the proposed BTD scheme. These sensors were equipped with the MTS310 sensor boards. The MICA2 features a low power processor and a radio module operating at 868/916 MHz enabling data transmission at 38.4Kbits/s with an outdoor range of maximum set to 50 meters-taking no fading obstacles in-between for better and clearer signal strength. The TinyOS operating system is hosted onto MICA2 using the Nested C (NesC) language. A dynamic topology with the mobility expressed in Section IV.A is implemented, where the BTD scheme assigns the Traffic-oriented Sleep and Wake durations. In the evaluation of the proposed scheme we took into account the signal strength measures as developed in [2] and [3] and the minimized ping delays between the nodes in the end-to-end path. The underlying communication supports the Cluster-based Routing Protocol (CRP) [21]. A common look-up application is being developed to enable users to share resources on-the-move which, are available by peers for sharing. This application hosts files of different sizes that are requested by peers in an opportunistic manner.

A. Mobility Model used for mobile peers

In the proposed scenario the new speed and direction are both chosen from predefined ranges, [v_{min}, v_{max}] and [0, 2π), respectively [17] and [18]. The new speed and direction are maintained for an arbitrary length of time, randomly chosen from (0, t_{max}]. At the end of the chosen time, the node makes a memoryless decision of a new random speed and direction. The movements are expressed as a Fractional Random Walk (FRW) on a Weighted Graph [22].

B. Real-time performance testing and evaluation using the MICA2 sensors equipped with the MTS310 sensor boards

In this section, we present the results extracted after conducting the real time evaluation runs of the proposed scenario. In the utilized scenario we have used 30 nodes with each link (frequency channel) having max speed reaching data transmission at 38.4Kbits/s. The wireless network is organized in 6 overlapping clusters which, may vary in time in the active number of the nodes. Each source node transmits one 512-bytes (~4Kbits-light traffic) packet asynchronously and randomly each node selects a destination. The speed of each device can be measured with the resultant direction unit vector [3] and the speed. Each device has an asymmetrical storage capacity compared with the storages of the peer devices. The range of the capacities for which devices are supported are in the interval 1MB to 20MB.

Figure 4 shows the fraction of the remaining Energy through time in contrast to the comparison and evaluation extracted for different schemes during the real-time experimentation. As all schemes aim to reduce the Energy consumption, the proposed scheme behaves satisfactorily in contrast to the scheme developed in [6]. Figure 5 shows the Successful packet Delivery Ratio (SDR) in regards to the simultaneous requests in the intra-cluster communicating path, for both statically located nodes (no movement) and mobile nodes. Figure 6 shows the Average

 ISBN: 978-1-61208-238-7

Throughput with the Total Transfer Delay in (μsec) is shown, for different mobility models. Figure 6 depicts the different Throughput responses that the proposed scheme exhibits in contrast to the mobility characteristics, for full node mobility, moderate and low (30%) mobility.

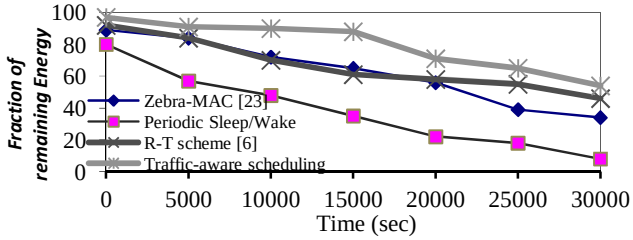


Figure 4. The fraction of the remaining Energy through time using real-time evaluation for different schemes.

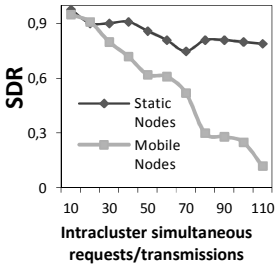


Figure 5. The Successful packet Delivery Ratio (SDR) with the simultaneous requests.

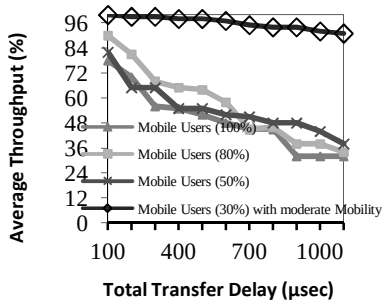
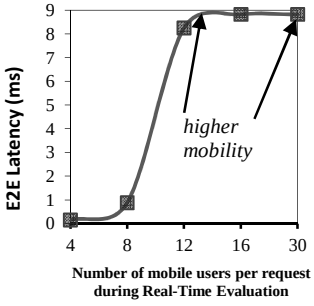


Figure 6. The Average Throughput with the Total Transfer Delay (μsec).



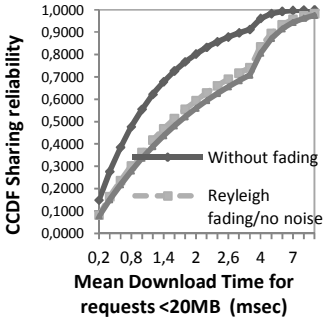


Figure 7. The End-to-End Latency with the number of requests for the users during Real-Time evaluation; and the CCDF for the Sharing Reliability with the Mean download Time for requests over a certain capacity.

The End-to-End Latency with the number of requests for the users during Real-Time evaluation is shown in Figure 7 indicating the number of users that are utilized in the presence of high mobility. Likewise, Figure 7 shows the respective Complementary Cumulative Distribution Function (CCDF or simply the tail distribution) with the Mean download Time for requests over a certain capacity. The later results were extracted in the presence of fading and no-fading communicating obstacles. Figures in 8 show the network lifetime with the number of Mobile Nodes; and the Throughput response of the system hosting the proposed scheme with the Number of requests for certain fading evaluated characteristics.

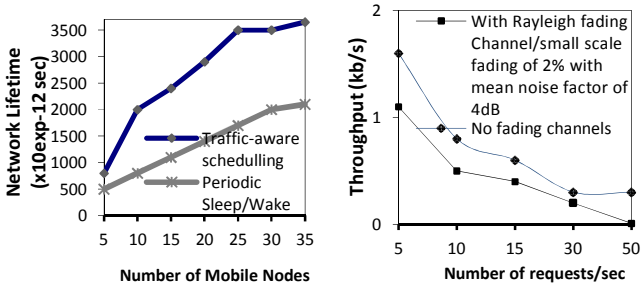


Figure 8. Network Lifetime with the Number of Mobile Nodes; and the Throughput response of the system hosting the proposed scheme with the Number of requests for certain fading measures' characteristics.

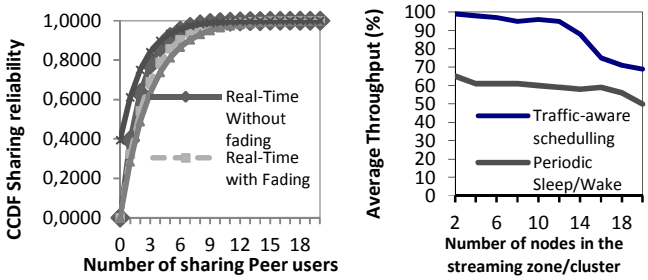


Figure 9. CCDF Sharing Reliability with the Number of sharing Peer-users; and the Average Throughput with the number of Nodes in the streaming zone.

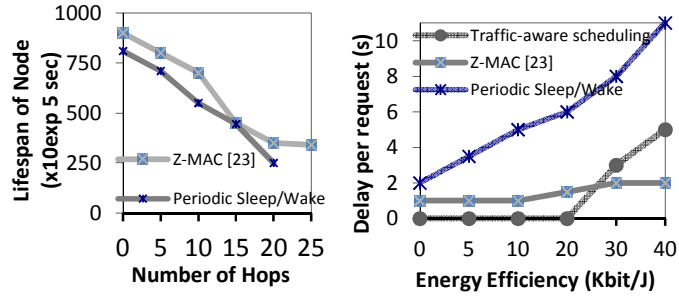


Figure 10. Lifespan of each node with the number of hops for different schemes Real-Time comparisons; and the measures for the Delay requests with the corresponding Energy efficiency.

Results obtained presented in Figure 9, show the CCDF Sharing Reliability with the Number of sharing Peer-users; and the Average Throughput with the number of Nodes in the streaming zone. The results extracted for CCDF Sharing Reliability with the Number of sharing Peer-users, were for both Simulation experiments and Real-Time estimations so that there is a comparison axis between the simulated results and the results extracted from Real-Time Traffic and experiments. In addition, the Average Throughput in contrast to the number of Nodes in the streaming zone was evaluated in Real-Time using periodic and the proposed scheme. It is undoubtedly true that the proposed scheme enables higher Average Throughput response in the system, whereas comparing with the results extracted from Figure 4, the proposed scheme enables greater network lifetime by using this activity Traffic-based scheme. Figures in 10, show the Lifespan of each node with the number of hops for different schemes Real-Time comparisons; and the measures for the Delay requests with the corresponding Energy efficiency. Results in Figure 10 show that the network lifetime can be significantly prolonged when the

 ISBN: 978-1-61208-238-7

BTD is applied, in contrast with the results obtained in Real-Time experiments for the [23] and for the periodic Sleep/Wake scheduling.

V. CONCLUSIONS AND FURTHER RESEARCH

This work considers the BTD scheme hosted on wireless nodes during the resource exchange process. This research proposes and examines a backward estimation model for allocating -upon estimation- the time-duration that a node is allowed to sleep (according to the traversed traffic) so that it conserves Energy. The scheme uses the cached mechanism for guaranteeing the requested resources and a model for the Sleep time estimation based on the incoming Traffic that traverses the nodes. According to the Real-Time results extracted, the designed model guarantees the end-to-end availability of requested resources while it reduces significantly the Energy Consumption, while it maintains the requested scheduled transfers. Performance evaluation and the results extracted in Real-Time show that this method uses optimally the network's and system's resources in terms of capacity and EC and offers high SDRs particularly in contrast with other similar existing Energy-efficient methods. Next steps and on-going work within the current research context will be the expansion of this model into a Multi-level Markov Fractality Model so that it associates the different moments of the Traffic activity and it will be able to extract the Sleep-time estimations for the nodes, in order to enable them to conserve Energy, while maintaining the resource sharing process on-the-move.

ACKNOWLEDGMENT

We would like to thank the European FP7 COST Action IC0703 "Data Traffic Monitoring and Analysis (TMA): theory, techniques, tools and applications for the future networks", for the active support and cooperation.

REFERENCES

[1] J. Tang, M. Musolesi, C. Mascolo, V. Latora, and V. Nicosia, Analysing Information Flows and Key Mediators through Temporal Centrality Metrics. In Proceedings of 3rd Workshop on Social Network Systems (SNS 2010). Paris, France. April 2010, pp. 256-262.

[2] C. X. Mavromoustakis, "On the impact of caching and a model for storage-capacity measurements for energy conservation in asymmetrical wireless devices", IEEE Communication Society (COMSOC), 16th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2008), September 25 & 26 2008, "Dubrovnik", September 27, Split and Dubrovnik, pp. 243-247.

[3] C. X. Mavromoustakis and K. G. Zerfiridis, "On the diversity properties of wireless mobility with the user-centered temporal capacity awareness for EC in wireless devices". Proceedings of the Sixth IEEE International Conference on Wireless and Mobile Communications, ICWMC 2010, September 20-25, 2010-Valencia, Spain, pp. 367-372.

[4] C. X. Mavromoustakis, "Synchronized Cooperative Schedules for collaborative resource availability using population-based algorithm", Simulation Practice and Theory Journal, Elsevier, Volume 19, Issue 2, February 2011, pp. 762-776.

[5] C. X. Mavromoustakis and H. D. Karatza, "A tiered-based asynchronous scheduling scheme for delay constrained energy efficient connectivity in asymmetrical wireless devices", The Journal of Supercomputing, Springer USA, Volume 59, Issue 1, (2012), pp. 61-82.

[6] C. X. Mavromoustakis and H. D. Karatza, "Real time performance evaluation of asynchronous time division traffic-aware and delay-tolerant scheme in ad-hoc sensor networks", International Journal of Communication Systems (IJCS), Wiley, Volume 23 Issue 2 (February 2010), pp. 167-186.

[7] O. Sheluhin, S. Smolskiy, and A. Osin, Self-Similar Processes in Telecommunications, ISBN: 978-0-470-01486-8, 334 pages, March 2007.

[8] W. E. Leland, M. S. Taqq, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," IEEE/ACM Trans. Networking, vol. 2, no. 1, 1994, pp. 1-15.

[9] J. Yu and A. P. Petropulu, "Study of the effect of the wireless gateway on incoming self-similar traffic," IEEE Trans. Signal Proceesing, vol. 54, no. 10, 2006, pp. 3741-3758.

[10] X. Zhuang and S. Pande, "A scalable priority queue architecture for high speed network processing," in Proc. 25th IEEE International Conf. Computer Commun. (INFOCOM'06), 2006, pp. 1-12.

[11] Q. Cao, T. Abdelzaher, T. He, and J. Stankonic, "Towards optimal sleep scheduling in sensor networks for rare-event detection", the 4th international symposium on Information processing in sensor networks, 2005, pp. 34-39.

[12] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks", IEEE Transactions on Computers, 2003, vol 14, 4, pp. 334-342.

[13] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", 21st Conference of the IEEE Computer and Communications Societies (INFOCOM), 2002, Vol. 3, pp. 1567-1576.

[14] T. Dam and K. Langendoen, "An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks,", the 1st international conference on Embedded networked sensor systems, 2003, pp. 112-118.

[15] H. Cunqing and P. Y. Tak-Shink, "Asynchronous Random Sleeping for Sensor Networks", ACM Transactions on Sensor Networks (TOSN), 2007, Volume 3 Issue 3, pp. 1123-1132.

[16] I. Jawhar, J. Wu and P. Agrawal, Resource Scheduling in Wireless Networks Using Directional Antennas, appears in: IEEE Transactions on Parallel and Distributed Systems, Sept. 2010, Volume: 21 Issue:9, pp. 1240-1253.

[17] O. Ibe, Markov Processes for Stochastic Modeling, ISBN-10: 0123744512, Academic Press (September 16, 2008).

[18] C. Mavromoustakis, "Optimizing the End-to-End Opportunistic Resource Sharing using Social Mobility", Proceedings of the First International Conference on Intelligent Systems and Applications, INTELLI 2012, April 29-May 4, 2012 - Chamonix / Mont Blanc, France, pp. 41-46.

[19] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, Dynamics of IP traffic: A study of the role of variability and the impact of control, 1999, ACM SIGCOMM pp. 301-313.

[20] http://www.xbow.com/pdf/ containing the Specifications Crossbow Technology, Industry's First End-to-End, Low-Power, Wireless Sensor Network Solution for Security, Monitoring, and Tracking Applications, last accessed on 20/3/2012.

[21] A. Boukerche, R. Nelem-Pazzi, and A. Martirosyan, Energy aware and cluster-based routing protocols for large-scale ambient sensor networks, Proceedings Ambi-Sys '08 Proceedings of the 1st international conference on Ambient media and systems 2008, pp. 306-311

[22] A.L. Barabási and R. Albert. Emergence of scaling in random networks. Science, 286(5439), 1999, pp. 509–512.

[23] A. Rhee, M. Warrier, J. Aia, and M. Sichitiu, Z-MAC: a hybrid MAC for wireless sensor networks in IEEE/ACM Transactions on Networking vol. 16, no. 3, 2008, pp. 511-524.

.

 ISBN: 978-1-61208-238-7

Using Spacial Locality and Replication to Increase P2P Network Performance in MMO Games

Ross Humphrey, Alexander Allan and Giuseppe Di Fatta
School of Systems Engineering
The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
gk004759@reading.ac.uk, A.J.M.Allan@student.reading.ac.uk, G.DiFatta@reading.ac.uk

***Abstract*—Massively Multiplayer Online Games (MMOGs) are increasing in scale and popularity, which is putting strain on the classical client-server(C/S) architecture. As a consequence there is a growing research interest in the adoption of Peer to Peer (P2P) architectures to spread the load throughout participating client machines. However this presents many challenges, amongst which is creating a shared virtual space between nodes, an area known as Interest Management. When using the Region-Based model of Interest Management the game world is mapped to a logical space which is broken into regions managed by peers. When a player's avatar moves through the game-space it moves through these regions, and must download content from the appropriate peer. Finding this peer can be handled by a look-up on a Distributed Hash Table with a circular key. This work explores the advantage of mapping Distributed Hash Table(DHT) keys using a locality preserving function instead of a conventional uniformity enforcing hash algorithm within a P2P protocol for MMOGs. Content retrieval robustness in terms of handling node failures is also explored with multiple data replication techniques analysed and compared. The performance difference is measured in terms of hop count, node stabilisation and node failure. Results show that using locality sensitive hashing and 12 node replication provided favourable performance across all three measurements used.**

Keywords-Peer-to-Peer Networks; Massively Multiplayer Online Games; Interest Management.

I. INTRODUCTION

A Massively Multiplayer Online Game (MMOG) allows large numbers of on-line players to interact with each other in the same persistent virtual space. Traditionally Client Server (C/S) architectures were employed as they were convenient in terms of implementation and security [1]. However, the popularity of MMOGs is rapidly increasing [2] which is putting increasing strain on the C/S architecture presenting, amongst other things, a large cost in terms of hardware and facilities, a single point of failure and a network bottleneck. This has led to a growing research interest towards architectures capable of harnessing the power of client machines in the form of Peer to Peer (P2P) MMOGs [3], [4], [5], [6].

A key problem when creating a P2P MMOG is maintaining a sense of shared space across all players. This can be solved by each player having a full copy of the game state and broadcasting any changes to all other players. However, this is not feasible as the number of messages needing to be sent over the network scales exponentially with the number of players.

A solution to this is to send players only relevant information, a process known as Interest Management [7]. This can be achieved by dividing the game up spatially. A fine grained approach to this is the *aura-nimbus* information model [8]. The *aura* bounds the presence of an object in space, with the *nimbus* (the area-of-interest) representing the boundary within which the object can perceive other objects.[9] While this allows very accurate Interest Management, it does not scale well due to the cost of computing the intersection between areas of interest and *auras* of objects [10].

Region-based interest management is an approximation which addresses the scalability issues of the pure aura-nimbus model by partitioning the game space into static regions [8], [11], [12], [13].

Traversal of these regions requires that a player query a lookup table of some kind in order to contact the node who is regional administrator. In a distributed environment this lookup can be achieved using a node ID as the index for a Distributed Hash Table (DHT). Chord offers one such scalable implementation of a DHT with a lookup performance of $O(log(n))$ and allows for index updates when nodes join and leave the network [14]. Chord was chosen as it represents a simple case of a distributed hash table in which to use as a test bed for experimentation relating to replication and region based interest management.

It is common practice when assigning DHT keys to do so in a manner which assigns IDs uniformly around the keyspace, a method known as consistent hashing. This is commonly achieved by using a cryptographic hash algorithm such as MD5[15] or SHA-1[16]. However, since in this application domain regions are traversed sequentially, a locality preserving hash function could be more efficient in terms of DHT hop count than a conventional one. This work presents a comparison of four different keyspace generation methods for a Chord DHT. Consistent hashing given by the MD2[17] and MD5 cryptographic algorithms is compared to linear and Hilbert curve based methods (outlined in Section II, Subsections C and D). The effectiveness of the four methods is measured in terms of hop count and node keyspace stabilisations within a simulation of a P2P protocol for MMOGs.

This work also uses the same simulation to look at the the effectiveness of implementing a replication algorithm to maintain the games' state in event of node loss.

 ISBN: 978-1-61208-238-7

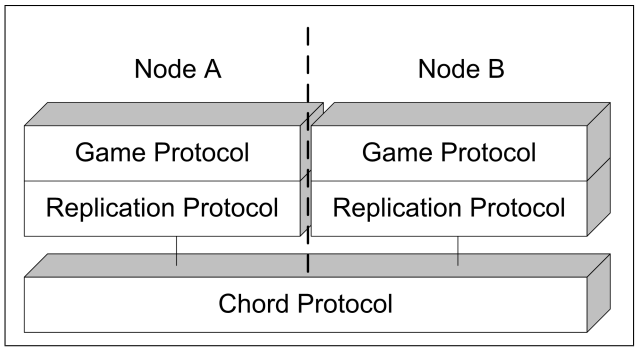


Fig. 1. Components of the simulation protocol stack

The rest of the paper is organised as follows. Section II provides additional detail on the ID mapping and replication techniques used within the simulated environment. Section III provides a description of the simulated P2P MMOG environment. Section IV describes the comparative analysis of the ID mapping algorithms and data replication techniques. Section V provides the experimental results and their interpretation. Concluding remarks are given in Section VI.

II. Overview of Techniques

A. Chord DHT

Chord was used as the underlying P2P protocol for this simulation. Chord is a Distributed Hash Table (DHT) which allows efficient mapping of keys onto nodes in a peer to peer environment [14]. By using Chord a node can locate another node within $O(log(n))$ hops. The key/value pairs are spread throughout the network giving Chord its ability to scale to a large number of peers.

The Chord protocol layer is comprised of a number of nodes. Each node contains a section of the DHT with the key/value pairs stored within it. The section held within each node is changed when a node leaves or joins the network. A stabilisation takes place when this table is changed. A stabilisation consumes both network and hardware resources as it involves querying the nodes on the network and changing values within a node dependent on queries made. The DHT within this simulation uses a 128-bit key space; this key space can be changed depending on the ID allocation used for the node.

Each node within a Chord ring has an identifier that indicates where the node is mapped in the logical ID space. The ID in a chord network is typically generated using a one way consistent hash function such as MD5. Two forms of consistent (non locality preserving and uniformity enforcing) hashing were implemented as comparison with the two locality preserving key generation algorithms. The two implementations decided on were MD5 and MD2 which both produce a 128-bit ID.

B. Consistent Hashing - Message Digest Algorithms MD2 and MD5

A message digest algorithm is a one way cryptographic hash function which outputs a string of fixed length (a hash) for an arbitrary size of input data.

The MD2 and MD5 algorithms are derivations of the popular Merkle-Damgård method [18] and output 128-bit words for any given input. These methods of consistent hashing are commonly used within the Chord protocol to map IP addresses into the key space and will be used in comparison with locality aware hashing methods.

C. Locality Preserving - X-Axis Linear curve

A simple method of embedding locality in a 1D index is by an ordered linear traversal along an axis. The 2Dimensional space is broken down into a grid of a finite granularity and the grid squares are assigned an index sequentially. The X-Axis method fixes the y axis for each traversal of the x axis. This has the effect that locality is preserved effectively in the x-coordinate at the expense of relatively poor locality preservation in the y-coordinate.

The ID allocation operates by assigning IDs to nodes in a linear fashion across the x-axis. The nine squares of the grid in Figure 2 are traversed in the following sequence of coordinates:

$$(0,0),(1,0),(2,0),(0,1),(1,1),(2,1),(0,2),(1,2),(2,2)$$

This method is compared to the consistent hashing (uniformity enforcing) method provided by MD2 and MD5, and with the locality preserving Hilbert curve as a method of key generation for a Chord DHT.

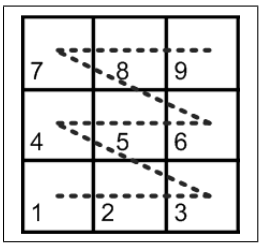


Fig. 2. X-axis linear curve in a 3x3 2-dimensional grid

D. Locality Preserving - 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 2D space (and by extension in an n-dimensional hypercube) [19], [20]. This work was followed in 1891 by that of David Hilbert [21] who published his own version of the space-filling curve including illustrations for construction (Figure 3). Hilbert's variant proves to have performance advantages (in terms of how locality and how well 'compact regions' of 2D space are represented) over other space-filling curves [22], [23]. This explains its attraction

 ISBN: 978-1-61208-238-7

as a contemporary multi-dimensional indexing method. 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 means of dimensionality reduction, it transforms the data from n to 1 dimension by assigning each point in space a number.

We compare the key space generated by the Hilbert Curve with those generated by the consistent hashing algorithms MD2 and MD5, and with the simple X-axis curve.

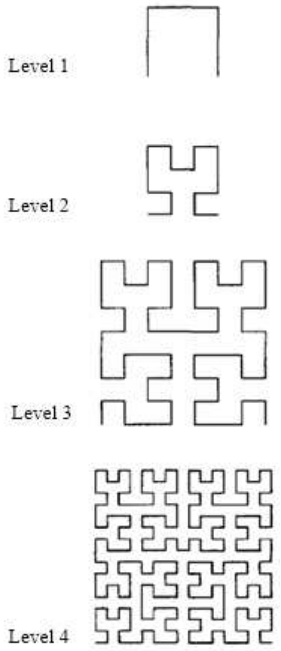


Fig. 3. The first 4 levels of Hilbert's curve in 2 dimensions

E. Replication

The elastic nature of multiplayer online games does not lend itself well to a static P2P network. As players log out spuriously or disconnect from the network data is lost and cannot be recovered until the peer next logs in to the network. This is a problem when objects in the game world go offline and other players wish to access them.

In order to solve this problem replication must be implemented so that when a player leaves the network, the persistent world in the game continues without the data being lost. A replication strategy was devised that would allow for data to be replicated to other nodes in the network. This replication strategy involves mirroring information using a set of replication nodes.

If for example node n queries node a for data d, but finds that a is no longer present in the network, n can then query a's replication nodes. As long as at least one of a's replication nodes remain online, n will be be able to retrieve d.

Formally each node n has a set of R_n replication nodes $Rep_n = \{r_0^n, ..r_i^n\}$ where $0 \leq i \leq R_n$ which each carry a copy of its data. One of these replication nodes is chosen randomly, the others are chosen from n's successor nodes on the DHT. The number of replication nodes used at node n, R_n, varies and is based on the popularity of the data at n. If data is requested more than once every 5 seconds from n then Rep_n will contain all DHT successor nodes and one random node.

When data is modified at node n or any member of Rep_n, a lock message is sent to n and each member of Rep_n. When the data is successfully modified, its new value is broadcast to n and members of Rep_n. When all these nodes have successfully received the new value, an unlock message is broadcast amongst replicated nodes.

III. Simulation of a P2P MMOG protocol

The proposed system contains a number of components that are used to simulate a P2P MMOG (Figure 1). The system adopts three separate protocols: the Chord DHT Protocol, the Game Protocol and the Replication Protocol.

1) The Chord DHT Protocol: is used to find the information being searched for by the user in the network. The layer uses the nodes provided by the simulated network, when searching for the requested data. The search data requested is provided by the Game Protocol that receives the requests from the Traffic Generator. The traffic generator builds requests based on coordinates. The coordinates are relative to the position within a game world. From this, a message is generated and sent from the sender node within the simulator.In a real MMO application a user would provide the requests. If an ID cannot be found in the network the modified protocol will research using the successor list of the node (where the node data is replicated to).

2) The Game Protocol: is used in the product to store game data within nodes. The Game Protocol layer is used to parse messages between layers as well as building initial request messages. The Game Protocol also has the functionality to store successful searches in a local cache at each node.

The Game Protocol receives initial messages from the Traffic Generator. Valid message types can be created in the traffic generator and parsed to the Game Protocol layer where they are automatically handled depending on their type. The Game Protocol also receives replication message instructions from the Replication Protocol to spread data in the network to reduce failures.

3) The Replication Protocol: is a set of methods containing logic to replicate data around the network. Locking and unlocking of data is set up within this layer of the product as well as logic to ensure consistency of data throughout the network (such as pushing new data to all replication nodes). The Replication Protocol layer interacts directly with the data held within the Game Protocol layer. The Replication protocol is explained in greater depth in Section III. A locking and unlocking mechanism is also employed. The locking and unlocking messages can be sent by any node holding a replication with a time released unlock being employed in the event of a node failing whilst the data is locked.

 ISBN: 978-1-61208-238-7

These protocols interact with each other to perform simulated user actions created by the Traffic Generator. These user actions are in the form of object lookup requests given coordinates in the game space which simulate the user moving through a physical 2D space.[24]

The network actions are dealt with by the P2P simulation environment PeerSim [25]. This allows simulation of a real time dynamic P2P network environment with adjustable node failure and churn rate.

The list of game objects are assigned IDs based on four different mapping techniques: MD2, MD5, X-Axis linear curves and Hilbert Curves. The methods are explained in more detail in the following section.

IV. Comparative Analysis of ID Mapping Algorithms

A comparative analysis of the ID mapping and replication techniques described was conducted within the simulation described in section II. The effectiveness is measured in terms of Chord hops, node failures and node stabilisations:

4) Hops: The number of hops that take place within the simulation model from the sender node to the receiver node determines search time. The simulator calculates three metrics for performance comparison:

- **Max Hop** Maximum number of hops taken by the simulator to find data at a receiver node in the network.
- **Min Hop** Minimum number of hops taken by the simulator to find data at a receiver node in the network.
- **Mean Hop** Average number of hops over a complete simulation.

The lower the number of hops the faster data can be retrieved in the network. This results in higher performance.

5) Stabilisations: A stabilisation takes place in the network when data within the DHT is changed by a node. The DHT is changed when a node leaves the network or a new node joins the network. A stabilisation consumes both network and hardware resources (CPU cycles, memory) as it involves querying the nodes on the network and changing values within a node dependent on queries made.

6) Failures: A failure occurs in the network when the sender node fails to find the data being queried. Failures can occur on the network when:

- Data has not been replicated and the master node has left the network.
- A collision in the ID space has occurred (only observed when hash collisions have occurred).
- The ID cannot be found within the network ID space.
- A failure in the (simulated) physical network occurs.

Having a high number of failures is undesirable as it gives a player an inaccurate representation of the MMO game world. In having a high number of failures a player may not be able to retrieve character information leading to a poor overall game experience.

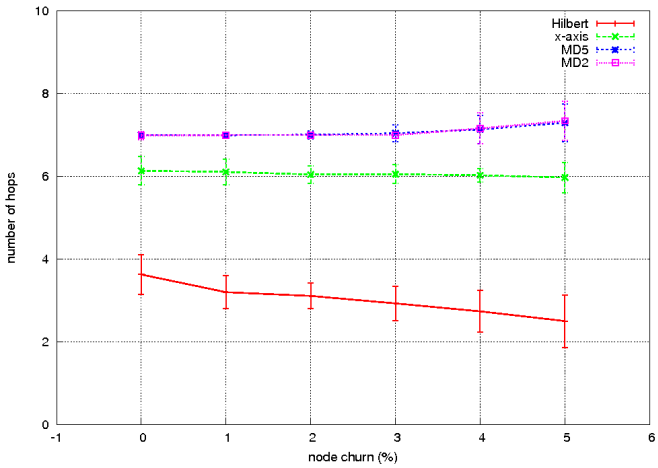


Fig. 4. Average Hop rate per cycle vs churn rate

A. Simulation 1

A network of 10,000 nodes was created and the network initialised on top of this. The node ID was created by MD2, MD5, X-Axis Locality and Hilbert indexing to produce four different data sets. For each data set the churn rate was increased from 0-500 nodes (0-5%) per cycle in increments of 100. At each of these increments the hop count and node stabilisations were recorded. The results from each data set were gathered 100 times and were averaged for each variable.

B. Results for Simulation 1 Scenario

In using X-axis locality, the average number of hops required to find data on the P2P network is reduced (see Figure 4), and in using Hilberts curve the number of stabilisations can be reduced (see Figure 6). In reducing stabilisations the amount of network traffic is also reduced which in a real physical network would result in better performance (less bandwidth) and a decrease in physical resource utilization at each node (memory, CPU cycles).

When searching within a local area within the game application the performance of Hilberts curve increases on average (see Figure 4), this is a result of the locality preserved within the logical ID space[26][27]. A local cache of network addresses is used which increases in size over time. This is independent of churn resulting in a decrease of hops over time.

C. Simulation 2

A network of 10,000 nodes was created and the network initialised on top of this. The node ID was created by MD2, MD5, X-Axis Locality and Hilbert indexing to produce four different data sets. For each data set the churn rate was increased from 0-500 nodes (0-5%) per cycle in increments of 100. At each of these increments the failure rate was recorded. The results from each data set were gathered 100 times and were averaged for each variable.

D. Failure Rate

Each ID generation method was tested where no replication mechanism is used (figure 5). These results show that when

 ISBN: 978-1-61208-238-7

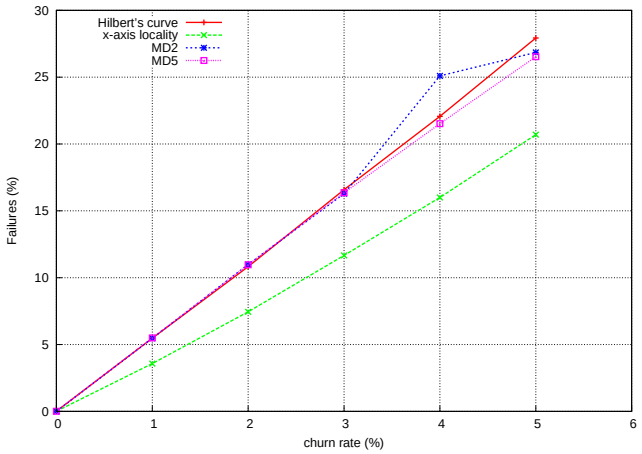


Fig. 5. Average failure rate per cycle vs churn rate: no replication strategy

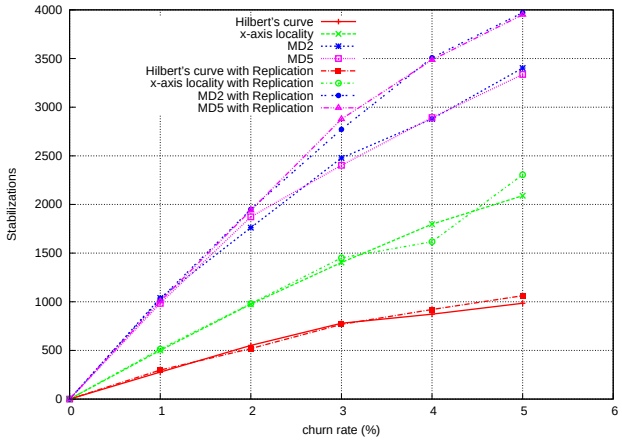


Fig. 6. Average stabilisations per cycle vs churn rate

using the Chord protocol without replication strategy the number of failed queries increases linearly with the churn rate.

When using replication in the network the number of failures is reduced to a negligible number for all methods of ID generation, regardless of the churn rate (up to 50%).

Replication also improved the hop performance of using Hilberts curve to map IDs in a logical space when using a random query. In doing so Hilberts curve performs as well as X-axis locality (see Figure 4). By searching a local area, performance is still significantly faster (also seen in Figure 6). Stabilisations are unaffected by the use of replication, with Hilberts curve ID mapping still offering the most significant performance increase. When using replication a higher number of messages is sent to distribute data in the network which results in more data being sent across the network. The initial testing was carried out in a random manner to test performance in a generalized application. Later testing using geographical proximity was used to test whether locality had a bearing on the lookup time of data in a P2P network.

In using locality aware IDs, the number of hops in a P2P network can be reduced as well as the number of stabilisations required. Within a game application locality offers a significant performance enhancement (see Figure 4). The failure rate however is mostly unaffected and requires another technique to improve performance.

The performance of a P2P network can be beyond that of random IDs with and without replication. This is done using a combination of locality aware ID mapping in a logical ID space and replication.

V. Conclusion and Future Work

This paper explored two aspects of the implementation of a MMOG over P2P protocols. We showed how using a locality aware ID mapping when performing Interest Management in a logical space can have a positive effect on the stability and efficiency of a DHT based P2P protocol. This is measured in terms of number of hops and stabilisations. Specifically, we show that by using Hilberts curve to map IDs with respect to the regions of interest within a logical game space hop performance in a DHT can be doubled (on average). We additionally outline and implement a data replication scheme and show how it leads to the number of failures in a P2P network being reduced. Replication also reduced the number of hops (on average) when using; Hilberts curve, MD2 and MD5 to map IDs to a logical space. In general, when using Hilberts curve as ID generation for Interest Management, in combination with the replication method we propose, the network performance is improved (over the other implementations explored). This should be of consideration to developers of P2P MMOGs who are in need of a node ID scheme for a DHT implementation and data replication strategy. Future work will look at a more extensive implementation of MMOG features such as global game-state management and player communication and the problems these pose for a P2P implementation.

References

[1] J. Mulligan and B. Patrovsky, *Developing online games: An insider's guide*. New Jersey: New Riders, March 2003.

[2] B. Woodcock, "An analysis of MMOG subscription growth," *MMOGCHART.COM [On-line]*, vol. 24, 2008, available = http://tinyurl.com/d6kpl5s [May 24, 2012].

[3] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. IEEE, March 2004.

[4] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, pp. 30–31.

[5] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling massively multi-player online gaming applications on a p2p architecture," in *Proceedings of the IEEE international conference on information and automation*. IEEE, 2005, pp. 7–12.

[6] M. Ratti, S.and Pakravan and S. Shirmohammadi, "A distributed latency-aware architecture for massively multi-user virtual environments," in *Haptic Audio visual Environments and Games, 2008. HAVE 2008. IEEE International Workshop on*. IEEE, 2008, pp. 53–58.

[7] K. Morse, "Interest management in large-scale distributed simulations," Department of Information and Computer Science, University of California, Irvine, California, Tech. Rep. 96-27, 1996.

[8] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," in *Proceedings of the 1st workshop on Network and system support for games*. ACM, 2002, pp. 14–22.

 ISBN: 978-1-61208-238-7

[9] J. Lee, H. Lee, S. Kangm, S. Kim, and J. Song, "Ciss: An efficient object clustering framework for dht-based peer-to-peer applications," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 51, no. 4, pp. 1072–1094, 2007.

[10] S. Singhal and M. Zyda, "Networked virtual environments: design and implementation," *Recherche*, vol. 67, p. 2, 1999.

[11] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, and P. Barham, "Exploiting reality with multicast groups," *Computer Graphics and Applications, IEEE*, vol. 15, no. 5, pp. 38–45, 1995.

[12] T. Funkhouser, "Ring: a client-server system for multi-user virtual environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM, 1995, pp. 85–95.

[13] H. Abrams, K. Watsen, and M. Zyda, "Three-tiered interest management for large-scale virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 1998, pp. 125–129.

[14] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[15] R. Rivest, "The md5 message-digest algorithm," *Internet activities board*, vol. 143, 1992.

[16] F. PUB, "Secure hash standard," *Public Law*, vol. 100, p. 235, 1995.

[17] B. Kaliski, "The md2 message-digest algorithm," RSA Laboratories, Cambridge, Massachusetts, Tech. Rep. 1319, 1992.

[18] R. Merkle, "Secrecy, authentication, and public key systems." *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, 1979.

[19] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Mathematische Annalen*, vol. 36, no. 1, pp. 157–460, 1890.

[20] A. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, pp. 314–330, 1968.

[21] D. Hilbert, "Ueber die stetige abbildung einer line auf ein fluchenstuck," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891.

[22] G. C. and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Transactions on Image Processing [On-line]*, vol. 5, no. 1, pp. 794–797, Jan 1996, available = http://tinyurl.com/c9qse9k [May 24, 2012].

[23] 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.

[24] A. Montresor and M. Jelasity, "A walkable kademlia network for virtual worlds," in *INFOCOM'09 Proceedings of the 28th IEEE international conference on Computer Communications Workshops*. IEEE, 2009, pp. 313–314.

[25] M. A. and M. Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 99–100.

[26] Z. Gharib, M. Barzegar and J. Habibi, "A novel method for supporting locality in peer-to-peer overlays using hypercube topology," in *ISMS '10 Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation*. ACM, 2010, pp. 391–395.

[27] B. Ratti, S. Hariri and S. Shirmohammadi, "Nl-dht: A non-uniform locality sensitive dht architecture for massively multi-user virtual environment applications," in *ICPADS '08 Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*. ACM, 2008, pp. 793 –798.

 ISBN: 978-1-61208-238-7

Fingerprint Verification using Cloud Services with Message Passing Interface over PC Clusters

Fazal Noor, Majed Alhaisoni*, Antonio Liotta+*
*Computer Science and Software Engineering Department
University of Hail, Saudi Arabia
+Department of Electrical Engineering and
Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
f.noor@uoh.edu.sa, majed.alhaisoni@gmail.com, aliotta@tue.nl

***Abstract*—Nowadays cloud-computing services are being offered by various organizations. Peer-to-Peer (P2P) networks can be used as a collaborative computing environment to solve computationally intensive problems. In this work, we use a PC cluster to simulate a P2P network and present results of a computationally intensive image matching algorithm (fingerprint verification). Collective communications are used to transfer images to destination peers over a network. Communication to computation time ratio are calculated of transferring of fingerprint images of various sizes on the internet. As transfer of raw images are communication intensive, a proposed method is to use FBI approved Wavelet Scalar Quantization (WSQ) compression method at the source before transmitting to the destination nodes. We study the viability of fingerprint identification and/or verification service offered by cloud computing. In particular, we present a distributed fingerprint verification algorithm.**

Keywords-PC Cluster; Normalized Correlation Method; Minutia Method; Cloud Services; Latency; Bandwidth; Message Passing Interface; Phase Correlation method; Log-Polar; communication to computation ratio.

I. INTRODUCTION

Cloud-computing services are becoming common nowadays [1],[2]. There are many types of services being offered to customers of cloud-computing. As the demand for cloud computing grows, different types of applications appear, which require intensive computing power. In such cases, PC clusters are more affordable and a cheaper alternative for buying supercomputers, which are very costly.

Authentication of a person is required to access places of high security and can be done in several ways, verification by knowledge such as passwords, verification by possession such as id cards or passports, or verification by biometrics such as fingerprints [3],[4]. Authentication can be done locally, such as access to restricted areas, Personal Digital Assistants (PDAs), computers, etc. Here we look at authentication done remotely via cloud computing and therefore security issues which arise are of major concern, especially in keeping the data secure. Remote verification is necessary when the original fingerprint is stored at a remote site. One such application is verification by possession and by biometrics (fingerprints), which may be required at airports, border points, checkpoints, etc. All these check points require verification by possession and for further confirmation verification by biometrics may be done with the use of mobile wireless devices. Travel safety is a major concern not only for the governments but also for any person for example boarding a plane. Security has been a great issue at airports due to terrorist activities. Currently, verification is by possession of a valid passport. Consider the following scenario at an airport, verification of passengers' identities by fingerprint biometrics also. Each passenger goes through a checkpoint and his/her fingerprint is scanned by an ultrasonic scanning device. The fingerprint(s) are sent with a tag (person's identity number and country code for fast look up) for verification at data centres which may be distributed around the world. There the tag is used to retrieve from database the person's fingerprint and verification made. Since a passenger at airport A in country A may be a citizen of another country, say B, and therefore normally his enrolled fingerprint would be enrolled in the database located in his/her country B. Every country has laws protecting the privacy and security of their citizens. The scanned fingerprint image has to be transmitted through the internet to the country to verify the identity of a passenger. A database of fingerprints maybe in hundreds of millions or billions depending on the population of the country. The fingerprint image size *NxM* varies depending on the device used. For a size of 768 x 768 at 500 dpi, assuming 8 bits grayscale image, the amount of storage would be 589,824 bytes [7]. Therefore, database size would require a total of image size x population size of storage bytes.

 ISBN: 978-1-61208-238-7

The problem is to authenticate the person based on his fingerprint scanned and sent for verification by cloud-computing services. One question is how much time would it take for the verification result to arrive back (i.e., the time to communicate fingerprint image plus person's data and time to process for verification). The time then depends on part where the databases are located. Ideally one location containing all countries fingerprints is preferred, but, in reality, due to various reasons, such as privacy issues and security concerns would be located at multiple locations. Another question is how reliable and accurate is the verification. Can we trust the result which comes from another country? Assuming each country would have their own database, therefore *J* scanned fingerprints would have to be transmitted via cloud-services points to *K* countries for verification of an identity of a person as in Figure 1. How accurate is the verification result? Fingerprints are considered sensitive information by any government and should not fall into the wrong hands. Another scenario is the person carries a biometric card which holds the person's fingerprint already enrolled. Then, both the scanned fingerprint image and the one on the biometric card is sent to a cloud service point for verification and result communicated back as "pass/fail". Reliable service is one of the key goals. This paper focuses on two key issues *Speed* and *Accuracy* of fingerprint verification. For *Speed*, communication bound and compute bound problems of fingerprint image(s) identification or verification at distributed databases are explored. The main contribution in this paper is presenting a distributed algorithm based on correlation method using 9 patches for high accuracy of fingerprint verification.

The paper is organized as follows: in Section II, some related work is presented. In Section III, fingerprint background is presented, and, in Section IV, methods are presented. Section V shows the results, Section VI presents the discussion, and in Section VII, conclusion and future direction are given.

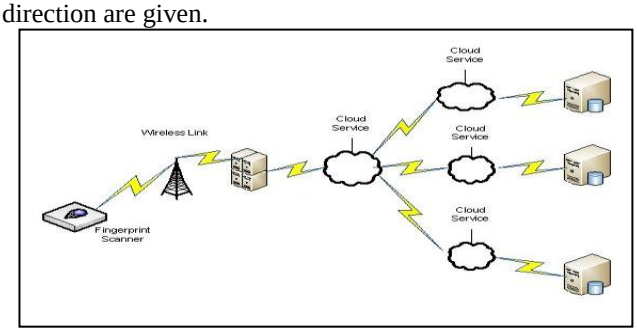


Figure 1. Fingerprint verification through peer Cloud Services to peer Cloud Services located around the world.

II. RELATED WORK

Work on fingerprint identification and verification is vast in the literature. There are many algorithms appearing in the literature mainly based on minutia properties [3]-[7]. In this work, we mainly use a correlation method for verification for several reasons. One reason is that we store the original fingerprint image (using Wavelet Scalar Quantization compression technique, a standard set by FBI) [7] and not only the extracted features as done by methods based on the minutias. Many algorithms in literature are fast and have high reliability, but are not 100% accurate [3] [4]. Researchers have also proposed combination of biometric methods, for example fingerprint plus voice recognition [6]. Government of South Africa is allowing banks to have access to fingerprint database [8]. Chang et al. [9] have implemented a real-time video/voice over IP (VVoIP) applicatons on a Hadoop cloud computing system [15]. For access control, to prevent illegal intrusions, they have used facial recognition and fingerprint identification via cloud computing. It takes about 2.2 seconds to exactly identify the subject [9]. In our work, we present a distributed fingerprint verification algorithm and the fingerprint database are not local to cloud-computing services but within the domain of each country.

III. FINGERPRINT BACKGROUND

Fingerprint-based identification is one of the oldest biometric technique [6]. A fingerprint consists of three-dimensional lines called ridges and the spaces between them are called valleys. Fingerprint identification is different from fingerprint verification. In identification, the question is to answer whose fingerprint is this. In verification, the question is, are you who you claim to be. In fingerprint identification, a large database has to be searched and match is of a form 1:N, whereas in verification the original image is to be matched with the live scan image and the match is of 1:1 form. Time required for identification is much larger than the time for verification.

As raw fingerprint image storage demands large amount of memory space; usually, images are not stored, but their properties are stored such as minutia type, etc. Here Wavelet Scalar Quantization (WSQ) [7], a compression technique developed by the FBI is used to compress images for transmittance or storage. WSQ has a better preservation of fine details over other compression techniques. A fingerprint image of 589284 bytes is compressed to WSQ image of size 45621 bytes, a compression ratio of 12.9 [7].

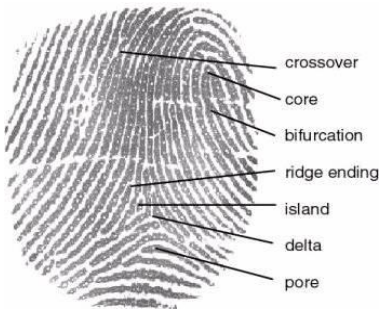


Figure 2. Fingerprint showing various features labeled [6].

 ISBN: 978-1-61208-238-7

IV. METHODS

A. Fingerprint Methods

Algorithms use various techniques at different stages for example, during preprocessing (using segmenatation, or enhancement), during alignment (before matching, during matching, displacement, scale, rotation), during feature extraction (e.g., minutia, singular points, ridges, counts, orientation field, texture measures, raw/enhanced image parts), and during comparisons (based on minutiae global, local, ridge pattern geometry or texture, and correlation) [3]. All categories fall under estimation and approximation theory and the main ones are summarized below:

Minutiae Based Method [3]

1. Image acquisition or capture by a device. Objective is to capture image of the center of the finger as this part contains unique features. There are different technologies available in the market. The three main are optical, silicon, and ultrasound. Ultrasound is better than the other two.

2. Extracting unique characteristics of the fingerprint and their locations. A fingerprint consists of various ridges and valleys and formation of loops, arches, and swirls. Minutiaes are extracted which are of mainly two type a. ridge endings and b. bifurcations.

3. Creation of minutiae template: Type, location, position, quality, direction of ridges, etc.

4. Template matching between enrollment template with the verification template.

Correlation Based Method [6]

The mean-square difference is defined as

$$E(p,q) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}[G(i,j) - H(i+p, j+q)]^2 \quad , \qquad (1)$$

where $G(i,j)$ is the feature image and $H(i,j)$ is the original image. For an ideal case, where there is no noise in the images, an exact match will make E zero for all possible points. In practice, E would not be zero due to noise in the original image and the feature image. Therefore, a decision is made on a comparison with a threshold level $T(p,q)$. If $E < T$, then the decision is said to be a match has occurred otherwise the decision is no match. In practice, the Normalized Cross-Correlation (NCC) given below [6],

$$R(p,q) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G(i,j)H(i+p, j+q)}{\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G^2(i,j)\right]^{1/2}\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} H^2(i+p, j+q)\right]^{.1/2}}, \quad (2)$$

is used. When evaluating algorithms the maximum limits on algorithms are imposed such as time limits of enrollment and comparison and size limits on template and memory. Typical values maximum limits used in fingerprint verification competition are shown in Table 1 [3].

TABLE I. FINGERPRINT VERIFICATION COMPETITION LIMITS.

Maximum Limits on Algorithms				
	Enrollment seconds	Comparison seconds	Template KBytes	Memory MBytes
Open Category	10	5	No limit	No limit
Light Category	.5	.3	2	4

The minimum and maximum values depend on type of databases; see [3], for further details.

B. Distributed Fingerprint Verification Algorithm

Most algorithms are of minutia type compared to correlation type. Here we chose to implement algorithm based on correlation method as it does not require much preprocessing as compared to minutia type. With preprocessing in minutia type algorithms there is a possibility of false registrations of minutias due to poor image quality of fingerprints.

A cloud-service point may use a PC cluster to speed up a fingerprint verification. The steps of an algorithm based on correlation method are outlined as follows:

1. WSQ compression is performed on raw fingerprint image, tagged, and sent to cloud services point.
2. The tag is used to find the original image from the database.
3. A procedure is used to extract 9 feature patches of size ***N x N*** pixels from the original image and their locations are marked and stored. The 9 patches are selected toward the center of fingerprint image and equal distance apart. Core or delta points would be preferable to be included in one of the patch.
4. Scatter ***P=9*** feature patches and their locations to each node in a PC cluster. Assume the original image is accessible by all nodes in a PC cluster.
5. Each node tries to detect the patch on the scanned fingerprint image by calculating the normalized cross-correlation equation (2) and sends the result back to master node.

 ISBN: 978-1-61208-238-7

6. If majority of patches match with those in the scanned image, the scanned image is super-imposed on the original image. If there is mis-alignment, further techniques, such as Phase Correlation Method combined with log-polar method (translation and rotation) are used to align the image. The normalized correlation value is compared with a threshold *T*, and if it is greater than or equal to 0.9, is considered a match. The neighboring patches may also be checked to achieve even greater accuracy.
7. If majority of patches do not match, declare a non-match.
8. Send the result "pass" or "fail" back to source.

There are two types of errors that may occur in matching fingerprints, one is named a False Match rate FMR (False Acceptance) and the other False Non-Match FNMR (False Rejection). A trade-off between the two errors exists, depending on the threshold *T*. The point where FMR equals FNMR is called the Equal Error Rate.

V. RESULTS

A. Beowulf PC cluster Specifications

Our test-bed consists of a PC cluster including 20 Lenovo machines with the following specifications: Intel Core™ 2 Duo CPU, E4400 2.00GHz, 1.00 GB of RAM. Network Card: Broadcom Netlink, Gigabit Ethernet, Driver date 8/28/2006 version 9.81.0.0. The PCs are connected to a Gigabit D-Link Ethernet switch. Each machine has a RedHat Enterprise AS Linux operating system installed, and uses LAM 7.0.6/MPI 2.

B. Model

A PC cluster of size 20 nodes is used to model 20 airports. Each airport is located in a different country. Also assume each node in a cluster which represents an airport is also a processing centre (cloud service point). The cloud service point is assumed to have access to fingerprint databases around the world. The link between any two nodes may contain a number of routers with different latencies and bandwidths.

C. Experimental result

Here, we present results of our algorithm using 10 nodes in a PC cluster. Given an original fingerprint of size *N x M*, partition the image into *B* blocks each of size *N/patch size*. For example, if *N*=512, then for patch size of 16 pixels there are *B*=32 blocks, choose 9 patches located approximately in the center, for example, blocks no. (12,12), (12,15), (12,18), (15,12), (15,15), (15,18), (18,12), (18,15), and (18,18). Other ways can be used to choose the patches, for example, choosing a patch at the core or delta. The 8 others chosen equidistant from the center patch enclosing the core or delta point. Also, the number of patches selected is arbitrary, however we choose 9 to achieve greater accuracy. The master node uses Message Passing Interface (MPI) to scatter the scanned fingerprint and the 9 patches to the 9 nodes in a cluster. Each node searches in the neighborhood of each block by sliding the patch pixel by pixel to find maximum normalized correlation. The database of fingerprints consisted of 20 people with 3 prints of the same finger for a total of 60 images. Figure 3 shows a sample of two prints of the same finger with the second print clipped on the top. Figure 4 shows the 9 patches selected of size 32x32 pixels and Figure 5 shows a sample of normalized correlation. We experimented with different patch sizes of 16x16, 32x32, and 64x64 pixels. Patch sizes of 64x64 pixels exhibited sharpest peak in comparison to peaks obtained by using patches of sizes 16x16 and 32x32 pixels. The FAR for threshold values of *T* =0.85 was 0.02% for the patch size of 32x32. For patch size of 16x16, FAR starts to occur when *T* was set to a value of 0.8. For patches of size of 32x32 and 64x64, FAR starts to occur when *T* was set to a value of 0.7. The FRR was 11.3%, which is due to images having rotation. It is observed 9 patches are more than sufficient to discriminate an impostor from the genuine fingerprint.

Figure 3. Two fingerprint impressions from the same finger.

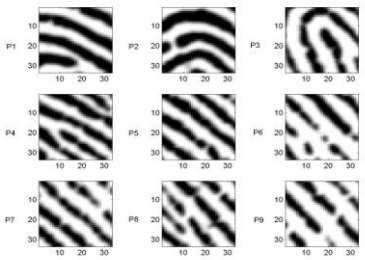


Figure 4. Nine patches of size 32 x 32 pixels.

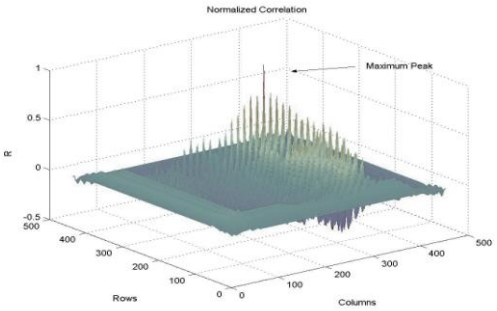


Figure 5. Sample maximum peak obtained by normalized cross-correlation.

D. Communication and Computation Times

The waiting time of the i^{th} passenger will be the time to scan his/her finger, the time to transmit the fingerprint image with a tag (data such as civil card id number and country code), the time to perform the verification through a cloud service point and the time to receive the result,

$$Time_i = t_{scanA} + t_{sd} + t_{verification} + t_{ds}, \quad (3)$$

where t_{scanA} is the time to scan the fingerprint by a device, t_{sd} is the time taken to transmit the image plus the tag from source A to destination B, $t_{verification}$ time taken by the peer (PC Cluster or Super Computer to perform the computation of verification and depends on the hardware and algorithms used), and t_{ds} the time to receive the result as "Pass" or "Fail". If all passengers are local then local database is accessed, if passengers are international then of course distributed database. Can the result be obtained in real-time? From table 2 the scan time dominates both the communication and compute time. It is important to know how much computation time is in comparison with communication time. There are numerous fingerprint verifications algorithms and increasing in numbers [3], each algorithm having its own speed and accuracy. Communication time would depend on the network links, routers in between, etc. Queue delays at cloud service points. Latency will vary with each path as paths change. Therefore, each path would have a different time. Computation time is function of both algorithm and hardware used. If T_s = time to execute on a single computer then using a PC cluster the time to execute would be approximately T_s / n , where n is the number of computers in a PC cluster. Table 2 shows time to process fingerprint verification by algorithms based on minutia method and correlation method. Correlation method is more compute intensive than minutia method.

TABLE II. TIME TO PROCESS FINGERPRINT VERIFICATION

Algorithms based on	*Approximate Time in seconds*				
	Scan	***Send X_s***	***Process C_s***	***Receive Y_r***	***Total***
Minutia	5 sec	.5	.3	.4	6.2
Correlation	5 sec	.5	1	.4	6.9

VI. DISCUSSION

Fingerprint recognition is one of the oldest and most popular biometric technologies being used in commercial applications. A consensus among countries has to be made on setting standards and how fingerprint databases are to be accessed by cloud-computing service points. Highly secure encryption methods for protection of fingerprint data is a must. Another point is where to store fingerprint databases of any country, so it is securely accessed. Should it be stored at cloud-service premises or in country of origin? This is to protect from any fraudulent use. If not on cloud service premises, the cloud-services points would have to have contracts with each country either to access their fingerprint database or with the countries cloud-computing service. In other words, peer cloud-computing to peer cloud-computing services [10] [11]. The services should be highly reliable, secure, and results with ideal accuracy of 100%.

Compute bound problems: Each country may be using different algorithms of extracting fingerprint features and storing fingerprint data. Therefore, different algorithms with various execution times depending on complexity and type of hardware. Here, the goal would be reduction of search time and low computational complexity with high accuracy of verification.

Communication bound problems: There are many, such as encryption and compression of raw images, and secure transmittance over the internet. A universal tag has to be agreed among countries for easy retrieval of fingerprint data from a large database. In networks, data is not sent in a continuous stream, but in packets. Fingerprint image sizes *N x M* vary according to various devices used. Usually, *N>M* with values anywhere from 256 to 768. Therefore, transmission can be anywhere from 64KB and up. Line speeds range from slowest to fastest as shown in Table 3. For example, a 10 MB transfer of data at 9600 bps would take around 3 hours.

Since communication time maybe defined as,

*T = Latency + Bandwidth*message length,* (4)

Latency becomes significant when small size packets are sent often and less significant for large packet size [12]-[14]. Latency would be *d/C,* where *d* is distance and *C* is speed of light $300\ x\ 10^6$ *m/s*. Actual latency over the networks would be larger. Bandwidth indicates the maximum rate at which message can be sent.

 ISBN: 978-1-61208-238-7

Some problems of scanned images: The devices to scan are different in home country and foreign country. Therefore, the fingerprint images of the same person are differently captured. A solution is for all countries to use devices which meet universal standards and specifications.

Each country will have different computing power, for example, different PC cluster with different hardware /software. Each country will have a different algorithm for verification purpose. Therefore, different accuracy and computation times will be observed.

Verification to be done at source or destination ? In this case, image data is received from country of origin and processed with image at source. The problem is that, usually, the image is not stored, but only its properties are stored such as minutia type, location, orientation, etc. Therefore, the method used would most probably be not known.

Verification may be done at destination and more appropriate. In this case, image is sent with a tag to a cloud-service point, which, in turn, would send image to country of origin and processed with image data at the destination using the same method used by that country.

There are country security issues on how to deal with unfriendly countries or uncooperative countries. Images are sent via a third country, compressed and encrypted. In other words, cloud computing is done via a third country. Another question arises on whether can we trust the country's results. Also, on how to deal with non-participant countries.

TABLE III. DIFFERENT LINES AVAILABLE FOR DATA TRANSMISSION.

	Speed of Lines		
	Kbps	Mbps	Gbps
1	9.6	1.024	0.622 (OC12)
2	14.4	1.544 (DS1, T1)	1 (1000Base-T)
3	28.8	2.048 (E1, ISDN-32)	2.4 (OC48)
4	33.6	25.6 (ATM 25)	9.6 (OC192)
5	56	34 (E3)	
6	64 (ISDN)	45 (DS3, T3)	
7	128 (ISDN-2)	51 (OC1)	
8	256	100(100Base-T)	
9	512	155 (OC3)	

VII. CONCLUSION AND FUTURE WORK

In this paper we have looked at various issues concerning fingerprint verification via cloud-computing services. As numerous different devices capture fingerprint images differently, standards are required to ensure image quality is good. Countless fingerprint verification algorithms are appearing and have to ensure that they meet very near 100% accuracy. Biometric methods are probabilistic methods with decision based on estimations. Accuracy of current algorithms in the literature is not 100%. In our experiments, the threshold values depends on the patches selected. We presented a distributed fingerprint verification algorithm based on normalized correlation. We looked at communication issues such as secure exchange of fingerprint data through the internet. Computation issues such as high accuracy and reliable methods of verification. As communication time is higher than computation time, it would be more appropriate to use multi-core computers rather than a PC cluster. Peer cloud services to peer cloud services would be required to have secure contracts among each other. WSQ would be used for transmitting fingerprint images through the Internet and storage.

ACKNOWLEDGMENT

The authors would like to thank CSSE Research Centre at University of Hail for carrying out experiments on the PC cluster. This work was supported by University of Hail Research Centre grant no. 1433/6/15 and in partially supported by ARTEMIS project DEMANES (Design, Monitoring and Operation of Adaptive Networked Embedded Systems, contract 295372).

REFERENCES

[1] K. R. Jackson, et. al., "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud", Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference, Nov. 30 2010-Dec. 3 2010, pp. 159-168.

[2] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using Amazon Elastic Computing Cloud", ACM 2008 The Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) Conference, pp. 94-103, 2008.

[3] R. Cappelli, et al., "Performance Evaluation of Fingerprint Verification Systems", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 1, pp. 3-18, January 2006.

[4] D. Maltoni, "A Tutorial on Fingerprint Recognition", Advance Studies in Biometrics, pp. 43-68, 2003.

[5] R. Cappelli, et al, Fingerprint Verification Competition at International Joint Conference on Biometrics (IJCB2011), Washington DC, 2011.

[6] L. O'Gorman, Chapter 2 Fingerprint Verification, Biometrics : Personal Identification in Networked Society by Anil K. Jain, Ruud Bolle, Sharath Ankanti, pp. 1-20, Springer 1999.

[7] http://www.c3.lanl.gov/~brislawn/FBI/FBI.html, [retrieved: 6, 2012]

[8] http://saitnews.co.za/e-government/supports-banks-access-fingerprint/, [retrieved: 6, 2012]

[9] B. Chang, et. al., Adaptive Performance for VVOIP Implementation in Cloud Computing Environment, LNCS, 2012, vol. 7198/2012, pp 256-365, 2012.

[10] M. Li, W. Lee, A. Sivasubramaniam, "Efficient Peer-to-Peer Information Sharing over Mobile Ad Hoc Networks", In MobEA, pp. 1-6, 2004.

[11] X. M. Huang, C.Y. Chang, M.S. Chen, "PeerCluster: A Cluster Based Peer-to-Peer Sytem", IEEE Transactions on Parallel and Distributed Systems", vol. 17, No. 10, Oct. 2006, pp. 1110-1123.

 ISBN: 978-1-61208-238-7

[12] M. Matsuda, T. Kudoh, Y. Ishikawa, "Evaluation of MPI Implementation on Grid-connected Clusters using an Emulated WAN Environment", Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03), pp. 10-17, 2003.

[13] G. Huston, "Measuring IP Network Performance", The Internet Protocol Journal, vol. 6, no. 1, March 2003, http://www.cisco.com/ipj , [retrieved: 6, 2012]

[14] F. Noor, M. Alhaisoni, and A. Liotta, "An Empircal Study of MPI over PC Clusters", The Third International Conference on Advances in P2P Systems, AP2PS 2011, November 20-25, Lisbon, Portugal, pp. 65-70, 2011.

[15] http://hadoop.apache.org, [retrieved: 6, 2012]

 ISBN: 978-1-61208-238-7