ICIMP 2011

The Sixth International Conference on Internet Monitoring and Protection

March 20-25, 2011

St. Maarten, The Netherlands Antilles

ICIMP 2011 Editors

Go Hasegawa, Osaka University, Japan

Constantion Paleologu, University 'Politehnica' Bucharest, Romania

ICIMP 2011

Foreword

The International Conference on Internet Monitoring and Protection (ICIMP 2011) held on March 20-25, 2011 in St. Maarten, The Netherlands Antilles, continued a series of special events targeting security, performance, vulnerabilities in Internet, as well as disaster prevention and recovery. Dedicated events focused on measurement, monitoring and lessons learnt in protecting the user.

Internet and Web-based technologies led to new frameworks, languages, mechanisms and protocols for Web applications design and development. Interaction between web-based applications and classical applications requires special interfaces and exposes various performance parameters.

The design, implementation and deployment of large distributed systems are subject to conflicting or missing requirements leading to visible and/or hidden vulnerabilities. Vulnerability specification patterns and vulnerability assessment tools are used for discovering, predicting and/or bypassing known vulnerabilities.

Vulnerability self-assessment software tools have been developed to capture and report critical vulnerabilities. Some of vulnerabilities are fixed via patches, other are simply reported, while others are self-fixed by the system itself. Despite the advances in the last years, protocol vulnerabilities, domain-specific vulnerabilities and detection of critical vulnerabilities rely on the art and experience of the operators; sometimes this is fruit of hazard discovery and difficult to be reproduced and repaired.

We take this opportunity to thank all the members of the ICIMP 2011 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 who dedicated much of their time and efforts to contribute to the ICIMP 2011. 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 are grateful to the members of the ICIMP 2011 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICIMP 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in Internet Monitoring and Protection

We are convinced that the participants found the event useful and communications very open. The beautiful places of St. Maarten surely provided a pleasant environment during the conference and we hope you had a chance to visit the surroundings.

ICIMP 2011 Chairs
Go Hasegawa, Osaka University, Japan
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Seppo Heikkinen, Tampere University of Technology, Finland

ICIMP 2011

Committee

ICIMP Advisory Chairs

Go Hasegawa, Osaka University, Japan
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Seppo Heikkinen, Tampere University of Technology, Finland
Guillaume Valadon, French Network and Information and Security Agency, France
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Constantion Paleologu, University 'Politehnica' Bucharest, Romania
Michael Grottke, University of Erlangen-Nuremberg, Germany

ICIMP Publicity Chairs

Eduardo Cerqueira, Federal university of Para, Brazil
Gerardo Fernández-Navarrete, University of Málaga, Spain
Ejaz Ahmed, Queensland University of Technology (QUT)- Brisbane, Australia

ICIMP 2011 Technical Program Committee

Jemal H. Abawajy, Deakin University, Australia
Ejaz Ahmed, Queensland University of Technology (QUT)- Brisbane, Australia
Basheer Al-Duwairi, Jordan University of Science & Technology, Jordan
Manos Antonakakis, Georgia Tech, USA
Javier Barria, Imperial College London, UK
Lasse Berntzen, Vestfold University College - Tønsberg, Norway
Jonathan Blackledge, Warsaw University of Technology, Poland
Dario Bottazzi, Laboratori Guglielmo Marconi, Italy
Matthias R. Brust, Technological Institute of Aeronautics, Brazil
Kevin Butler, University of Oregon, USA
Christian Callegari, University of Pisa, Italy
Marco Casassa Mont, Hewlett-Packard Laboratories, UK
Eduardo Cerqueira, Federal university of Para, Brazil
Richard Chow, PARC, USA
Denis Collange, Orange-tfgroup, France
Christopher Costanzo, U.S. Department of Commerce, USA
Peter Danielis, University of Rostock, Germany
Jianguo Ding, University of Luxembourg, Luxembourg
Martin Drotleff, Commerzbank AG - Frankfurt am Main, Germany
Matthew Dunlop, Virginia Polytechnic Institute and State University, USA
Mohamed Eltoweissy, Pacific Northwest National Laboratory, USA
William Enck, Penn State University, USA
Gerardo Fernández-Navarrete, University of Málaga, Spain
Nicolas Fischbach, COLT Telecom, Germany
Ulrich Flegel, Offenburg University of Applied Sciences, Germany

Table of Contents

Doubletree With Many Sources

Tony McGregor
Computer Science Department
The University of Waikato
Hamilton, New Zealand
tonym@cs.waikato.ac.nz

***Abstract*—Most active measurement projects are limited by the number of measurement points and, consequently, the number of perspectives they have on the Internet. The goal of the RIPE NCC Atlas project is to deploy up to 100,000 active measurement monitors around the Internet. An extended version of Hubble, which finds routing "black holes" is a motivating application. Increasing the number of measurement points by two orders of magnitude requires new measurement approaches. For example, Atlas Hubble needs to perform traceroute type path discovery from many sources to a small number of destinations. It is important to optimise the load placed on the destination monitors, especially if they are located at the edges of the Internet. Doubletree is a path discovery optimisation technique that may be applicable. To date, Doubletree has only been investigated for a small number of sources to many destinations. This paper reports on a simulation study of Doubletree for the many sources, few targets case. Initial results indicate that Doubletree may be very effective in this case but further work is needed to understand the impact of the sharing of coordination information.**

***Keywords*-Doubletree; traceroute; active measurement;**

I. INTRODUCTION

This paper explores the effectiveness of the Doubletree [1] algorithm in a Hubble-like [2] application where there are up to 100,000 monitors. Hubble looks for routing "black holes" in the Internet. A black hole is defined as a routing failure that persists for at least 15 minutes where some, but not all, parts of the Internet can not reach a given destination. The system operates in two main modes: target *discovery* mode, where "reachability events" of this type are found, and reachability *analysis* mode, where as much detail as possible about the location of the fault is found. In broad terms, discovery mode operates using reachability measurements while analysis mode uses path discovery.

To utilise a measurement system like Hubble in an environment where there are up to two orders of magnitude more monitors than existing measurement systems requires special attention to the load created by the measurement on the network. This paper looks at the part of the Hubble load that occurs once a reachability event has been found. In this phase, Hubble investigates the event by sending traceroute style probes from all its vantage points to the target of the reachability failure. In the case of Atlas/Hubble, there might be tens of thousands of distinct vantage points (e.g., one per Autonomous System). Running the original traceroute from this large number of sources has the potential to overwhelm the destination, especially if the destination probe is located with a home user or mobile device with limited bandwidth. We investigate whether Doubletree can reduce this load.

Doubletree is designed to reduce the number of probes sent when team probing occurs. It reduces the number of probes sent to discover hops close to the source when multiple destinations are explored from the same monitor by starting to probe with a Time To Live (TTL) > 1. Once the path from this mid-point to the destination has been discovered (by sending successive packets with incrementally greater TTLs), Doubletree sends probes with incrementally smaller TTLs starting with the original mid-point TTL less one. Probing stops when a node that has been discovered before is re-discovered. This aspect of Doubletree is useful for the Atlas/Hubble application but it is the other half of the Doubletree methodology (which reduces the number of probes sent to discover hops near the destination) that, at face value, appears particularly useful to Atlas/Hubble.

Doubletree reduces redundant link discovery near the destination in a similar way to how it reduces probing to hops near to the source except that a global list of hops discovered near destinations is shared by all monitors. In the original Doubletree work, simulations with 24 source monitors showed substantial savings. In Atlas, however, there will many more monitors. As a starting point, we assume one trace from every Autonomous System (AS) that hosts an Atlas monitor giving approximately 22,000 sources in our simulation.

Real-time behaviour influences the effectiveness of Doubletree. For example, if several probes reach a link at about the same time the second and subsequent monitors are unlikely to have already received the stop-set information indicating that this link is already known.

The rest of the paper is organised as follows: Section II describes the simulation of Doubletree, including the main assumptions and simplifications. Section III presents the main results of the investigation. Information required to reproduce this work is provided in Section IV. The paper ends with a review of the current results and further work needed to fully understand the use of Doubletree in this context.

II. Simulation

This Section describes the simulation of Doubletree undertaken for this project. It includes the derivation of the input data used and the major assumptions made.

A. Topology

To be realistic, the simulations need a topology that matches the Internet as well as possible. In particular, the length of paths and the branching nature (node in-degree and out-degree) of the topology should be as similar to the real Internet as possible. To this end, the simulator topology used is based on a map of the Internet derived from CAIDA's Archipelago [3] infrastructure running the Scamper [4] tool (from here on referred to as Scamper for simplicity although we note that Scamper can be run in other environments).

Scamper attempts to measure the global topology of the Internet. Traceroute style measurements are taken from a relatively small number of sources to many destinations. Scamper uses the intra-monitor part of Doubletree to reduce the cost of probing hops near a monitor but does not use the inter-monitor (global stop-set) part of Doubletree. The output from Scamper is a set of runs where each run contains a traceroute style probe to every destination address from one member of the team of monitors. A single run started on 3 Jan 2009 from a team of 13 monitors was used as the topology input to the simulator. In future work, we plan to repeat the simulations with other scamper data sets to determine the stability of our results against the particular scamper data set used.

The arrangement of nodes and links alone is not enough to route packets through a network; routing information is also required. In the simulator, routing information is represented in a table of destinations and next-hop at each node. This information is inherent in the scamper data set and we simply maintain it in the simulator topology data structure.

B. Interfaces vs Routers

Scamper, like all traceroute based tools, discovers interfaces not routers; the raw data does not show which interfaces are on the same router. The simulator topology is, therefore, also built in terms of interfaces not routers. This is not problematic for the simulation, which also proceeds in terms of packets being passed from interface to interface. When we refer to nodes in the topology model we are referring to a particular interface (not a router). Similarly, links are between interfaces.

C. Discarded paths

Some of the paths in the scamper data are not usable in the simulator, mostly because they are not well formed. For example, scamper discovers loops in some paths. In others, it abandons tracing because too many nodes do not reply with a TTL expired message. In these cases, a complete path from the source to the destination is not discovered and the path can not be used in the simulation. Of the 5.6 million paths discovered in part or full by scamper, 241,763 (4%) are omitted from the simulation topology because of these reasons.

D. Symmetric Paths

Scamper data is not a complete map of the Internet. In particular, it contains paths from the monitors to the destinations but not the reverse. It also does not measure paths between destinations. The first issue is resolved in the simulator by adding a symmetric path from the destination back to the source. From other work [5], we know that Internet paths are not always symmetric. We do not believe that the symmetric nature of paths in the simulation topology significantly affects our results because it is the overall structure of the Internet (i.e., path lengths and branching) not the exact details of particular paths that is important. However, without a measured non-symmetric topology, we have no way of demonstrating that this is the case.

The omission of paths between destinations is mostly not problematic for the simulator because packets are only sent between sources and destinations.

E. Alternative Paths

In some cases, Scamper discovers alternative paths between nodes within the network. In the Internet, alternative paths may arise because of load balancing or because the topology has changed during the measurement. As a consequence, Scamper may discover different alternatives between the same nodes within the network when it probes between different source/destination pairs.

In the simulator topology model, alternative paths between nodes are maintained. The same path variant is used when packets are sent from a source to a destination as was discovered when scamper measured the route between the two. In the simulator topology data structure, this is done by including a source as well as the destination in the next-hop table at a node.

This approach does not necessarily match the behaviour of the Internet in all cases, however it is likely to be correct in most cases. If the source of the alternative paths is a path change during scampers probing, either path is acceptable for the simulation. It is not required that we maintain both paths in this case but it is acceptable. In the case where there are alternative paths due to load balancing, using the same path as the one scamper discovered for this source/destination pair will mostly match the behaviour of the Internet. This is because per-destination and per-flow load balances are most common in the Internet while per-packet load balances are rare [6].

F. Missing hops

During traceroute style probing, it is common for some hops to not reply with a TTL expired message. Often the

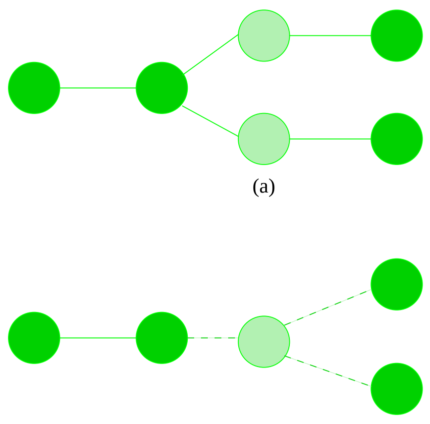


Figure 1. Modelling Missing Hops

hop is known to exist because later hops do respond but the address of the hop can not be discovered. Approximately 22% of hops are not identified in the traces we used. Within the simulator, these non-responding nodes are given a unique address. The addresses are constructed so that they do not collide with the IP addresses that most nodes have. To support this, the simulator has two classes of address, an IP address (printed as a doted quad) and a "missing" address (printed as an "m" followed by a unique id).

This procedure does not exactly replicate the structure of the Internet at the time scamper was probing because it is possible that a missing hop in two different paths might be the same interface but this approach inserts two different interfaces (see Figure 1). From the Scamper data, it is not possible to tell which scenario is correct.

G. Missing Topology

Although it is the most extensive macroscopic Internet topology discovery system available, Scamper does not discover all links in the Internet. The extent of missing topology is not currently known. What Scamper discovers is the path used from the vantage points it has at the time the measurement was taken. Because the simulation is based on edges measured by Scamper, the simulation matches how packets were routed at that time. If the paths Scamper finds are typical of all Internet paths, the missing topology has no significant impact on the simulations.

H. Selection of Endpoints

The simulation uses a sub-set of the possible end-points in the topology. The selection is based, in part, on the AS (Autonomous System) that an endpoint belongs to. The translation from IP address to AS number (ASN) is based on data collected by the RIPE NCC Routing Information Service (RIS) [7]. RIS data contains route dumps from specially deployed routers that passively peer with operation routers in many ISPs. The routing data collected describes Internet routing as a set of elements with an IP address range and a list of ASs that traffic may be router through to reach IP addresses in the range. For more details see [7]. A database of address ranges and the last ASN in the path was created from RIS data from 30 March 2009 to allow IP addresses to be matched with a host AS. On occasion, there is more than one last hop ASN for a given IP address. In this case, the first ASN discovered in the data set is used.

For simulations of many sources to a few destinations, the Scamper source monitor addresses are used for the *destination* addresses. One source address present in the Scamper data is selected from each AS for the destination addresses. If there is more than one source in the same AS, the first source discovered is used.

As explained in Section III (Results), some simulations were performed from a few sources to many destinations. In this case, the selection process is the same but with the source and destination roles reversed.

This methodology produces a topology with 4.2 million nodes (interfaces including sources and destinations) and 55 million links between nodes.

I. Probing Strategy

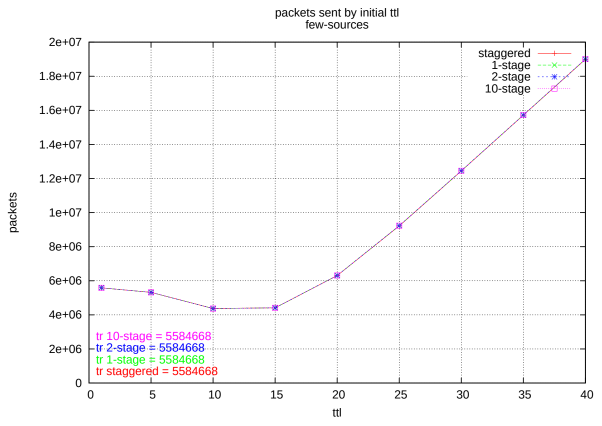
Several different strategies for when to start sending probes to a particular destination are modelled. The simplest is "1-stage" in which, all probing to all destinations starts at the beginning of the simulation. At the other extreme is "staggered" probing where, the next destination is not started until the previous one is complete and each monitor waits until the previous monitor is complete. Between there extremes we modelled "2-stage" and "10-stage" where the first two (or ten) destinations are probed sequentially (as in "staggered" probing) then the remaining probes are sent when they are complete.

The different probing strategies explore whether slower initial probing enhances the performance of Doubletree by allowing probes started later to benefit from more information from the early probes. In particular, the impact of probing on the last hops is particularly important because that is where there is the most potential for congestion from the traffic created by the large number of monitors. If the early probes learn the final hops, most later probes would not re-probe them. Staggered probing is not suitable for deployment because it will take too long to complete; it is simulated to provide a best case against which the performance of the other strategies can be compared.

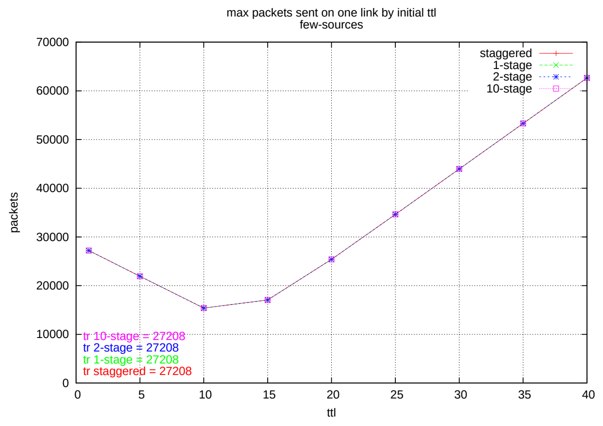
Unlike the traditional traceroute, Doubletree does not send multiple probes per TTL value. For comparison the traceroute model in the simulator also sends one probe packet per TTL value.

III. Results

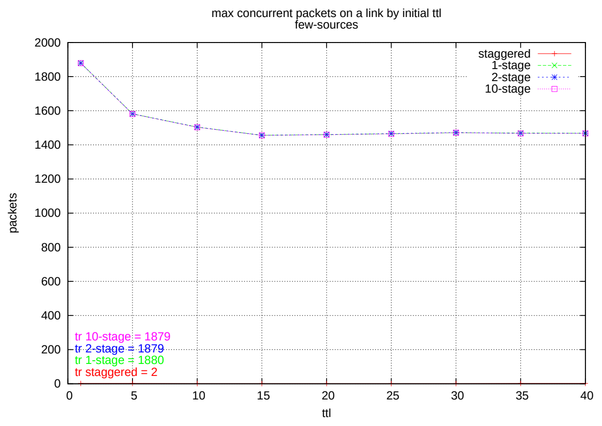
The following sections describe the results of simulation of Doubletree, first for the few sources case and then for many sources.



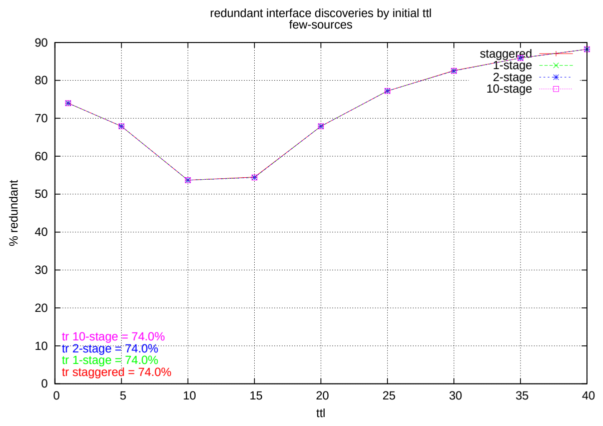
(a)



(b)



(c)



(d)

Figure 2. Few Sources, many destinations

A. *Few monitors to many destinations*

Although our motivation, the Atlas/Hubble application, leads us to be most interested in the performance of Doubletree with many monitors to a few destinations, we present the reverse arrangement so that our results can be compared with previous Doubletree simulations and because we explore more parameters than previous studies have.

Figure 2(a) shows the effect, on the total number of packets sent, of varying the starting TTL that Doubletree probes from. At low initial TTLs, the local stop set is less effective. At high starting TTLs, extra probes are sent for paths that are shorter than the starting TTL. The trade off between these factors, in this topology, suggests a starting TTL of between 10 and 15. There are no significant differences between the different probing strategies (staggered, 1-stage etc) and, as a consequence, the lines are indistinguishable on the graphs. The equivalent values for traceroute, which always starts probing from TTL 1, are shown in the bottom left of the graph.

The number of packets sent by Doubletree with an initial TTL of 15, is approximately 4.4 million compared with 5.5 million for traceroute. A similar trade off can be seen with the maximum number of packets sent on a single link and the highest number of concurrent packets on a link (see Figure 2(b) and (c)).

Even with Doubletree, there must be some redundant discoveries. These occur because Doubletree probing cannot stop in either direction until an interface that has already been seen is re-discovered or the source or destination is reached. When the initial TTL is too small for effective operation of the local stop set, or too big for the global stop set, further redundancies occur. Concurrent discoveries may also cause redundant probes. Figure 2(d) shows the percentage of interface discoveries that are redundant. Again, initial TTL around 10-15 is most effective with about half (55% for TTL=10) of the interface discoveries being redundant compared to 74% for traceroute.

B. *Many monitors to a few destinations*

The many sources to a few destinations case is shown in figures 3 and 4. In this case, the local stop set is lately ineffective and most optimisation comes from the global stop set. The cost of exchanging the global stop-set is not

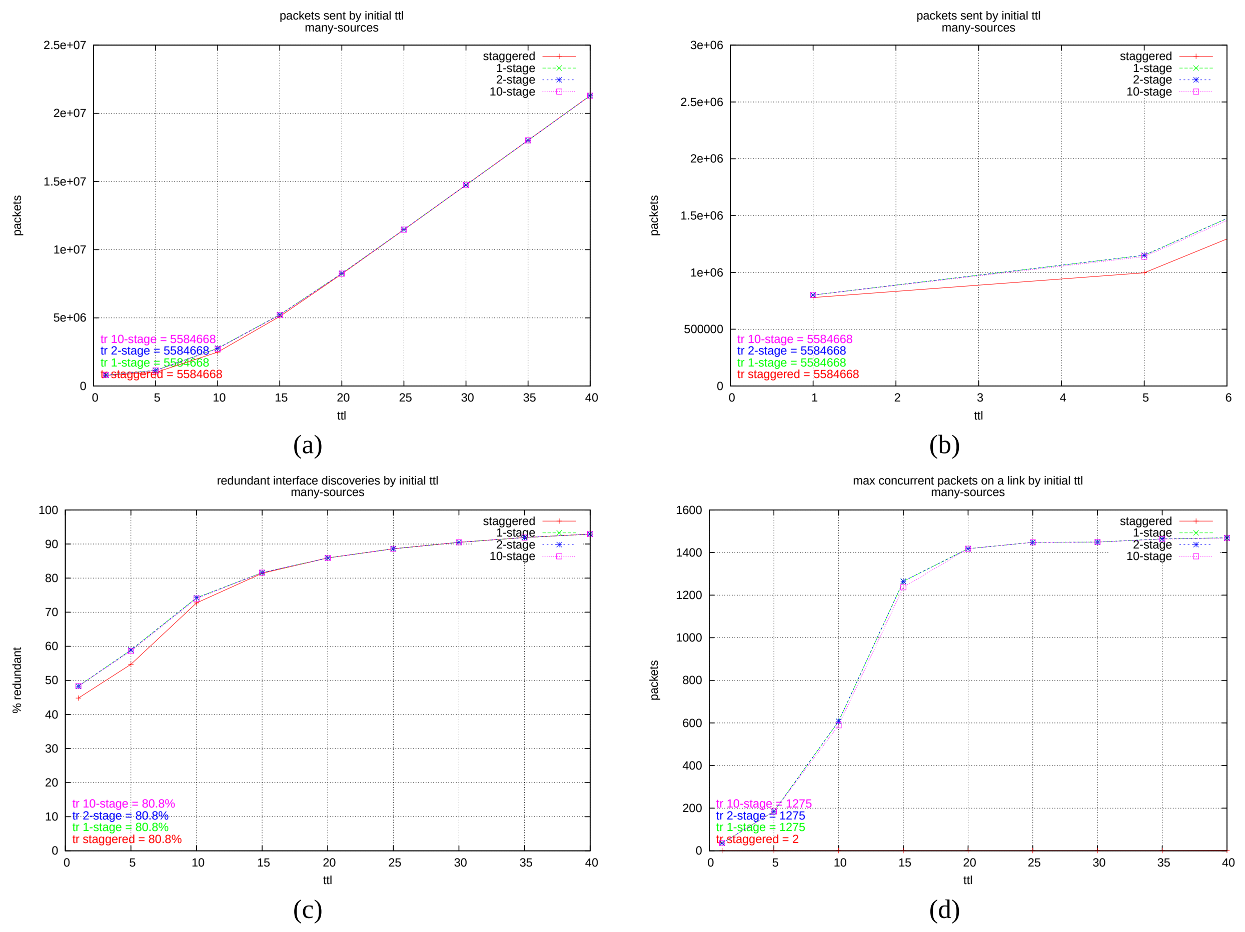


Figure 3. Many Sources, few destinations

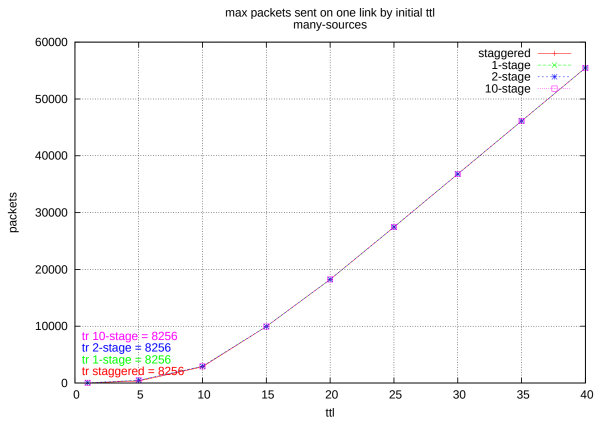
included in these results.

Figure 3(a) shows the effect of varying the initial TTL on the number of packets sent. Because of the predominant effect of the global stop set, which is most effective at smaller starting TTLs, there is no trade off between the effectiveness of the local and global stop sets. The number of packets increases as the starting TTL increases. Figure 3(b) shows an enlarged view of the area near the origin. At an initial TTL of 1, the number of packets sent using Doubletree is approximately 770,000 compared with 5,600,000 for traceroute. Note, however, that the effect of communicating the global stop set will add additional packets to the Doubletree case.

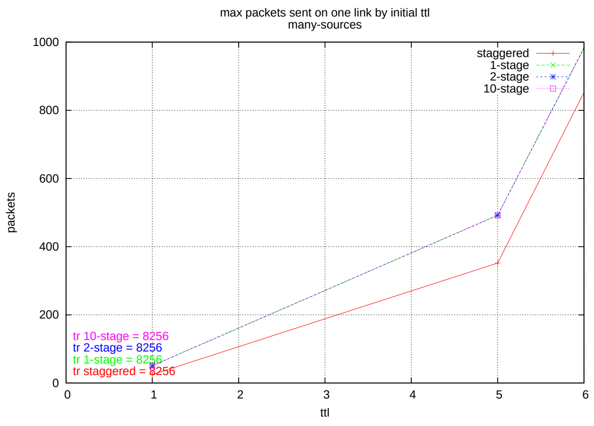
Figure 3(c) shows the percentage of redundant discoveries for the many sources case. Unlike the few sources case (Figure 2(d)), there is no trade off. At initial TTL of 1, there are less than 50% redundant discoveries (45% for staggered start times and 48% for the other probe starting strategies). Traceroute has 81% redundant discoveries for the many sources case.

The number of packets sent, especially the number of packets sent on the links close to the sources and destinations, is of particular interest in the Atlas/Hubble scenario that motivated this study. The total number of packets concurrently sent on a link is shown in Figure 3(d). For an initial TTL of 1, the largest number of concurrent packets on a link is 2 for staggered starts and 36 for the other probing strategies. The former is inherent in the staggered strategy, which does not send a new probe until the previous one has finished. The equivalent values for traceroute are 2 and 1275 respectively.

The maximum number of packets carried by any one (unidirectional) link during the course of the simulation is shown in Figure 4(a). An enlarged view of the origin in shown in Figure 4(b). For an initial TTL of 1, there are only 25 packets sent on the most most loaded link using staggered traces and 51 for the other cases. Classical traceroute (but with one probe per TTL, not 3) has a maximum load on a single link of approximately 8300. This is a very encouraging result but, again, we note the omission of the cost of exchanging the global stop set.



(a)



(b)

Figure 4. Maximum packets sent on a Link (many Sources, few destinations)

IV. Reproducibility

Reproducing the results described here requires the simulator code, the scamper and RIS data sets and some associated tools. The simulator code can be obtained from [8]. The simulator is written in C with some preprocessing of the topology files written in Perl. gcc 4.3.2 and Perl 5.10.0 were used to produce the results for this paper.

The scamper data set is not allowed to be redistributed but can be obtained from CAIDA via the request form at [9]. As previously noted, the data set used for this study is the Scamper run from 3 Jan 2009. The data set consists of 13 files with names in the pattern `daily.l7.t1.c000359.20090103.`mmm-cc`.warts.gz`. mmm-cc should be replaced with the monitor identification codes: `amw-us bcn-es cjj-kr dub-ie hel-fi hlz-nz laf-us lej-de mnl-ph nrt-jp san-us syd-au yto-ca`.

The files are decoded using the `sc_analysis_dump` tool from the scamper package. For this work version `scamper-cvs-20070523o` was used. Scamper [4] can be obtained from the WAND website [10]. The version used for this work is preserved along with the simulator code at the address given above.

For the IP address to ASN translation, RIS data from 30 March 2009 was used. RIS data may be fetched from the RIPE NCC at [11]. The RIS data that was used for this work is also preserved with the simulator code at the address given above.

V. Conclusions

If the cost of exchanging the global stop set is excluded, Doubletree is expected to be very effective at reducing the load on the monitors in an Atlas/Hubble system. Simulations suggest the load on the most heavily utilised link reduces from 8300 packets to just 51 packets at $TTL = 1$.

These simulations do not model the effect of sharing the global stop set. Sharing the global stop-set will increase the total number of packets sent (perhaps very significantly). However, this extra load will be most concentrated around the nodes that coordinate the exchange of global stop set data. It is possible that, an Atlas/Hubble system could be built with more capacity in this/these area(s). It is not yet clear to what extent communication of the global stop set reduces the Doubletree gains at interfaces near the destination probe.

A second outcome of this work is the demonstration that Internet scale simulation is now possible on commonly available hardware. In this study, available memory and single core execution speed where the limiting factors. The simulator was custom written for this application. Significant effort was made to optimise the implementation in both memory usage and execution time. A simulation run (representing a single point on the graphs) used around 8GB of RAM and took between one hour and three days depending on the number of packets sent. As noted, Scamper does not measure all of the Internet. However, commodity hardware with more cores and much larger memory are now common and within the budget of most academic computer science departments.

Acknowledgements

This work was undertaken while the author was on sabbatical with the RIPE NCC. My thanks for their support both practical and academic during this time. I am grateful to the RIPE NCC and CAIDA for making the data used in this work available. Support for the CAIDA IPv4 Routed /24 Topology Dataset is provided by the National Science Foundation, the US Department of Homeland Security, the WIDE Project, Cisco Systems, and CAIDA Members.

REFERENCES

[1] B. Donnet, B. Huffaker, T. Friedman, and K. Claffy, *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4268, ch. Evaluation of a Large-Scale Topology Discovery Algorithm, pp. 193–204. [Online]. Available: http://dx.doi.org/10.1007/11908852_17

[2] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the internet with hubble," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 247–262. [Online]. Available: http://portal.acm.org/citation.cfm?id=1387589.1387607

[3] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: From art to science," *Conference For Homeland Security, Cybersecurity Applications & Technology*, pp. 205–211, 2009.

[4] M. Luckie, "Scamper: a scalable and extensible packet probet for active measurement of the internet," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 293–245.

[5] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 41–56, October 2006. [Online]. Available: http://doi.acm.org/10.1145/1163593.1163602

[6] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 153–158. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177100

[7] RIPE NCC. (2010) RIS:Routing Information Service. [Online]. Available: http://www.ripe.net/projects/ris/

[8] T. McGregor. Dobuletree witn many sources: reproducability data. *Last Accessed 10 Jan 2011*. [Online]. Available: http://byerley.cs.waikato.ac.nz/~tonym/papers/icimp11/reproduction/

[9] CAIDA: Coperative association for Interent Data Analysis. Topology data request form. *Last Accessed 10 Jan 2011*. [Online]. Available: http://www.caida.org/data/active/topology_request.xml

[10] Waiakto WAND Network Research Group. Web site. *Last Accessed 10 Jan 2011*. [Online]. Available: http://wand.net.nz/

[11] R. N. C. Centre. Routing information service website. *Last Accessed 10 Jan 2011*. [Online]. Available: http://ripe.net/ris/

Impacts of Improved Peer Selection on Internet Traffic in BitTorrent Networks

Peter Danielis, Jan Skodzik, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
Tel./Fax: +49 (381) 498-7272 / -1187251
Email: {peter.danielis;jan.skodzik;dirk.timmermann}@uni-rostock.de
Web: http://www.imd.uni-rostock.de/networking

Thomas Bahls, Daniel Duchow
Nokia Siemens Networks GmbH & Co. KG
Broadband Access Division
17489 Greifswald, Germany
Tel./Fax: +49 (89) 5159-22771 / -18237
Email: {thomas.bahls;daniel.duchow}@nsn.com

***Abstract*—Peer-to-Peer (P2P) file sharing generates by far the most Internet traffic reaching up to 70 % in some regions of the world. These data volumes pose a significant challenge to Internet Service Providers (ISPs) regarding traffic engineering. Because P2P routing is usually agnostic of the underlying topology, traffic engineering abilities of ISPs are inhibited and their core networks are overburdened with P2P data. To disburden ISPs' core networks, we propose a new algorithm for the BitTorrent (BT) protocol in order to improve peer selection. BT users are provided with accurate information on the hop counts to other BT users to select physically proximate users. Thereby, the initial Time-To-Live value (TTL) of outgoing IP packets is copied and inserted as part of the BT payload. At the packet's destination, the hop count is calculated as the difference between the copied TTL and the TTL of the IP header. We present simulation results for standard and modified BT implementation and discuss impacts on both the load of ISPs' core networks and BT users' download performance.**

***Keywords*-BitTorrent, Peer-to-Peer, Topology Awareness**

I. Introduction

Today, Internet traffic is dominated by Peer-to-Peer (P2P) data ranging from 43 % up to 70 % in different regions of the world. This is mainly caused by file sharing applications such as eMule or BitTorrent (BT). Particularly, BT traffic accounts for up to 80 % of P2P traffic, i.e., 56 % of overall Internet traffic [1].

On the one hand, Internet Service Providers (ISPs) benefit from the P2P hype through an increase of their operating income. This is because P2P applications are one of the main reasons for Internet users to subscribe for a broadband connection [2]. But on the other hand, the high P2P data volumes pose a significant traffic engineering challenge. This discrepancy between operating income and traffic engineering challenge puts network operators and ISPs into a difficult situation. Other traffic like HTTP is choked down because the network infrastructure is overburdened with P2P data. The main reason is that routing within logical P2P networks does not take the underlying physical Internet topology into account [3]. Usually, an unstructured P2P network overlay—on which we focus here—is constructed by choosing random peers [4]. Due to this arbitrary procedure, neighborship on the P2P overlay does *not* implicate proximity on the underlying Internet topology at all. This problem is usually denoted as topology mismatching problem between P2P overlays and physical network infrastructures [5]. Thus, two communicating P2P neighbors may be physically far away from each other although the desired content is often available on a physically more proximate peer as well [6]. Communication with physically distant peers uses long data paths, e.g., regarding the hop count. This consumes more bandwidth, which is highly inefficient when the load of the network is already high. It can therefore cause traffic congestions [7]. In contrast, communication with proximate peers consumes less bandwidth and traffic in the core network diminishes. ISPs benefit from traffic reduction in their core networks as more bandwidth is available for other applications. Moreover, traffic congestions can be reduced as well.

Consequently, our contribution was motivated by the idea of disburdening ISPs' core networks by reducing the physical path lengths, i.e., the hop count. We developed a new algorithm for the BT protocol to use hop count as additional selection criterion for peers (besides their download performance). However, the hop count for determining proximity is not part of packets. Therefore, we also propose a new approach to provide P2P users—*BT users in particular*—with information on physical hop counts to other BT users. We modified the standard BT implementation such that the initial Time-To-Live value (TTL) of outgoing IP packets is inserted as part of the BT payload. At the packet's destination, the modified BT algorithm calculates the hop count from the inserted initial TTL and the received IP packet's TTL.

However, a BT user primarily wants to download desired content as fast as possible. He is usually not aware of or even not interested in the underlying transport mechanism. Thus, a user would not select the most proximate BT user among all users, which provide the desired content with nearly the same upload capability. However, we assume BT users to be cooperative by selecting close-by users unless they do not suffer from this decision regarding their performance. We define performance as the time required to get desired content, which we call the users' Quality of Experience (QoE). In the best case, it is beneficial for users and ISPs. In the worst case, we want the performance to be equal to or not considerably lower than the standard case when not using the hop count.

Briefly summarized, the main contributions of this paper are the following:

- Investigations are carried out on how to calculate the hop count and provide it for BT users.
- Improved peer selection for BT is described, where hop count information is used by a modified BT algorithm to select close-by users.
- Simulation results for standard and modified BT algorithm are presented and (dis-)advantages for both ISPs and BT users are discussed.

The remainder of this paper is organized as follows: Section II contains a comparison of the proposed approach with related work. Section III explains the computation of the hop count from the TTL and how to provide the initial TTL. Section IV addresses improved peer selection for BT. Section V shows impacts of improved peer selection on the load of an ISP's core network and BT users' performance. The paper concludes in Section VI.

II. Comparison with Related Work

Many approaches, e.g., [8] and [9] to *construct* unstructured topology-aware overlay networks do exist. These approaches improve performance significantly and avoid unnecessary traffic by exploiting network proximity. However, they require adding structure to unstructured P2P networks following physical network characteristics. Furthermore, traffic overhead is created for maintaining this structure. In contrast to these approaches, we do not intervene with the *construction* of unstructured P2P networks. Instead, we slightly modify the BT protocol to provide the hop count and select proximate peers by means of a new BT choking algorithm. Thereby, no modification of the construction algorithm is necessary and no additional packets have to be sent to determine the distance, i.e., the hop count between peers.

Also, there are approaches to *shape* P2P traffic in a more efficient way *with* network support. The IETF has planned to develop a protocol for Application-Layer Traffic Optimization (ALTO). By means of an ALTO server, peers can obtain information "to perform better-than-random initial peer selection" [10]. The Portal for P2P Applications (P4P) project aims at allowing more effective cooperate traffic control between P2P applications and ISPs via dedicated trackers to localize P2P traffic [11]. Moreover, in the course of the Network-Aware P2P-TV Application over Wise Networks (NAPA-WINE) project, the impacts on the underlying transport network shall be minimized when distributing P2P-IPTV datastreams [12]. Thereby, a so-called "network-peer can perform actions to optimize the P2P overlay taking into account the status of the transport network". [13] suggests to use autonomous system (AS) hop count to achieve locality-awareness in BitTorrent-like P2P applications. However, the tracker is required to know the Internet topology and peers have to obtain dynamic distance information by P4P or content distribution networks. As opposed to these approaches, our approach is completely self-sufficient and does not require network support but does the necessary modifications solely in the BT application.

III. Computing Hop Count for BT

Since hop count information is neither stored in the IP header nor elsewhere, it is necessary to compute it. There are two methods for hop count calculation [14]. One is the so-called active measurement. The other is denoted as passive measurement. For active measurement, ICMP ECHO packets are used. Although this method mostly results in an accurate hop count, applying it to many hosts in current and prospective P2P scenarios is impractical since enormous traffic overhead is created. Contrary, passive measurement simply means subtracting the final TTL of a received IP packet from its initial TTL. This is ideal for computing hop counts of many hosts as no extra packets have to be sent. Consequently, this approach is chosen for calculating the hop count. However, the problem with passive measurement is that the initial TTL is not available in IP packets. Thus, it must be made available.

The TTL is part of the IPv4 (further referred to as IP) header [15]. It is used as hop counter. Thus, each router processing a packet decrements the TTL by one. To calculate the hop count from TTL, the initial TTL of an outgoing IP packet is required. Then, this value can be subtracted from the final TTL of the IP header at the packet's destination to get the hop count. As shown in [16], due to the heterogeneity of the Internet, there is *no* unique initial TTL. The initial TTL depends on the operating system.

Consequently, the question is: How to provide the initial TTL? Therefore, we propose a modified BT algorithm to insert the initial TTL into BT messages.

A. Standard (tracker-based) BT algorithm

For every shared file in BT, an own P2P net is created. To search for a file, usually a web site is contacted to get a *.torrent* file. This file contains, among other things, the address of a *tracker* and information about the file to be downloaded. The tracker is contacted to get a list of BT users holding the file (or parts of it—so-called *chunks*). Thereby, all BT users, which are interested in this file, form a so-called *swarm*. Complete downloader serving the whole files are called *seeds*. Incomplete downloaders are called *leechers*.

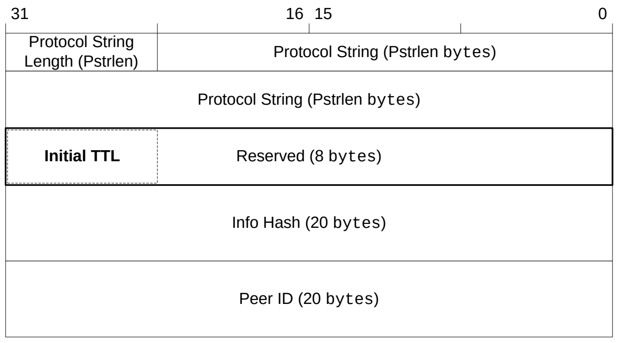

31
16 15
0
Protocol String Length (Pstrlen)
Protocol String (Pstrlen bytes)
Protocol String (Pstrlen bytes)
Initial TTL
Reserved (8 bytes)
Info Hash (20 bytes)
Peer ID (20 bytes)

Figure 1. Composition of a BT Handshake message.

For selecting a user who may download a chunk, BT applies the so-called *choking algorithm* [17]. Put in a nutshell, this is a variant of the tit-for-tat strategy. Only users offering sufficient upload performance are given download in return. The choking algorithm to determine a user that may download chunks is executed periodically because upload performance of users can change quickly.

B. Including the Initial TTL in BT messages

To introduce as few overhead as possible into the BT algorithm and BT traffic, the initial TTL is only included into necessary BT messages. Two types of messages can be distinguished in BT flows: the tracker requests and responses and the messages between BT users. Thereby, messages between BT users are solely exchanged via TCP sockets. One message necessary for BT user interaction is the *Handshake* message (see Figure 1) [18]. Per BT specification 1.0, which is widely used for BT protocol implementations, Pstrlen is set to 19 and Protocol String = "BitTorrent protocol".

A BT Handshake is sent by the initiator of a connection between two users of a swarm. In return, the recipient of the Handshake message has to respond with a Handshake message himself. In both the Handshake message and the response to it, the modified BT algorithm directly inserts the initial TTL. The authors suggest to use the first byte of the eight *Reserved* bytes since these bytes can be used to change the BT protocol behavior.

C. Providing the TTL in the BT application

As TCP sockets are used for communication, IP header fields like the TTL are not available in applications per se. Therefore, following two questions are answered to clarify how to make it available:

1) How to get the initial TTL of outgoing BT Handshakes? The initial TTL is retrieved via the BSD sockets compatible function *getsockopt*, which is both Windows and Unix-compatible. Thereby, the specific socket option *IP_TTL* is used, which is supported by TCP sockets for outgoing packets.

2) How to get the final TTL of incoming BT Handshakes? Since TCP sockets do not offer support for providing the TTL of incoming packets, the pcap (packet capture) interface is used *additionally*. Unix-like OS apply the pcap implementation *libpcap*, whereby Windows OS use a port of libpcap called *WinPcap*. Using filters like the Berkeley Packet Filter, already the kernel can be instructed to copy only packets, which match the composition of a BT Handshake, to the BT application. Thus, the kernel buffer is not overfilled with packets, which could lead to high packet loss. Possible alternatives are *raw sockets* offering direct access to the network layer as well. However, they do either forward each packet to the application or do not forward TCP packets at all (depending on the OS implementation) and are thus not feasible.

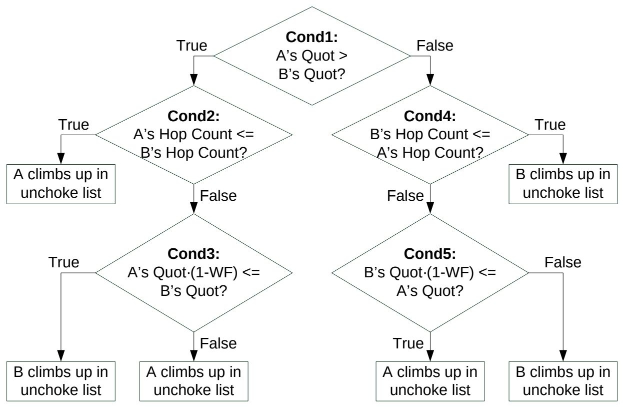

Cond1: A's Quot > B's Quot?
True
False
Cond2: A's Hop Count <= B's Hop Count?
Cond4: B's Hop Count <= A's Hop Count?
True
False
True
False
A climbs up in unchoke list
B climbs up in unchoke list
Cond3: A's Quot·(1-WF) <= B's Quot?
Cond5: B's Quot·(1-WF) <= A's Quot?
True
False
True
False
B climbs up in unchoke list
A climbs up in unchoke list
A climbs up in unchoke list
B climbs up in unchoke list

Figure 2. New BT choking algorithm for calculating a user's position in the unchoke list. The hop count is used as additional selection criterion.

IV. Improved Peer Selection for BitTorrent

The modification of BT for improved peer selection concerns BT's choking algorithm. The standard choking algorithm selects BT users that may download chunks solely depending on their offered upload performance (except *optimistic unchokes*). Users are ranked in an unchoke list such that the user with the highest upload performance (i.e., the highest *service rate*) is on top. Usually, a fixed number of users on top of the list (e.g., 4) may download concurrently. In the modified version, users are no longer ranked only depending on their upload performance. Instead, for two users A and B *in another user's unchoke list*, quotients for those users A and B are calculated as

$$Quotient = Service\ Rate/Hop\ Count.$$

The quotient (denoted as *Quot* in the algorithm depicted in Figure 2) determines a user's rank in the unchoke list. If A's quotient is greater than B's, i.e., condition 1 = true (denoted as *Cond1* in Figure 2) and A's hop count is smaller than or equal to B's (Cond2 = True), A climbs up in the unchoke list.

However, if A's hop count is greater than B's (Cond2 = False), A's quotient is multiplied (i.e., weighted) by a variable factor (*1 - hop count weighting factor* (WF)) with WF values ranging from 0 to 1. If A's weighted quotient is smaller than or equal to B's quotient (Cond3 = True), B climbs up in the unchoke list because B's hop count is weighted more than A's upload performance. Otherwise (Cond3 = False), A climbs up because A's upload performance is weighted more than B's hop count. In the else-branch of the algorithm, we have stated the analog conditions for B's quotient being greater than A's.

As an exemplification for the algorithm, let us assume a user A with high service rate and high hop count and a user B with moderate service rate but very low hop count. Following the calculation rule for a user's quotient, A is assigned a relatively low quotient compared to B's quotient regardless A's high service rate. Still, A's quotient be greater than B's quotient in this example (Cond1 = True) although B's hop count be significantly smaller than A's hop count (Cond2 = False). However, B's hop count can be given an even higher weight by assigning an approriate, i.e., high WF value to let B climb up in the unchoke list (Cond3 = True).

To summarize, WF is used to make a compromise regarding the weighting of a user's upload performance and his hop count. A high WF value results in BT users with low hop counts on top of the unchoke list almost regardless their upload performance. Vice versa, a low WF value leads to a higher weight of a user's upload performance. Thus, users with high upload performance are put more probably on top of the unchoke list nearly irrespective of their hop count.

One approach for the selection of close-by BT users is to let the user decide directly. The alternative approach, which we follow in this paper, is to integrate an automatic selection mechanism into the BT algorithm.

V. Evaluation of Standard and Modified BT Algorithm

In this section, simulation results are shown using the network simulator ns-2. Results include a comparison of standard with modified BT algorithm in terms of the number of hops between users, the load of the core network, and users' QoE.

A. Simulation Setup

To implement the BT algorithm in ns-2, a BT patch from [19] has been used. The BT algorithm has been complemented by the dynamic nature of peers, i.e., the BT users' behavior of continuously leaving and entering the BT network. According to [20], BT users follow a Weibull distribution when arriving at the BT network (inter-arrival time). Session lengths of BT users, i.e., the time how long they stay in the BT network each time they appear are implemented to follow a Weibull distribution as well. When BT users leave the BT network after a session, they return after a uniformly distributed time (downtime). Finally, BT

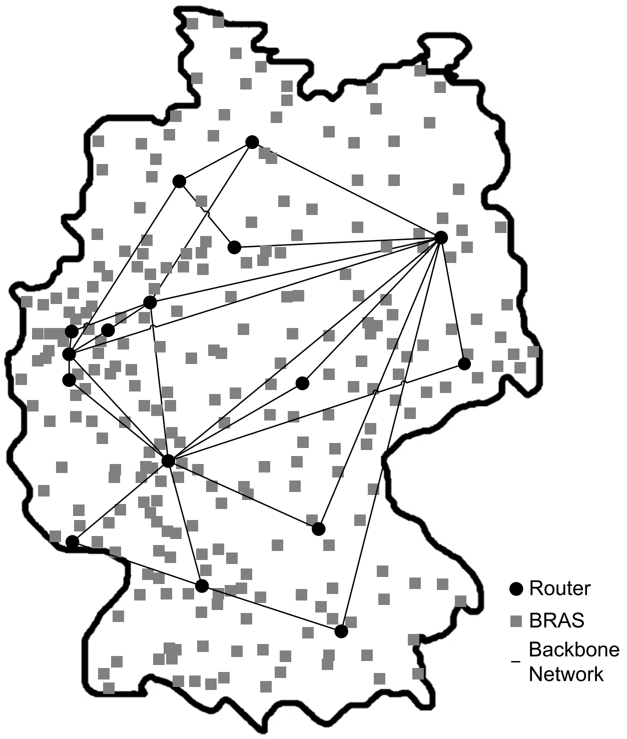


Figure 3. Telefonica's network infrastructure in Germany [21]

users stay in the BT network for a while after the completion of a download (lingering). The lingering time is modeled with a Weibull distribution.

At the start of each simulation run, one BT user is a seeder. This seeder stays in the network until each BT user of the swarm has finished his download of a file of 100 MB in size. The maximum number of users that may download concurrently from another user is set to 4.

For the simulation, a topology has been developed, which maps Telefonica's backbone network in Germany [21]. Telefonica possesses one of the most capacious network infrastructures in Germany. The schematic layout of the developed topology is depicted in Figure 3 and consists of

- a backbone network of routers,
- Broadband Remote Access Servers (BRAS) that are connected to routers, which are linked to DSL Access Multiplexers (DSLAMs),
- and BT users, which are connected to DSLAMs.

In accordance with Telefonica's network infrastructure in Germany, the developed topology comprises 16 routers. The number of BRAS is set to 4 per router (resulting in 64 BRAS) and there are 6 DSLAMs per BRAS (resulting in 384 DSLAMs). The number of BT users is set to 200 for the simulations. The routers form a static structure like the one apparent in Figure 3. BRAS are uniformly distributed around routers and in the same way, DSLAMs are uniformly distributed around BRAS. Users are randomly connected to

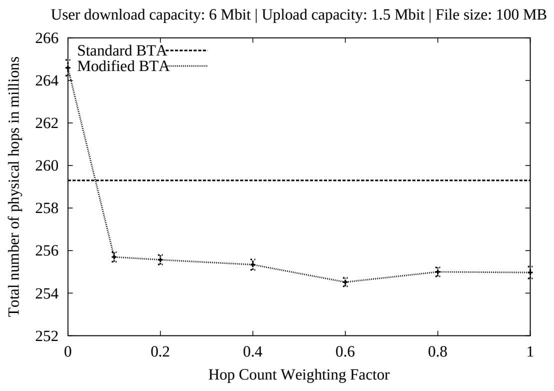


Figure 4. Number of hops for varying WF values. The result for standard BT is independent from WF values and therefore constant.

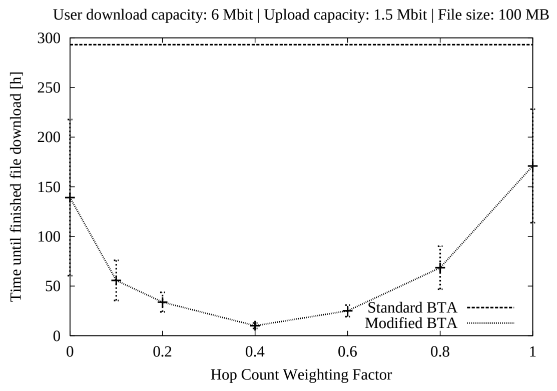


Figure 5. Time until the last BT user has downloaded the file for varying WF values. The result for standard BT is independent from WF values and therefore constant.

DSLAMs. The number of BRAS, DSLAMs, and users is fixed for all simulations. The bandwidth between routers and between routers and BRAS has been set to 20 Gbit/s. The bandwidth between DSLAMs and BRAS is set to 1 Gbit/s. These bandwidths values are common values in practice. Each BT user is assigned a download capacity of 6 Mbit and an upload capacity of 1.5 Mbit, which are reasonable values for an asymmetric Internet access.

B. Simulation Results

Both standard and modified BT algorithm (BTA) have been simulated on the developed topology. In our simulations, the following values have been determined for varying WF values to compare both algorithms:

- Number of physical hops: Summed up number of physical hops that data has to travel through the network during the simulation.
- Data volume in the core network: Summed up data volume passing the routers of the core network during the simulation.
- Time necessary for the last user to finish a file download: Time needed until the last BT user has finished the file download during the simulation.

As users are connected to DSLAMs randomly, for each WF value on the x-axis, 38 measurements have been taken. In the diagrams, the mean value of those 38 measurements is depicted on the y-axis for each WF value. For the modified BT, the 95 % confidence interval (CI) is depicted to demonstrate that the measurements' precision is sufficient to draw conclusions.

The calculated mean value for standard BT is independent from WF values and thus constant. Therefore, the CI is not charted for standard BT (CI for the data volume in the core network: 73 Mbit, CI for the number of physical hops: 234344, CI for the time necessary for the last user to finish a file download: 157 h).

As apparent from Figure 4, the number of hops decreases when applying the modified BTA except for WF = 0. For WF = 0, the simulation results show a minor increase of the number of hops by 2 % compared to the standard BTA. This is due to the fact that hop count is considered by the algorithm in Figure 2 but users' upload performance dominates as peer selection criterion. Thereby, the degrees of freedom are limited and the number of hops increases. Please remember, that hop count is *always* considered by the modified BTA and an increasing WF value solely *boosts* the hop count's influence. Any other WF value decreases the total number of hops. In fact, for WF = 40, 60, 80, and 100 %, we achieved the highest reduction of approximately 2 %.

This relatively slight decrease in the number of hops results in a significant lower load of the core network for the modified BT variant. Table I illustrates this fact, showing a reduction of the core load by 11 % for WF = 40 and 60. The slight increase of the core load for WF = 0 % has the same reasons that apply for the increased number of hops for WF = 0.

Furthermore, our simulations show that the time necessary until the last BT user has downloaded the complete file is considerably lower if the modified BTA is applied (see Figure 5). In fact, time is even decreased by up to 97 % for WF = 40 %. This tremendous decrease of time involves a reduction of the core load by 11 % and a decrease of the number of hops by 2 %. Thus, choosing WF values between 40 and 60 % is obviously benefical for both BT users and the core network as these values offer the highest degrees of freedom regarding peer selection.

Table I
DATA VOLUME IN THE CORE NETWORK FOR VARYING WF VALUES. THE RESULT FOR STANDARD BT IS INDEPENDENT FROM WF VALUES AND THEREFORE CONSTANT.

WF	Standard BTA	Modified BTA		
	Data volume [Gbit]	Data volume [Gbit]	Data volume reduction [%]	CI [Mbit]
0	29.10	29.48	-1	79
0.1		26.24	10	82
0.2		26.20	10	84
0.4		26.03	11	75
0.6		25.99	11	68
0.8		26.16	10	80
1.0		26.19	10	90

VI. CONCLUSION

This paper proposes a new choking algorithm for the BT protocol to preferably select physically close-by BT users in order to disburden ISPs' core networks. The selection criterion is the hop count. It is calculated from the difference of the initial TTL value of a packet's IP header and the TTL value at the packet's destination. As the initial TTL is not directly available, it is inserted into BT Handshake messages by the modified BT algorithm.

The simulations carried out for the BT algorithm clearly show that ISPs benefit from a modified BT using the hop count as additional selection criterion for download partners. The load of the ISP's core network is alleviated by up to 11 %. Thereby, traffic is localized (the number of hops is reduced by up to 2 %). Moreover, our simulation show that users' QoE tremendously increases as time for the last BT user to finish a download is decreased by up to 97 %.

Future work will focus on providing the hop count for further P2P file sharing protocols such as eMule's unstructured eDonkey2000 and its impacts on Internet traffic.

ACKNOWLEDGEMENT

The authors would like to thank the Broadband Access Division of Nokia Siemens Networks GmbH & Co. KG in Greifswald, Germany for their inspiration and continued support in this project. This work is partly granted by Nokia Siemens Networks.

REFERENCES

[1] H. Schulze, K. Mochalski (ipoque), "Internet Study 2008/2009," 2009.

[2] T. Mennecke, "DSL Broadband Providers Perform Balancing Act," 2005.

[3] V. Aggarwal, S. Bender, A. Feldmann, and A. Wichmann, "Methodology for Estimating Network Distances of Gnutella Neighbors." GI Jahrestagung (2), 2004, pp. 219–223.

[4] R. Steinmetz and K. Wehrle, *P2P Systems and Applications, Springer Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2005.

[5] H. Wan, N. Ishikawa, and J. Hjelm, "Autonomous Topology Optimization for Unstructured Peer-to-Peer Networks." ICPADS, 2005, pp. 488–494.

[6] A. Rasti, D. Stutzbach, and R. Rejaie, "On the Long-term Evolution of the Two-Tier Gnutella Overlay," no. 4146697. INFOCOM, 2006.

[7] X. Xiao and L. Ni, "Internet QoS: A Big Picture," vol. 13. IEEE Network Magazine, 1999, pp. 8–18.

[8] S. Merugu and E. Zegura, "Adding Structure to Unstructured Peer-to-Peer Networks: The Use of Small-World Graphs." JPDC, 2005, pp. 142–153.

[9] Y. Liu, X. Liu, L. Xiao, L.M.Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems." INFOCOM, 2004, pp. 2220–2230.

[10] IETF, "Application-Layer Traffic Optimization (alto)," 2009. [Online]. Available: http://datatracker.ietf.org/wg/alto/charter/

[11] H. Xie, Y. R. Yang, A. Krishnamurthy, and Y. L. A. Silberschatz, "P4P: Provider Portal for Applications." ACM SIGCOMM, 2008, pp. 351–362.

[12] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi, "Building a cooperative P2P-TV application over a wise network: The approach of the European FP-7 strep NAPA-WINE." IEEE Communications Magazine 46 (4), 2008, pp. 20+22.

[13] B. Liu, Y. Cao, Y. Cui, Y. Lu, and Y. Xue, "Locality Analysis of BitTorrent-Like Peer-to-Peer Systems." 7th IEEE CCNC, 2010, pp. 1–5.

[14] K. Fujii and S. Goto, "Correlation between Hop Count and Packet Transfer Time." APAN/IWS, 2000.

[15] Information Sciences Institute, University of Southern California, "Internet Protocol Specification," RFC 791, September 1981.

[16] Swiss Education & Research Network (SWITCH), "Default TTL Values in TCP/IP," 2002.

[17] B. Cohen, "Incentives Build Robustness in BitTorrent." First Workshop on the Economics of Peer-to-Peer Systems, June 2003.

[18] " Bittorrent Protocol Specification v1.0," 2009. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[19] K. Eger, T. Hofeld, A. Binzenhofer, and G. Kunzmann, "Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations." UPGRADE-CN'07, 2007, pp. 9–16.

[20] D. Stutzbach and R. Rejaie, "Understanding CHurn in Peer-to-Peer Networks." ACM SIGCOMM Internet Measurement Conference, 2006, pp. 189–202.

[21] Telefonica, "Unser Netz," 2009. [Online]. Available: http://www.telefonica.de/wholesale/unser-netz.html

A Configurable FPGA-Based Traffic Generator for High-Performance Tests of Packet Processing Systems

Andreas Tockhorn, Peter Danielis, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
Tel./Fax: +49 (381) 498-7269 / -1187251
Email: {andreas.tockhorn;peter.danielis;dirk.timmermann}@uni-rostock.de

***Abstract*—For the evaluation of high-speed packet processing systems, high performance traffic generators are needed. They have to be configurable to produce various traffic patterns and achieve preferably full Gigabit link utilization. The evaluation of packet processing systems has become a critical task as current open-source—especially software—tools have a lack of throughput and suitable hardware generators are very expensive. Thus, an FPGA-based testing framework containing a traffic generator and a performance monitor, each of them based on an FPGA, was developed. In order to evaluate different packet processing systems, this framework maintains a high level of configurability. Summarized, this paper presents an FPGA-based traffic generator, dedicated to high throughput testing of packet processing systems.**

***Keywords*-Traffic Generator, Performance Measurement, Network Testing**

I. Introduction

Today, an increasing number of users subscribe for an Internet connection. Due to their increasing bandwidth demands, Packet Processing Systems (PPS), which execute tasks like packet classification, manipulation, and forwarding have to process packets at very high rates. To guarantee their correct functionality, they have to be tested under worst-case conditions i.e., with maximum traffic load. Consequently, traffic generators are needed for generating traffic to fully utilize links in order to evaluate a PPS's performance under these conditions. At the same time, traffic generators have to to be configurable to generate various patterns of traffic occurring in practice. Software-based traffic generators provide high configurability but have limited performance. Traffic generators implemented in hardware e.g., on a Field Programmable Gate Array (FPGA) are able to generate traffic at very high rates. However, they often lack a high degree of configurability as providing high configurability in hardware results in enourmous hardware costs or are too expensive for the proposed use.

Thus, this work has been inspired by the need of an affordable and configurable traffic generator offering full Gigabit Ethernet link utilization. The developed framework consists of an FPGA-based traffic generator, an FPGA-based traffic monitor to measure Gigabit link utilization, and a software tool to configure the traffic generator. An unique selling proposition of the proposed architecture is its integration in the hardware design process of PPS by reusing the therefore developed testbenches.

The traffic generator's performance is compared to that of the open-source software based traffic generator packETH [1]. PackETH provides convenient handling to carry out extensive tests and, to the best of our knowledge, its performance has not been evaluated before. We exemplarily derive the necessary level of configurability from tests to evaluate the performance of a recently published PPS called IPclip (IP Calling Line Identification Presentation) [2].

Briefly summarized, the main contributions of this paper are the following:

- Investigations on requirements for a configurable traffic generator are carried out.
- A combined hardware-software framework consisting of a configuration tool, traffic generator, and traffic monitor is proposed.
- The traffic generator's performance results for the IPclip use case are presented. They are compared to the performance results achieved for the traffic generator packETH.
- Integration of the framework into the hardware design process of PPS is illustrated.

The remainder of this paper is organized as follows: Section II contains an overview of related work. Section III investigates requirements for a configurable traffic generator. Section IV introduces a combined hardware-software prototype for the developed traffic generator. Section V presents performance results for both the proposed packet generator and packETH. The paper concludes in Section VI.

II. Related Work

There is a wide variety of tools to generate traffic—both commercial and open-source solutions.

One example for a commercial software solution is a highly configurable traffic generator offered by ZTI for Windows XP and Vista [3]. It achieves a link utilization of up to 97.4 % for 1 Gbit/s Ethernet links. However, for testing PPS as in the authors' targeted use case, ZTI's traffic generator is both too complex, does not offer full Gigabit Ethernet link utilization, and is not complimentary.

Most of available open-source software traffic generators are designed for Linux, either as Linux kernel modules or user space tools. Compared to kernel modules, user space tools have a limited performance as kernel modules directly operate on the network device driver bypassing the kernel networking subsystem. The Kernel-based Traffic Engine (KUTE) is an example for a Linux kernel module [4]. KUTE is able to send UDP packets and approximately achieves up to 41.4 % link utilization for Gigabit Ethernet links in the authors' investigated test cases. Examples for user space tools are the Brawny and RobUst Traffic Engine (BRUTE), the Real-time UDP Data Emitter (RUDE), the Multi-Generator (MGEN), and the Internet Traffic Generator (ITG) [5][6][7][8]. None of them achieves higher link utilization rates than KUTE does.

Hardware traffic generators mostly base on Field Programmable Gate Arrays due to their flexibility and high performance [9][10]. In [9], the authors state their architecture is scalable from 50 Mbit/s up to 2.5 Gbit/s but do not carry out performance evaluations. The work proposed in [10] aims at evaluating high performance QoS traffic servers rather than achieving full link utilization. There are some approaches combining general-purpose PCs with network processors such as BRUte on Network prOcessor (BRUNO) [11]. However, the authors do not carry out performance evaluations either.

In contrast to the authors' approach, none of the open-source packet generators' performance is sufficient to test PPS with full link utilization. Furthermore, most FPGA-based traffic generators are designed for dedicated purposes or, in the authors' opinion, lack a detailed evaluation of performance for extensive test cases.

Another group of traffic generators are commercial Hardware based generators e.g., from IXIA [12]. Depending on the traffic patterns to be sent, these generators are suitable for testing PPSs with high loads. Most of them can easily be extended for higher generation loads, by adding extension cards with further generation engines. However, these systems are very expensive. Depending on the desired configuration, thousands of dollars up to more than 100,000 dollars have to be spent. Therefore, these well-performing traffic generators are not appropriate for the proposed scenario.

III. Requirements for a configurable Traffic Generator

On the one hand, traffic generators for testing PPS have to provide a high level of configurability to be able to configure extensive test cases. To exactly reproduce practical traffic conditions, it is necessary to be able to adjust all necessary parameters. Preferably, every single frame should be definable. Thereby, a frame's adjustable parameters depend on the PPS' use case. On the other hand, traffic generators should achieve high traffic rates anyway. Especially if the performance of a PPS shall be evaluated, achievable throughput of the traffic generator is of high importance. Since an optimal combination of these two requirements can only be satisfied by expensive hardware based traffic generators, an FPGA-based traffic generator was developed.

Before the specific requirements for the presented traffic generator will be worked out, its use case shall be briefly explained. The PPS to be tested, is the recently invented IPclip mechanism [2][13]. IPclip is implemented on the access nodes of an internet service provider. It is a mechanism providing Trust-by-Wire in IP-based networks by adding trustworthy location information to IPv4 and IPv6 packets. Thereby, IPclip adds some additional header options to the IP header. These options include location information e.g., a GPS position and some information about the access node. Since the IPclip prototype has already been functionally verified [14], further investigations concerning its performance and the stability of its implementation under very high traffic loads shall be made. Since it inserts additional information to the IP headers and therefore increases packet lengths, it can be necessary to drop a certain percentage of frames in case of full link load. To reproduce these real scenarios, it is mandatory to use test equipment capable of generating traffic fully utilize a 1 Gbit/s Ethernet link. As exposed in the preceding section, only hardware-based traffic generators are able to provide appropriate traffic patterns in conjunction with high configurability.

Link utilization denotes the quotient of achieved throughput (TP) divided by the theoretically possible TP for a link as stated in Formula 1:

$$Link\ Utilization = \frac{Achieved\ TP}{Theoretically\ Max.\ TP} \tag{1}$$

Thereby, TP is the number of bytes traversing a link per second (see Formula 2).

$$TP = \frac{Number\ of\ Bytes}{1s} \tag{2}$$

When generating frames with a traffic generator, the frame size in bytes is an important parameter, since it directly determines the maximum number of frames i.e., headers, which have to be generated. The total number of bytes on a 1 Gbit/s link is calculated as the sum of following values:

- 12 bytes Inter Frame Gap (IFG)
- 7 bytes preamble and 1 byte Start-of-Frame-Delimiter (SFD)
- frame size in bytes
- 4 bytes Frame Check Sequence (FCS)

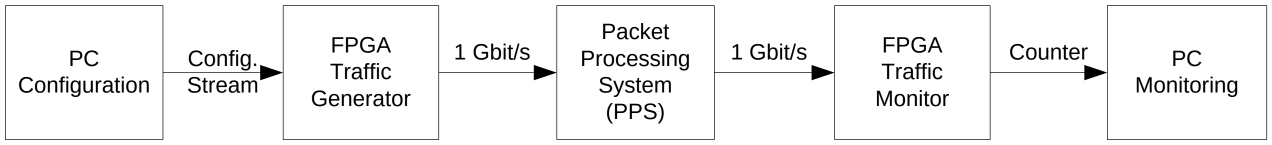


Figure 1. Test setup consisting of PC running the configuration PC, hardware traffic generator, PPS under test, and monitoring HW and PC.

The IFG, preamble, SFD, and FCS are added by transceiver chips and cannot be influenced by the traffic generator. Thus, to calculate the total number of bytes per link, 24 bytes have to be added to the size of each sent frame.

The smallest size an Ethernet frame can have is 60 bytes (without IFG, preamble, SFD, and FCS). That is, 84 bytes are occupied on a 1 Gbit/s Ethernet link for each frame. As we focus on PPS, which take their decisions solely based on header information of frames, processing minimum length frames represents the worst-case. On a 1 Gbit/s Ethernet link, at most approximately 1.5 million smallest size frames can be transmitted per second. That is, the corresponding maximum number of headers to be processed by the PPS. Contrary, only the headers of 81,000 frames have to be processed when sending maximum length frames of 1,514 bytes. Consequently, achieving full link utilization for smallest size frames is a critical task for traffic generators.

To enable the proposed traffic generator to fully utilize a Gigabit link for minimum sized frames, some limitations concerning the configurability of header options had to be made. Considering the IPclip use case, header options have been defined to be configurable based on their relevance for the IPclip mechanism. These fields include IP source address, two VLAN tags, and the length of the to be generated frame.

Summarized, the proposed architecture for a configurable traffic generator shall be able to fully utilize a 1 Gbit/s Ethernet link and be configurable in regard to the IPclip use case. A further major requirement was to be able to reuse the hardware testbench of the developement process of the IPclip system.

IV. Prototype Realization

To fullfill the above mentioned requirements, an architecture divided into a HW and a SW part was developed.

The software part enables convenient and flexible configuration and only generates values for header options, which shall be variable. The hardware, implemented on an FPGA ensuring throughput of 1 Gbit/s, is configured by this software running on a PC. The main task of the hardware is to generate standard conform frames i.e., IP packets based on these variable header options send from the configuration software. To clarify, the hardware implements a kind of traffic amplification. From a small configuration set, it generates a larger complete frame.

Figure 1 depicts the general setup for the usage of the proposed system architecture and a monitor to measure the output of the PPS to be tested. In the first phase, a software on a PC generates sets of values for the configurable header options. Following, these sets are transmitted via an Ethernet link to the FPGA, which implements the HW part of the traffic generator. The therefore sent frames (see Figure 2) consist of one byte, which determines the operation for the HW. This byte is followed by as many configuration sets as a maximum length frame can contain. The HW implemented on this FPGA is the main functional part of the generator. It receives these configuration sets. From each set it generates one Ethernet frame. These generated frames are mostly fixed in their composition. Only the options, which have to be configurable in order to test the PPS, are generated and sent from the PC. The resulting frames are transmitted on the link to the PPS. After the previously mentioned steps, the task of traffic generation is finished. The PPS under test receives the generated frames. It processes them and transmits the traffic to another FPGA. This one implements a traffic monitor able to count the number of received frames and bytes and collect other information about the incoming data stream. These values are transmitted via a second Ethernet link to a PC. On this PC, a software determines performance metrics of the PPS e.g., regarding frame rate, bandwidth, and packet drops. Only a HW-based monitor was able to correctly measure the traffic under high loads. This especially applies for traffic streams consisting of many minimum length frames.

During the evaluation of the traffic generator itself, nearly the same setup has been used. It only differs in the HW being directly connected with the monitor. Thereby, the parameters of the generated traffic could be measured conveniently. The implemented traffic monitor is not capable of evaluating fine grained statistical information about the traffic. It does not bother with timing data e.g., jitter etc., since this is out of

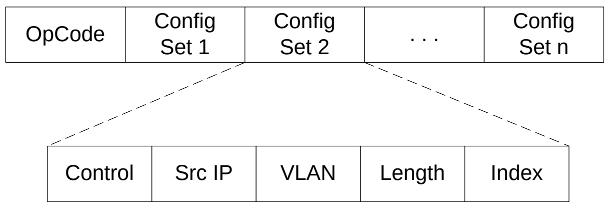


Figure 2. Structure of a configuration frame containing an operation byte followed by as many configuration sets as possible.

the scope for this work and the envisaged use case of IPclip.

The fundamental idea of this architecture is an amplification of the configuration stream to a data stream of 1 Gbit/s. In order to reduce the amount of data to be generated by and sent from the PC, the configurable header options are as small as possible. An example traffic stream shall be generated, in which frames are 500 bytes long on average. Each frame has a specific frame length (2 byte), source IP address (4 bytes), and include one VLAN tag (2 byte). One control byte is needed to define, which options of the header to be configured with this set. Therefore, a configuration set for one single frame consists of 9 bytes. Resulting from this scenario, amplification of link utilization by a factor of 55 is realized. This example only clarifies the fundamental mechanism. It does not take into account values like inter frame gap, preamble, and start of frame and FCS.

A. Hardware

This section illustrates the hardware architecture of the traffic generator (see Figure 3). Fundamentally it consists of two finite state machines (FSM). At the input of the traffic generator, an FSM receives frames from the configuration software running on a PC. These frames contain commands from the software part and configuration sets, respectively. All configuration sets are stored into a first in first out memory (FIFO). Since the link to the PPS shall only be fully utilized and no deterministic traffic patterns shall be sent, it does not matter if this memory is full resulting in dropped configuration sets. However, it is more important that the FIFO never runs empty to maintain highest load on the PPS under test. Consequently, the generation of frames does not start until the FIFO is filled for the first time.

The second FSM depicted in Figure 3 generates frames. Therefore, it reads a set from the FIFO. First of all, it processes the control byte determining, which header options to be configured. Based on this information, a frame of the desired length and the configured header options is generated. Consequently, all header options as well as the payload of the frame, which are not configured by the configuration set, are determined statically or randomly. Whether these fields shall be filled with random or static data is determined by the configuration software. Since, the proposed traffic generator shall be able to build frames of many different protocols, there exists corresponding state machines to implement the needed header structures.

Although the proposed architecture is able to realize an enourmous amplification in link utilization, the FIFO containing the configuration sets can run empty. Therefore, two solutions were implemented. The first one is to repeat the last generated frame. To configure this solution, the configuration PC sends a frame before the actual test run starts, which determines how often generated frames shall be repeated. For the envisaged IPclip use case the same frame can be repeated immediately. However, when testing PPSs,

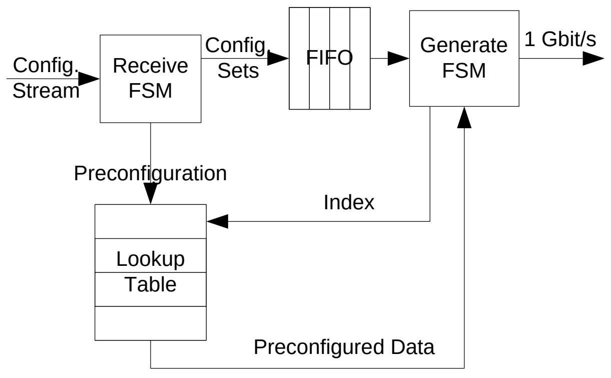


Figure 3. Architecture of the configurable traffic generator's HW.

which implement some caching functionalities, repetition of frames is no appropriate action. This would probably falsify the performance evaluation of those systems. Therefore, a second solution was implemented. Before the generation of frames starts, a configuration stage is introduced. In this phase the configuration software sends frames containing predefined data for long header options or a set of several header options. These predefined data is stored in a buffer within the traffic generator's hardware. During the process of generating frames, these sets of preconfigured data can be selected by an even smaller index, which is part of the variable configuration sets generated by the PC. A possible application of this solution might be the preconfiguration of IPv6 addresses. IPv6 addresses would take 16 byte to be transmitted if they were entirely configurable. It is much more efficient to transmit just an index for a preconfigured lookup table. With a two byte long index, a 65,536 deep table of IPv6 addresses can already be addressed.

B. Configuration Software

As mentioned at the beginning of this section, a software running on a host PC is used to transmit variable sets of configurable header options to the hardware part of the traffic generator. Therefore, a C++ program using the WinPcap library [15] to transmit the configuration sets over an Ethernet link to the described hardware was implemented. First of all, a protocol for these configuration frames had to be defined. Since the hardware of the traffic generator is directly connected to the configuration PC and therefore receives any frame sent from it, no addressing scheme is needed. Hence, no Ethernet header is required. Thus, it is possible to use as many bytes as possible for configuration sets. However, it is possible to implement an Ethernet and an IP header in order to be able to realize a spatial separation of the configuration PC and the hardware part of the generator. Figure 2 depicts the structure of the configuration frames. The first byte of a frame defines how the receiving state machine handles a frame. For example, a frame sets the replay counter, or writes a preconfigured table, or contains

Table I
THEORETICAL CONFIGURATION THROUGHPUT (TP) FOR VARIOUS SETS OF CONFIGURABLE PARAMETERS. THE NUMBERS IN THE FRAME LENGTH FIELD DEFINE THE AVERAGE LENGTH OF THE RESULTING FRAMES.

	Set 1	Set 2	Set 3
Frame length (2 Bytes)	X	X	X
IPv4 Address (4 Bytes)	X		X
IPv6 Address (16 Bytes)		X	
VLAN Tag (2 Bytes)		2X	
LUT Index (2 Bytes)		X	X
Set Size in Bytes	7	25	9
Config. TP [Mbit/s] (avg. length 60 Bytes)	85	303	109
Config. TP [Mbit/s] (avg. length 500 Bytes)	14	49	18

a set of configurable options to generate frames. In the usual case of sending configurable options, this operation byte is followed by configuration sets to be written in the afore mentioned operation FIFO. The length of each configuration set depends on the header options it determines. Hence, each set starts with the control byte that indicates the structure of the configuration set i.e., which options it configures. A variable number of configuration sets can be transmitted. Configuration sets are always transmitted using maximum length frames as each Ethernet frame implies an overhead. This overhead results from the inter frame gap, preamble, start of frame delimiter, and the frame check sequence.

Since the mechanism of amplifying configuration bandwidth to full 1 Gbit/s bandwidth was briefly summarized in the beginning of this section, the following description shall investigate it more detailed. To simplify the examinations, a configuration frame is assumed to include parameter sets of the same length. Table I depicts different scenarios of configurable header options.

Each column of the table illustrates a possible configuration set for the IPclip use case. Header options marked with a cross are determined by the configuration PC and therefore transmitted to the FPGA. The table also depicts the length of the resulting configuration set in bytes. The control byte of each configuration set has to be considered for the length of the set as well. Based on the length of a configuration set, the bandwidth of the configuration stream is evaluated, which is necessary to maintain a traffic stream of 1 Gbit/s at the output of the hardware part. Further overhead that has to be considered consists of inter frame gap, preamble, start of frame, and the FCS. Since the length of frames within the generated Ethernet stream has the most important influence on the required configuration bandwidth Table I depicts the required bandwidth once for minimum length frame and as a second example for a resulting Ethernet stream with an average frame length of 500 bytes. As apparent in a traffic stream consisting of longer frames on average, the demands to the configuration bandwidth and therefore to the configuration PC dramatically decreases.

Following, we describe the traffic generator's integration into the hardware development process of PPS. During the functional test of the PPS, before it is synthesized for an FPGA, it has to be functionally verified. Therefore, a testbench, which generates several different frames has already been implemented. This testbench is standard C respectively C++ code enriched with some SystemC code to connect it to the VHDL design process. The testbench's algorithmic part, determining the test patterns, can be reused for the configuration software of the proposed traffic generator. This part generates variable header options to test the PPS. Instead of further building frames from these information transmitted to the PPS's VHDL description within a simulator, variable header options are stored into the previously described configuration sets. These configuration sets are transmitted to the described hardware, where corresponding frames are transmitted to the physical prototype of the PPS.

V. EVALUATION

In this section, the achieved results for different scenarios are discussed. Furthermore, these results shall be compared with results achieved with packETH. In advance, the main parts of the test equipment have to be mentioned. The described hardware is synthesized for a Xilinx Virtex 4 FPGA (XC4VFX20). As a host PC for running the configuration software a Pentium D at 2.8 GHz with 2 GB RAM running Windows XP SP3 was used. To evaluate the performance of packETH, version 1.6 running on the same PC executing Ubuntu 8.04 is used.

First of all, throughput the software generator packETH achieves was measured for different frame length. As depicted in Table II, the longer the generated frames the higher the throughput of packETH on the Linux PC. As apparent from Table II, packETH does not achieve absolutely full link utilization even for maximum frames. However, for all these cases the proposed traffic generator reaches full link utilization independent from the length of generated frames. As pointed out in Section III, the most important requirement for our use case is to ensure full link utilization. This was not possible with packETH even with an increased process priority to minimize scheduling interrupts.

Further tests for several different configuration scenarios implied by the chosen IPclip use case achieved the same results. Since all the frames transmitted with packETH are configured before it starts transmitting frames, its throughput solely depends on the length of created frames. Therefore, the authors do not illustrate any other packETH results.

The output of the traffic generator only depends on a sufficient configuration bandwidth. Though, performance of the configuration software was evaluated for the worst case of 25 bytes long configuration sets (control byte, IPv6 address,

Table II
LINK UTILIZATION OF A 1 GBIT/S ETHERNET LINK.

Frame Size	Link Utilization in %	
	packETH	Traffic Generator
60	18.8	100
120	28.8	100
240	48.1	100
500	81.7	100
1000	99.3	100
1514	99.0	100

2 VLAN tags, frame length, index). As the host PC was able to achieve this theoretically evaluated throughput, the traffic generator is able to reach full utilization of a Gbit link for all other scenarios of the IPclip use case. However, the performance of the proposed architecture heavily depends on performance of the configuration PC. First attempts to test the configuration software show that the PC was capable of providing the required configuration bandwidth. However, the hardware part of the traffic generator was not able to constantly generate the desired traffic. Further investigations show that scheduling delays on the configuration PC interrupt the configuration stream for short periods so that the FIFO, which buffers the configuration sets on the FPGA, runs empty. This issue could be solved by increasing the process priority of the configuration software on the PC.

VI. CONCLUSION

This paper has introduced an architecture for a traffic generator for high throughput performance tests of packet processing systems. To ensure flexibility and a guaranteed full link utilization, the proposed architecture consists of a combination of hardware and software. Furthermore, this architecture integrates well into the hardware design process of packet processing systems by reusing an the therefore implemented testbench. Compared to already existing high performance test equipment, the proposed solution is significantly cheaper as it only requires a medium efficient PC and an FPGA development board including two Ethernet interfaces. As exemplarily shown for packETH, flexible and free software solutions cannot fulfill the authors' requirements to link utilization. This especially applies to the generation of minimum length frames. It could be shown that the developed packet generator achieves full 1 Gbit/s link utilization even for all scenarios implied by the IPclip use case.

REFERENCES

[1] M. Jemec, "packETH – Ethernet Packet Generator," 2011. [Online]. Available: http://packeth.sourceforge.net/

[2] S. Kubisch, H. Widiger, P. Danielis, J. Schulz, D. Timmermann, T. Bahls, and D. Duchow, "Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP," in *Proc. of 1st ITU-T Kaleidoscope Conference*, May 2008, pp. 375–382.

[3] ZTI, "Traffic Generator and Measurement Tool for IP Networks (IPv4 & IPv6)," 2008.

[4] S. Zander, D. Kennedy, and G. Armitag, "KUTE – A High Performance Kernel-based UDP Traffic Engine," Swinburne University of Technology, Tech. Rep. 050118A, January 2005.

[5] N. Bonell, S. Giordano, and G. Procissi, "BRUTE: A High Performance and Extensible Traffic Generator," in *Proc. of SPECTS*, ser. 37 (3), 2005, p. 839845.

[6] J. Laine, S. Saaristo, and R. Prior, "RUDE & CRUDE: Real-time UDP Data Emitter and Collector," 2008. [Online]. Available: http://rude.sourceforge.net/

[7] Naval Research Laboratory, "The Multi-Generator (MGEN) Toolset," 2008. [Online]. Available: http://cs.itd.nrl.navy.mil/work/mgen/

[8] S. Avallone, A. Pescape, and G. Ventre, "Analysis and Experimentation of Internet Traffic Generator," in *Proc. of New2an'04*, 2005, pp. 70–75.

[9] A. Abdo and T. J. Hall, "Programmable Traffic Generator with Configurable Stochastic Distributions," in *Proc. of Canadian Conference on Electrical and Computer Engineering*, 2005, pp. 747–750.

[10] J. M. Claver, P. Agust, G. Len, and M. Canseco, "A Reprogrammable and Scalable Multimedia Traffic Generator/Monitor on FPGA," in *Proc. of International Conference on Field Programmable Logic and Applications*, 2007, pp. 567–570.

[11] G. Antichi, A. D. Pietro, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "Design of a High Performance Traffic Generator on Network Processor," in *Proc. of 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, 2008, pp. 438–441.

[12] "Ixia - Leader in Converged IP Testing," 2011. [Online]. Available: http://www.ixiacom.com/

[13] H. Widiger, S. Kubisch, P. Danielis, J. Schulz, D. Duchow, T. Bahl, and D. Timmermann, "IPclip: An Architecture to restore Trust-by-Wire in Packet-switched Networks." Montreal, Quebec, Canada: The 33rd IEEE Conference on Local Computer Networks (LCN), October 2008.

[14] P. Danielis, S. Kubisch, H. Widiger, J. Schulz, D. Duchow, T. Bahls, D. Timmermann, and C. Lange, "Trust-by-wire in packet-switched ip networks: Calling line identification presentation for ip (hardware prototype demonstration)." University Booth, Munich, Germany: DATE 2008, March 2008.

[15] G. Varenni, L. Degioanni, F. Risso, and J. Bruno, "WinPcap, The Packet Capture Library and Network Monitoring Library for Windows," 2010. [Online]. Available: http://www.winpcap.org/

Self-synchronizing One-way Delay Measurement in the Internet

Frank Eyermann
Institut für Technische Informatik
Universität der Bundeswehr
Munich, Germany
Frank.Eyermann@unibw.de

***Abstract*—End-to-end packet delay is the network parameter with maximum impact on performance of distributed applications. This is especially true for soft real-time applications, which are delay-sensitive by definition, but also for applications relying on the TCP protocol whose sliding window mechanism performs badly in case of high packet delays. Therefore, measuring packet delay is an important task for both network operators and application developers. This paper presents a tool set for measuring and evaluating one-way end-to-end delay and packet loss that can be operated on standard PCs without additional external timing sources. We chose a script-based approach that can even be executed on virtualized platforms. The self-synchronization mechanism embodied in the trace evaluation is a distinctive feature that omits the need for expensive external clocks (as e.g., GPS receivers). We also show a wide-ranging set of measured traces and their most prominent statistical properties.**

Keywords-One-way delay, packet loss, measurement, tools

I. Introduction

Even though first mechanisms for ensuring quality of service (QoS) in IP networks have been proposed almost 15 years ago [3], they are still rarely used in the Internet. In general, only best-effort services that treat all packets equally and do not respect special requirements of single packets are available.

However, applications requiring an elevated level of QoS, as e.g., Voice-over-IP (VoIP), IP Television (IPTV), or Video-on-Demand (VoD), become more and more important for private users but also for Business-to-Customer (B2C) and Business-to-Business (B2B) communication. These applications only work satisfactorily if one-way end-to-end delay, packet loss, delay variation (also called jitter), and/or throughput are above or below a certain threshold. As techniques for ensuring these QoS parameters are still not embodied in today's networks, users can only "hope" that the network has sufficient performance.

Therefore, it is important to test regularly the actual performance of the Internet with respect to the above mentioned performance parameters. This work presents a tool set for measuring one-way delay, delay variations, and loss. The tools have already been used to capture a wide-ranging set of traces in EmanicsLab [4], [5].

The paper is structured as follows: First, in Section 2 we present a summary of the requirement analysis for the tools, followed, in Section 3, by related work on this topic. Later, in Section 4 we describe the tools we have developed for measuring traces and evaluating them. Section 5 discusses measured traces and their evaluation. Finally, Section 6 concludes the paper and summarizes the main results.

II. Requirements

Analyzing suitability of networks for multimedia or real-time services requires testing the network over a long period of time in order to cancel different load situation as regularly observed at different times of day or week. As continuous measurement would generate an enormous amount of measurement data, it may be preferable measuring short intervals of a few minutes scattered over a period of days or weeks. This requires the capability to flexibly schedule measurement runs. The number of packets sent during such a measurement run, the size of the packets as well as the frequency of packet generation has to be easily configurable. Furthermore, for documentation purposes and for easier repetition of experiments a script-based approach would be beneficial.

In addition, there are a number of non-functional requirements. First, the tools should be executable on different operating systems, including at least Linux and Windows. As more and more servers—especially in testbeds—are virtualized, the tools have to be tested on such platforms, too.

Precise one-way delay measurement is typically performed using additional hardware for synchronizing the sender and receiver host (e.g., GPS receivers). This increases efforts and costs drastically especially if cables have to be installed.

Synchronization, however, is not necessary if only relative delay or inter-packet delay variation is of interest: Packet delay consists of static and dynamic delay components: Static components are propagation, serialization, and processing delay [7]; Queuing delays are dynamic components. Even though the static components cannot be neglected, these constant values are only troublesome in case of satellite communication and do typically not exceed some tenth of milliseconds. More problematic is the dynamic part of the delay as, first, its share might be bigger than the static one and, second, the changes in delay lead to unpredictable arrival times at the receiver. Therefore, the tools should also be able to work with unsynchronized hosts and be able to measure the dynamic delay components.

III. Related Work

This section presents related work on delay, delay variation, and loss measurement in the IP networks.

A. IPPM

The goal of the *IP Performance Metrics working group* (IPPM WG) of the Internet Engineering Task Force (IETF) is to define metrics that can be applied to the quality, performance and reliability of Internet data delivery services [9]. In addition, the working group defined a general framework for accurately measuring and documenting the metrics [13]. The IPPM WG does not define or suggest how the performance parameters are measured. They emphasize on definitions and the unambiguous understanding what a parameter expresses, so that measurement results can be compared, shared and validated by different entities.

B. Measurement tools

Quite a number of tools for delay and loss measurement are available, including, e.g., *ping*, *cing* [1], *king* [8], *netperf* [11], or *scriptroute* [16]. All tools use probing techniques, i.e., they inject artificial packet (so-called probes) into the network and observe their behavior. A sub-group of these tools uses so called inference techniques: While delay measurement in general requires two programs, i.e., a sender that generates the probes, and a receiver that evaluates the probes, tools embodying inference techniques use standard behavior of protocol stack implementations on nodes in the network to receive feedback. Thus, these tools can combine sender and receiver functionality in one program but can only measure round-trip delay and not one-way delay. Examples for such programs include *ping* or *traceroute*.

None of the programs mentioned above is able to perform flexible script-based long-term one-way delay and delay variation measurements. Either the tools use inference techniques that by definition cannot measure one-way performance or their design does not include the possibility to schedule measurement runs. Furthermore, none of these tools have been tested on virtualized platforms or with unsynchronized hosts.

IV. Measurement Tool

A. Tool design

One-way delay, delay variation, and packet loss measurement requires a pair of programs: a sender, generating the probing packets as well as a receiver, collecting the probes and writing a log file.

All time intervals and timestamps are stored and transferred in units of 100µs. This value is a compromise between timer resolution and storage space. On the one hand, sub-milliseconds resolution is approximately one magnitude smaller as typical measurement values, and therefore, the effect of the rounding error is negligible. On the other hand, a signed 32-bit integer counting steps of 100 µs overflows only every 60 hours – long enough to detect any overflow of counters.

Measuring such small time differences is not possible using the built-in real-time clock (RTC) of PCs. Furthermore, accessing the RTC is quite slow thereby reducing program performance. Intel invented a quickly accessible, high-resolution timing source for their Pentium processor. The TSC (time-stamp counter) is a 64-bit processor register counting its clock cycles. The time resolution of this register is more than sufficient (on a 1 GHz processor, the register counts microseconds) and access to this processor register takes only a couple of processor cycles. All other x86 processor manufacturers later adopted the TSC for their processors [12].

The drawback of using this counter is the varying processor speed from computer to computer, and accordingly, the necessity to calibrate the TSC in order to produce comparable results. Furthermore, state-of-the-art processors may reduce their speed for preserving power in times with low load. This also influences the TSC counter. Therefore, during calibration and during send cycles busy waiting is necessary in order to prohibit power-saving features.

B. Scripting language definition

Deterministic probe sending schedules can be flexibly described using the following four script commands implemented by the tool:

- `at` *absoluteTime*
  waits until *absoluteTime*. The parameter time can be a real point in time (e.g., 13:05:23), or the syntax `*/n` can be used for hour, minute, or second. In this case `at` will wait until the current hour, minute, or second, respectively, can be divided through *n* without remainder (e.g., `at */2:00:00` will wait until the next full even hour).
- `send` *destAddress repeat size delayMillis*
  sends a burst of probes. Probes of *size* bytes are sent every *delayMillis* milliseconds to IP address *destAddress* for *repeat* times.
- `Loop` *iter*
      body
  `bend`
  processes the block *body* for *iter* times. If *iter* is 0, the command loops infinitely. *body* can be a series of commands including other loop statements.
- `wait` *millis*
  waits for *millis* milliseconds.

Respective scripts are interpreted by `Lattes`, a C++ application described in Section D.

C. Structure of Probes

The probes consist of three unsigned 32-bit integers as well as random padding data filling the rest of the packet. The first integer is the *Flow ID*. Each burst gets its own *Flow ID*. During start-up of `Lattes` a *Flow ID counter* is initialized with a value based on the current time. Every time `Send::doit()` is called the *Flow ID* is incremented by 1. The second integer is the *Serial number*, starting from 1 for the first probe of a burst and incremented by 1 for each probe sent. And finally, the third integer stores the time on the sender in units of 100 µs when the probe was generated. The probes are sent as UDP packets; therefore, the actual size of the IP packet is 28 bytes larger than stated in the *size* parameter.

D. Lattes

`Lattes` is an interpreter of the scripting language defined above. This section presents the design, implementation, and interfaces of `Lattes`. The receiver of the probes, `Latreceiver`, will be described in the following section.

Each command is implemented as a class inheriting from the class `Command` (see Fig. 1). The class `Command` has a virtual function `doit()`, which must be implemented by each not abstract child class and contain the functionality of the command. Based on this approach `Lattes` can be easily extended by simply implementing a new class inheriting from `Command`.

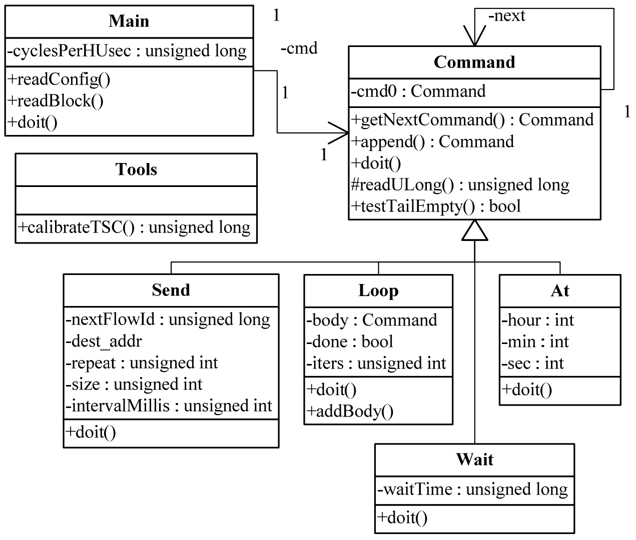


Figure 1. Class diagram for Lattes

During start-up of the program the `Main` class reads in the script. The method `Main::readBlock` reads line by line from the script and compares the start of a line with the command literals. If one command literal is recognized, the respective object is created and the rest of the line is handed over to its constructor, which evaluates the command's parameters. The objects are stored as a linked list.

Furthermore, the `Main` class calls `Tools::calibrateTSC()`, a static method that measures the increase of the TSC register per time unit.

After reading in the entire script and constructing all `Command` objects, the main program calls the `doit()` method of the first command. Each `doit()` method calls `doit()` of the next command in the chain after performing its functionality.

The `loop` command is the only command manipulating the control flow of the script. The class `Loop` has an additional attribute `body` that points to the first command of the body. The `next` attribute of the `Loop` class points to the first command following the corresponding `bend` statement. The `doit()` method of the class `Loop` calls `body->doit()` for *iter* times before continuing with the next command after `bend` (*i.e.*, `next->doit()`).

E. Latreceiver

`Latreceiver` was developed as receiver for the probes sent by `Lattes`. Common functions (*e.g.*, calibration of the TSC values or probe format definition) are shared between both programs. The program writes all events to a log file.

`Latreceiver` maintains a list of all currently running flows. If a probe is received, this list is searched for the Flow ID stored in the probe. If the ID cannot be found, the start of a new flow is detected and a new record containing information on the received packet is added to the list as well as a respective remark is written to the log file.

If a lost packet is detected, i.e., the serial numbers of two probes of the same flow are not consecutive, a remark is written to the log file. For consecutive packets the packet-to-packet delay variation Δt_i^{ptp} , i.e., the difference of the delay of the previous packet and the current packet, as well as the (not-normalized) relative delay Δt_i^{rel} is calculated and written to the log file (see following section). Reordered packets that have been passed by a successive packet are dropped by the receiver.

EmanicsLab uses the *MyPLC platform* [14] consisting of virtualized Linux systems. The virtualization produces a long latency between receiving a frame on the physical interface and the processing of the packet in user-space. This leads to linear dependency between consecutive probes as depicted in Fig. 2. Only virtualized systems show this effect.

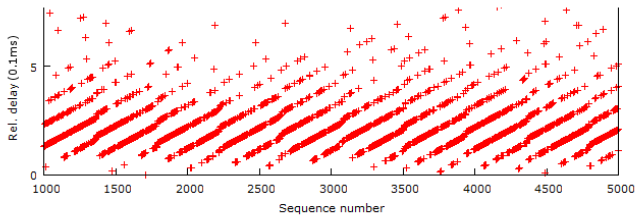


Figure 2. Trace collected on virtualized host

As discussed in [16] and [10], these distortions can be minimized or even eliminated if time stamping is already performed in the interrupt service routine, which copies the frame from the physical network device to memory. Consequently, `Latreceiver` uses the *pcap library* [18] for time stamping the packet already in the kernel allowing the usage of the programs also on virtualized systems.

`Latreceiver`'s main focus is on probe reception. All further processing, *e.g.*, splitting the log file in files containing only information about one flow, is task of post-processing tools described in subsequent sections.

F. Measurement value evaluation

`Latreceiver` as well as some post-processing tools are used to calculate different parameters from the measured values. `Latreceiver` directly calculates the packet-to-packet delay variation Δt_i^{ptp} and the (not-normalized) relative delay Δt_i^{rel} (see Equations 1 and 2)

$$\begin{aligned}\Delta t_i^{Sender} &= t_i^{Sender} - t_{i-1}^{Sender} \\ \Delta t_i^{Receiver} &= t_i^{Receiver} - t_{i-1}^{Receiver} \\ \Delta t_i^{ptp} &\equiv \Delta t_i^{Sender} - \Delta t_i^{Receiver} \\ &\equiv t_i^{Sender} - t_{i-1}^{Sender} - \left(t_i^{Receiver} - t_{i-1}^{Receiver}\right)\end{aligned} \qquad (1)$$

$$\Delta t_i^{rel} \equiv \sum_{j=0}^{i} \Delta t_j^{ptp} \qquad (2)$$

with t_i^{Sender} being the sending time of probe i as stored in the probe and $t_i^{Recevier}$ the point of time probe i was received. The variable Δt_i^{rel} denotes the accumulated delay variation, and thus, represents the dynamic component of the packet delay.

The term Δt_i^{ptp} can be positive or negative. If queues in routers grow, the packet-to-packet delay variation is positive as successive packets spend more and more time in router queues. In case the queues are shrinking, packet-to-packet delay variation is negative as each packet spends less time in router queues.

Assuming that the network is only temporarily over-loaded and router queues are empty at some time the relative delay can be normalized in a way that the smallest value for the relative delay is 0 (see Equation 3):

$$\overline{\Delta t_i^{rel}} \equiv \Delta t_i^{rel} - \min_j(\Delta t_j^{rel}) \qquad (3)$$

The relationship of the variables Δt_i^{ptp} and $\overline{\Delta t_i^{rel}}$ is visualized in Fig. 3. In this example the sender generates probes every 20 ms. These probes arrive at the receiver with different delays. In the example the network was overload in the time from t_1^{Sender} to t_3^{Sender}, and thus router queues grew; afterwards, until t_5^{Sender} the queues empty again.

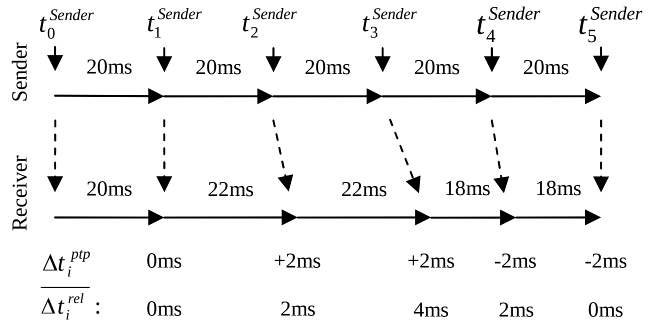


Figure 3. Impact of queueing on measurement variables

G. *Post-processing*

Fig. 4 shows the packet-to-packet delay variation Δt_i^{ptp} and the normalized relative delay $\overline{\Delta t_i^{rel}}$ of a flow consisting of 7000 probes (approx. 5 min) sent between two nodes.

In contrast to the theoretical concepts discussed above, the relative delay is increasing. All traces captured look similar; however, the slope of the relative delay changes from trace to trace including also negative slopes.

The reason identified is synchronization errors resulting from two factors: The first-order error of real-time clocks is not zero, *i.e.*, the time difference between two clocks is not constant but gets greater or smaller with time. This effect is called skew and can easily be observed on most clocks. This error, however, is typically much smaller than the error observed in the traces. In Fig. 4 the error is 110 ms per 300 s or approx. 30 seconds per day.

Additional measurement errors might be introduced because time at sender and receiver is measured in processor cycles and is then converted to real time units. Each time one of the programs is started the conversion factor is calculated with the help of a calibration routine (see Section IV.D). This calibration might add an additional error component.

Other reasons for changing delay, like path changes or traffic shaping by providers, cannot be held liable for this effect, as they do not cause continuously increasing or decreasing relative delay over minutes.

A more accurate calibration of clocks that would not only eliminate the clock offset but also the skew is only possible with additional hardware. But as the error is linear in time it also can be corrected *ex post*. Therefore, for each flow the slope of the dynamic delay has to be determined. This can be done by fitting a line onto the lowest delay values. This line represents a dynamic delay of zero.

The line and all observations are then projected to the x axis. This converts Fig. 4 into Fig. 5 (packet-to-packet delay variation is not shown as it does not change; scale of y axis is adapted).

H. *Self-synchronization*

Fitting the line to the observation values is not a trivial task. Quite a lot of algorithms exist for fitting straight lines into a cloud of observations (*e.g.*, ordinary least square estimation, OLS) but all these assume positive and negative

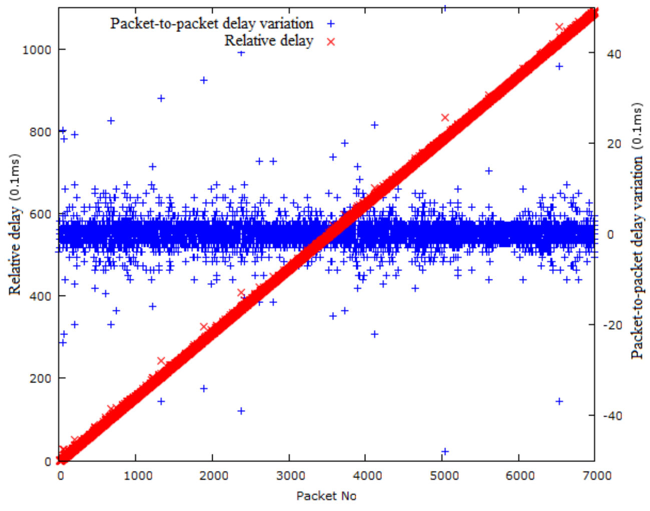


Figure 4. Sample trace showing the skew in relative delay

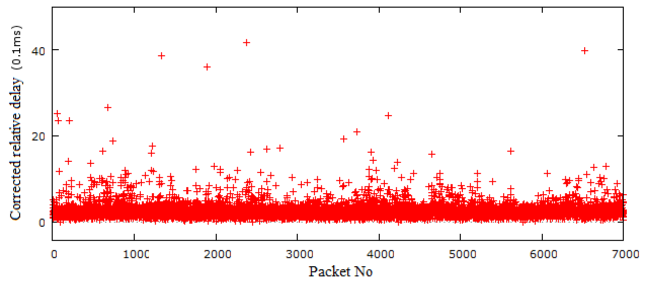


Figure 5. Corrected relative delay

variations from the mean (and not only positive as in the present case). For this work several approaches were developed and tested. The one that shows the best results iteratively searches from the right and left side for a pair of points (p_1, p_2), which best represent the line. The algorithm works as follows:

The smallest observations in a 1% interval from the right and from the left, respectively, are chosen as start values for p_1 and p_2. Based on the line through those both points, the observations are transformed for the first time. This typically produces also negative observations, as the guessed p_1 and p_2 have not been optimal.

The solution is improved by choosing the smallest point p' of the transformed observation set and replacing either p_1 or p_2, depending if p' is in the right or left half of the set. Afterwards, the set is transformed according to the newly constructed line again. This step is repeated as long as there are negative observations. Regularly only a few iterations are necessary.

This procedure describes a very robust and fast algorithm. Several thousand measurement traces (see next chapter) have been processed and afterwards their plausibility was checked statistically. The algorithm is robust to outliners as in the present scenario outliners can only be positive, the algorithm, however, considers only the lowest measurement values. Furthermore, the algorithm is universal as it does not rely on the structure of the data or any input parameters.

V. TRACES

With the help of the tools developed a wide ranging set of delay measurements is performed. All measurement runs are collected in EmanicsLab.

A. *EmanicsLab*

EmanicsLab is a European research network consisting of 20 nodes at 10 sites across Europe (see Fig. 6). EmanicsLab partners use the network for research activities in the area of network and service management, including distributed flow collection and analysis, distributed intrusion detection systems, as well as distributed monitoring and accounting systems. It is funded by the *European Network of Excellence for the Management of Internet Technologies and Complex Services (EMANICS)*.

EmanicsLab is based on MyPLC, the backend management infrastructure of PlanetLab [14]. Project partners can run their services and applications in own *slices*. A slice is a fraction of resources on a set of nodes implemented as virtual machines [2]. The nodes run a customized Linux operating system. Access control and establishment of slices is controlled remotely by a central management system. Project partners typically get root access to their slice.

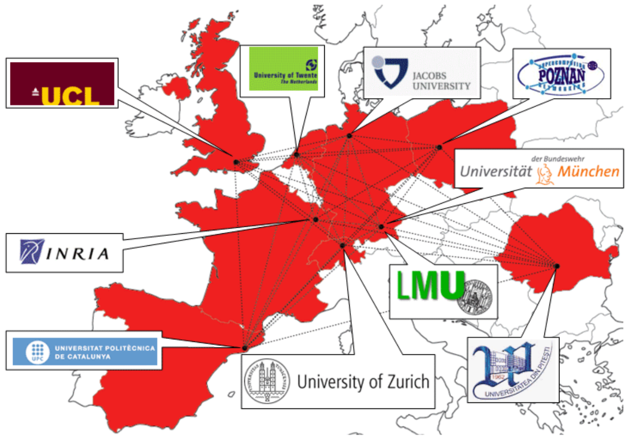


Figure 6. EmanicsLab nodes in Europe ([4])

B. *Measurement runs*

Seven out of ten sites of EmanicsLab have been participating in trace collection. The other three institutes joined EmanicsLab only after the measurements have already begun. In total, five different test runs have been scheduled. All tests are structured similarly in order to produce comparable results. Each run measures 36 times for five minutes the 42 unidirectional vertexes in a fully-meshed graph of all participants in a time frame of 72 hours. This adds up to 1512 measurement runs per test. The schedule of runs within a test ensures that one station is either sender or receiver of probes at one instance of time. In total about 18 Mio single measurements have been performed.

TABLE I. TRACE OVERVIEW

Name	Burst size (# of packets)	Probe size (byte)	Interval (ms)
Test01	30000	60	10
Test02	7000	60	50
Test03	3500	60	100
Test04	150000	60	3
Test05	250000	60	2

C. *Measurement results*

The measured data contains a huge amount of information. The data has already been used for the assessment of a multi-domain auditing system for end-to-end SLAs [6]. In the following some statistical properties as a proof of concept for the measurement tools are shown. These data, however, could also be helpful for application developers and network operators.

1) *Moments*

Fig. 7 and Fig. 8 show the mean, the 95%-, the 97.5%- and the 99%-percentiles of $\overline{\Delta t_i^{rel}}$. Percentiles are used instead of standard deviation or variance as the distribution of the packets typically belongs to the class of so-called heavy-tailed distributions, which do not have a finite standard deviation. Percentiles do not suffer from this effect.

As can be seen from the figures, for approx. two thirds of the measured paths the relative delay is negligible (the relative delay of 99% of all packets is between 1 and 2 ms). For some of the measured paths these percentiles are 100 times bigger. The reasons are heavily loaded Internet connection at two sites. Tests scheduled after these sites have upgraded their uplink show a more homogenous result.

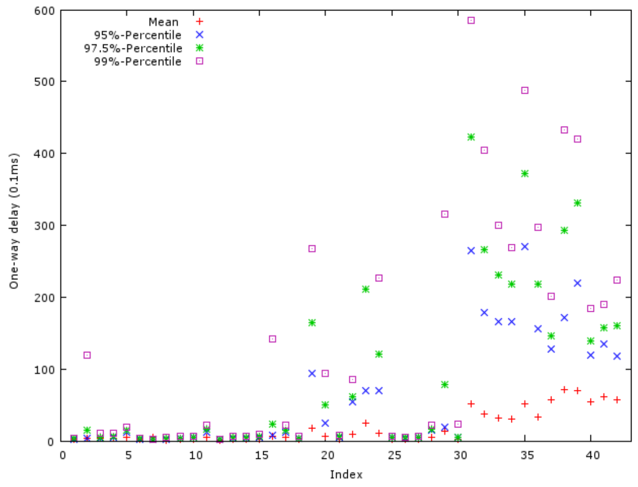


Figure 7. Mean, 97.5%- and 99%-Percentile of Test01

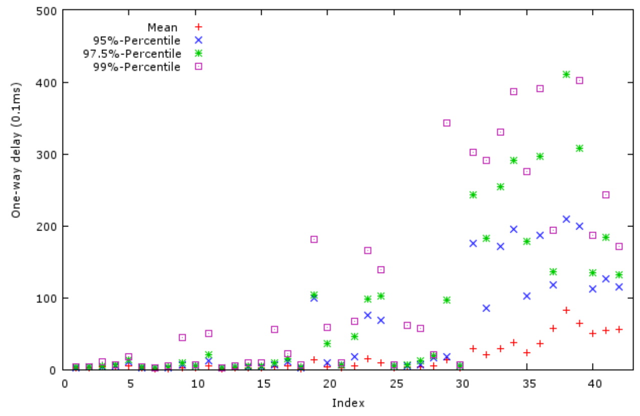


Figure 8. Mean, 97.5%- and 99%-Percentile of Test02

VI. Conclusion

This paper presented a tool set for measuring one-way delay and delay variations. Additional hardware for clock synchronization might be used but if not available as a fall-back the tools may remove first-order error of the clock, i.e., skew, in a post-processing step. Furthermore, the tool set is script-based allowing automated measurements over a longer period of time. Furthermore, the tools have been tested on virtualized platforms as frequently used in testbeds (e.g., PlanetLab).

In EmanicsLab, a testbed arisen out of the Network of Excellence EMANICS, a wide ranging set of measurement traces with over 18 Mio singleton measurements have been performed.

Acknowledgment

The authors wish to thank the members of the Chair for Communication Systems and Internet Services at the Universität der Bundeswehr Munich, headed by Prof. Dr. Gabi Dreo Rodosek, for helpful discussions and valuable comments on previous versions of this paper. The Chair is part of the Munich Network Management Team.

References

[1] Anagnostakis, K., Greenwald, M., and Ryger, R.; “cing: Measuring Network-Internal Delays using only Existing Infrastructure”, 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM), pp. 2112-2121, 2003.

[2] Bavier, A., et al; “Operating system support for planetary-scale network services”, Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation 2004 (NSDI'04), pp. 19-19, San Francisco, USA.

[3] Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S.; “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification”, RFC 2205, Sept 1997

[4] EmanicsLab, subproject of the European Network of Excellence for the Management of Internet Technologies and Complex Services (EMANICS), www.emanicslab.org

[5] European Network of Excellence for the Management of Internet Technologies and Complex Services (EMANICS), Project Number: FP6-IST #026854, www.emanics.org.

[6] Eyermann, F.; “An Auditing System for Multi-for Mulit-domain IP Carrying Service Level Agreements”, Doctoral Dissertation, unpublished.

[7] Filsfils, C. and Evans, J.; “Engineering a multiservice IP backbone to support tight SLAs”, Computer Networks, Volume 40, Issue 1, pp. 131-148, September 2002.

[8] Gummadi, K., Saroiu, S., and Gribble, S; “King: Estimating Latency between Arbitrary Internet End Hosts”, SIGCOMM Internet Measurement Workshop, pp. 5-18, 2002.

[9] IETF IP Performance Metrics working group, http://www.ietf.org/html.charters/ippm-charter.html

[10] Jain, M.; “Probing for Bandwidth Measurements”, article of the Planetlab-users mailing list from May 11th, 2005.

[11] Netperf homepage, http://www.netperf.org/netperf/NetperfPage.html, visited Jan. 2011.

[12] Pásztor, A. and Veitch, D., “PC Based Precision Timing without GPS”, Proceedings of the ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems, Marina Del Rey, pp. 1-10, June, 2002.

[13] Paxson, V., Almes, G., Mahdavi, J., and Mathis, M.; “Framework for IP Performance Metrics”, RFC 2330, May 1998

[14] PlanetLab, An open platform for developing, deploying, and accessing planetary-scale services, www.planet-lab.org

[15] Scharf, M.; “The Impact of Delay Variations on TCP Performance”, Proceedings of the 2nd Workshop on Modeling and Optimization in Mobile, Ad hoc and Wireless Networks (WiOpt '04), pp. 419-420, Cambridge, 2004.

[16] Spring, N., Wetherall, D., and Anderson, T.; "Scriptroute: A Public Internet Measurement Facility", USENIX Symposium on Internet Technologies and Systems (USITS), pp. 17-17, 2003.

[17] Spring, N., Peterson, L., Bavier, A., and Pai, V.; Using PlanetLab for network research: myths, realities, and best practices, SIGOPS Oper. Syst. Rev. 40(1), pp 17-24, 2006.

[18] TCPDUMP.org http://www.tcpdump.org/, visited Jan 2011.

Towards Optimized Probe Scheduling for Active Measurement Studies

N. Daniel Kumar, Fabian Monrose, and Michael K. Reiter
Department of Computer Science
University of North Carolina
Chapel Hill, NC, USA
{ndkumar, fabian, reiter}@cs.unc.edu

***Abstract*—Internet measurement studies often require prolonged probing of remote targets to collect information, yet almost all such studies of which we are aware were undertaken without considering the polluting effects of their unconstrained probing behavior. To help researchers conduct experiments in a more responsible fashion, we present a framework and technique that enables efficient execution of large-scale periodic probing without exceeding pre-set limits on probing rates. Our technique employs a novel scheduling algorithm and leverages knowledge of diurnal traffic patterns to make data collection more efficient (e.g., by probing servers only during periods in which probe results will be most useful). We evaluate our technique in the context of a real-world study and show that it substantially outperforms naïve probing strategies for accomplishing the same goal, sending more probes during useful periods, fewer probes overall, and probing at more precise intervals as required by our measurement applications.**

***Keywords*-probe scheduling; responsible network experiments.**

I. Introduction

Over the past several decades, researchers and practitioners have proposed countless techniques for understanding various characteristics about the Internet at large. For the most part, these pursuits have been grounded in empirical measurements that are either passive or active in nature. As their names imply, passive measurements typically involve observations taken from some form of capture device, while active measurements require the injection of specially crafted packets (so-called *probes*) into the network to infer characteristics of the phenomenon under scrutiny.

Recently, there has been a marked increase in the use of active measurements for understanding Internet topology and mapping (e.g., geo-location), path-variance and other end-to-end performance dynamics (e.g., bandwidth estimation), the spread of security-related events (e.g., worm outbreaks), and client and server demographics (e.g., website popularity), etc. To better enable such studies and improve data collection efforts, a number of scalable measurement infrastructures (e.g., CAIDA's Archipelago Measurement [10], Google's M-lab open platforms, and the Flexible Lightweight Active Measurement Environment [25]) have been made available to the research community for controlled experiments. Archipelago, for example, provides a centrally managed framework that supports a distributed shared-memory architecture that can be used to coordinate network measurements across the globe.

A common requirement in many of the Internet-wide studies being conducted today is the need to repeatedly probe some set of targets over time. Collectively, we rely on each experimenter's own restraint to limit the volume of traffic they inject into the network and/or target at a given resource. While some experimenters exercise such restraint to limit collateral damage, in the absence of accepted guidelines for such measurements and tools to enable experiments that respect them, the injection of billions of probes into the network during a short period of time is not uncommon (e.g., [3]); indeed, the literature is rife with examples of arguably egregious practices.

Such disregard for the collateral damage caused by network measurement experiments has raised enough concern that it has led to a call urging the community to move towards a set of best practices for active measurements. As Papadopoulos and Heidemann note [17], as practitioners it should be *our* job to design experiments carefully, in a manner that significantly lowers load without sacrificing measurement fidelity. In this paper, we present a technique for attempting to do just that. Specifically, we propose an efficient scheduling algorithm for probing measurement targets, which is efficient in that it can be tuned towards expeditious completion of the experiment while also observing some predefined maximum probing rate.

Of specific interest to us are studies of Internet demographics (e.g., inferring website popularity rankings [19], [24], exploring the prevalence of malicious domain name system (DNS) servers [6], and client-density estimation [7], [20]) where the fidelity of the experiment can increase if the targets are probed at ideal times. We aim to maximize the utility of each probe that we send, in contexts where the utility of a probe can be correlated with time-of-day, e.g., when it is expedient to probe targets during their peak (or off-peak) traffic periods. Our approach may also be extended for applications requiring that certain subsets of targets be probed contemporaneously, such as RadarGun [2] for IP alias resolution. We show that for experiments taking several days, and given limited probing resources, our scheduling algorithm outperforms several naïve strategies for probe scheduling according to intuitive metrics provided in §IV.

The remainder of this paper is organized as follows. Related work is presented in §II. In §III we outline our scheduling framework and algorithm, which we evaluate in §IV. We conclude in §V.

II. Related Work

As mentioned earlier, some experimenters have exercised restraint to limit collateral damage from the probes they inject. Bender *et al.* [2], for example, attempt to constrain their probing rates to avoid triggering rate limiters. Others [21], [11] have tried to take advantage of opportunistic measurement techniques in order to unobtrusively make network measurements without triggering alarms from intrusion detection systems. Unfortunately, these instances of restraint seem rare, and the academic literature contains numerous examples of experimenters probing at high rates (e.g., 260 packets per second [22]) or injecting high volumes of probes (e.g., 220 million [6] and 27 billion probes [3]),[1] which are arguably indistinguishable from attacks.

Previous work on scheduling active network monitoring activities has focused on preventing the simultaneous scheduling of activities that would interfere with each other's results and lead to inaccurate measurement reports [4], [8], [18]. Here we are not concerned with probes interfering with one another, but we focus rather on respecting a limit on the probe rate we induce on the network, while completing the probing experiment as quickly as possible. That is, because our targets are all distinct, by imposing limits on our probing rate we are able to mitigate (to some extent) packet loss due to network congestion. To complete the experiment quickly, we leverage knowledge of the *useful interval* in which each type of probe should occur, which is characteristic of the type of probing activities considered here but that is not utilized in these prior works. Moreover, we do not assume that information about the underlying network topology is readily available.

Additionally, prior work in this area provides no guidance on how to prioritize probes for scheduling in the absence of network topology, and so is not helpful in our setting. The work of Calyam *et al*. [4] applies the basic *earliest-deadline-first* (EDF) heuristic [15] that we use, but their scheduling is done in an offline setting; they do not utilize efficient data structures to leverage the scalability of the simple online heuristic. Additionally, Calyam *et al.* do not allow for the possibility of dropped probes, and so their algorithm simply fails if all of their probes cannot be issued on time.

Also related to our work (i.e., arranging as many targeted probe sequences as possible to fit within some maximum rate limit) is the literature on perfectly periodic or cyclic scheduling on parallel processors. The fundamental scheduling problems are still NP-hard [16], but efficient approximations [1] could be used for applications not requiring precisely periodic probes.

III. Approach and Assumptions

Our work is motivated by the need to find more efficient ways to schedule network probes. Specifically, we consider the common case wherein the prober needs to issue a series of probes to a given target, j, at some periodic interval, π_j. In some cases, each target might have its own probing period (e.g., the time-to-live- (TTL-) based intervals assumed in various types of DNS cache snooping investigations [19], [24]), or the period might be uniform across all measurement points (as is the case with several ID-based alias resolution studies for creating router-level topology graphs [2], [13], [22]).

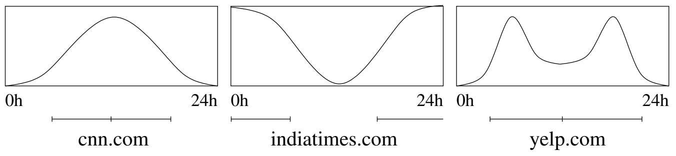


Fig. 1: Hypothetical curves illustrating the volume of DNS requests (for different websites) observed by a DNS resolver. In the cache inspection example, the goal is to probe the resolver about a given website w, but only during peak periods of activity for w.

We also assume that for some of the experiments, there is a notion of an idealized measurement window (i.e., a contiguous window of time) that we call a target's "useful interval", denoted $[s_j, e_j)$. The intuition here is that the fidelity of the experiment may be improved by probing targets during some region of time when we can expect the most utility out of our probes (see Figure 1). In the case of DNS cache inspection, for example, it makes little sense to probe a caching resolver [23] to infer density-based estimates of client populations using the server when most of its clients may be asleep or idle [7]. (We do not adapt these windows dynamically, although doing so would be useful for applications such as bandwidth estimation.) Lastly, we assume that the experiment dictates that a certain number of probes be sent to each target.

For the remainder of the paper, we assume that network traffic follows diurnal patterns—i.e., it is largely similar from day to day, but not from hour to hour, and it exhibits no major differences from week to week. Accordingly, we schedule probes to our targets in real time, using these diurnal traffic patterns. Our solution makes use of a simple and efficient priority queue, described in §III-C.

A. Constraints

Our scheduling algorithm must meet the following constraints:

1) periodic probes must be evenly spaced at intervals of $(\pi_j + \delta)$, for some relatively small δ compared to the periodicity π_j;
2) a predetermined number of probes N_j must be issued to each target by the end of the probing experiment; and
3) no more than L probes per second may be sent.

In what follows, we meet the first two constraints heuristically, but observe the probing rate limit constraint strictly.

B. Goals

Ideally, we would like to probe each unfinished target (a target j to which we have sent fewer than N_j probes) every day at times $\{s_j, s_j{+}\pi_j, s_j{+}2\pi_j, \ldots, \sim e_j\}$ within the target's useful interval $[s_j, e_j)$ until we have sent N_j probes to each target or the time for the experiment is exhausted. This would allow us to achieve perfect periodicity of π_j between our

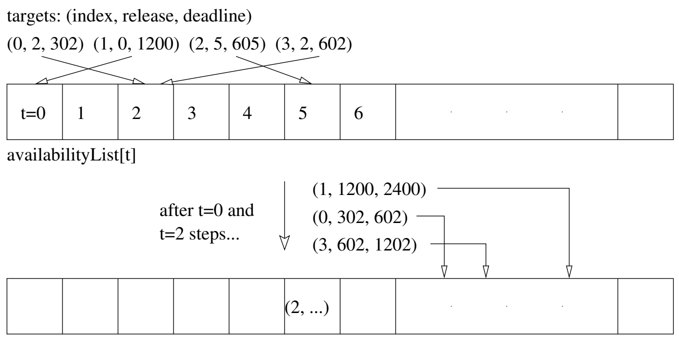


Fig. 2: An individual probe to a target j is initially placed into `availabilityList[`s_j`]` according to the beginning of its useful interval $[s_j, e_j)$. If the probe is issued, the succeeding probe to target j is placed into `availabilityList[`$s_j + TTL_j$`]`.

probes, and we would conserve probes by only issuing them within $[s_j, e_j)$. Unfortunately, with a large number of targets, there is no guarantee that we can do this without exceeding our rate limit of L probes/sec at some point in time. Since we must observe our probing rate limit (3) strictly, the other two constraints are met only heuristically.

Not every unfinished target will be probed on every day, but if a target j is probed on a given day, we require that it be hit with a sequence of (at most $\lceil \frac{e_j - s_j}{\pi_j} \rceil$) probes within the time interval $[s_j, e_j)$, such that in every fixed (non-sliding) window of duration π_j beginning at time s_j, there will be exactly one probe to the target j, until e_j is reached or a probe is dropped (described below). By the end of the experiment, up to N_j probes will have been issued to each target j, but no more.

C. Earliest-Deadline-First (EDF) Scheduling

We characterize a single probe by the tuple (j, r_j, d_j), where j is the target to be probed, and $[r_j, d_j)$ is the fixed window of duration π_j within which the probe should be issued. Thus, the first probe in a sequence is $(j, s_j, s_j + \pi_j)$, the second $(j, s_j + \pi_j, s_j + 2\pi_j)$, and so on. Recall that we do not enforce the periodicity π_j strictly; in principle, two probes may be as close together as 1 timestep or as far apart as $2\pi_j - 1$ timesteps.

We initialize our algorithm by populating an array called `availabilityList` of length $T = 86400$ (seconds in a day) with the first probes for each target, assigning a probe $(j, r_j = s_j, d_j = s_j + \pi_j)$ to position `availabilityList[`s_j`]`, the time at which the probe will become available for issuing. Starting at time $t = 0$, we push the probes from `availabilityList[`$t \bmod T$`]` onto a priority queue `Q` ordered by increasing d_j—"earliest deadline first". In our notation, the time t is definite and monotonically increasing; the interval edges $\{s_j, e_j, r_j, d_j\}$ are all within $[0, T)$ and represent times between 00:00:00 and 23:59:59.

At each timestep t, we pop and send as many probes as possible from `Q` without exceeding our rate limit L. For each probe (j, r_j, d_j) that we send, we place its successor probe $(j, d_j, \max(d_j + \pi_j, e_j))$ into `availabilityList[`d_j`]` if the target j remains to be probed. We thus aim to issue the successor probe in the π_j-length window immediately following $[r_j, d_j)$. If the first probe (j, r_j, d_j) is issued near the end of the window $[r_j, d_j)$, this means a lot of probing resources are being consumed around time t, and the succeeding probe is likely to be issued near the end of the window $[d_j, d_j + \pi_j)$, so that the probing periodicity π_j is attained heuristically.

The probes left in `Q` are carried over onto timestep $(t + 1)$, and then combined with the probes pushed onto `Q` from `availabilityList[`$t + 1$`]`. If a probe to j is dropped, i.e., it is popped off of `Q` at a time $t \geq d_j$, this means that the rate limit has been reached for the last π_j timesteps, and remaining probes to j are postponed until t re-enters the useful interval $[s_j, e_j)$ the following time, in order to ease the workload and reduce probe delay. More precisely, if a probe (j, r_j, d_j) is dropped, the probe $(j, s_j, s_j + \pi_j)$ is placed into `availabilityList[`s_j`]` and reconsidered at time $t \equiv s_j \pmod{T}$.

Thus, probes may be dropped from the ends of sequences, but never from the middle; we succeed in guaranteeing that there will be exactly one probe sent to j in every fixed window of length π_j from s_j until the end of the probing sequence. We also strictly observe our probing rate limit L (constraint (3) in §III-C). Our heuristic success in evenly spacing our probes at $(\pi_j + \delta)$ intervals (constraint (1)) and sending N_j probes to each target (constraint (2)) is illustrated in §IV.

D. Complexity

The use of a priority queue with amortized constant-time pushes and log-time pops (in the length of the queue) makes our approach scalable: for an experiment of duration D days, our worst-case time complexity is O($J \cdot N \cdot D \cdot \log J$), where J is the number of targets to probe and $N \doteq \max_j N_j$ is the maximum number of probes to send to a target. When scheduling online at each timestep, the time complexity is O($J \log J$). The space requirement of our algorithm is O(J).

IV. Evaluation

We evaluate our approach via a simulation of a week-long measurement experiment that aptly demonstrates a type of Internet-wide study which uses DNS cache inspection [9] to infer demographic information. For example, both Wills *et al.* [24] and Rajab *et al.* [19] used cache inspection techniques to infer the relative popularity of a set W of websites of interest. The basic idea is that the caches of a number of DNS resolvers across the globe are probed at regular intervals, and the responses are used to compute the request rate for each website $w \in W$. Intuitively, DNS entries of websites with higher hit rates will be refreshed more quickly than those with lower hit rates, and so the targets in W can be ranked accordingly. The probe periodicity is determined to match the authoritative TTL of the targeted website w. It is also assumed that the ideal probing interval (the "useful interval") for a pair (v, w) is the period when the frequency of requests for the website w from clients using resolver v is highest.

In the analysis that follows, we simulate probing 100,000 targets, each with their own useful interval and probing periodicity. Our probing periodicities, then, are target-specific as opposed to timezone-specific ($\pi_{(v,w)} = TTL_w$), as are our useful intervals ($[s_{(v,w)}, e_{(v,w)})$). We set the number of probes required per target to be $N_j = 50$ (for all targets j). The probing limit is set conservatively at $L = 100$ probes/second.

Lastly, client DNS traffic patterns were modeled [12] using a 21-day trace of outgoing `http` requests from a university campus network [5]. For each target, we calculated the mean μ and standard deviation σ of the target's DNS traffic pattern (Figure 1). We evaluated useful interval settings of $[\mu-\sigma, \mu+\sigma)$ (the mean window length being 9h30m, containing 70% of traffic) and $[\mu-2\sigma, \mu+2\sigma)$ (mean windows of 19h, 95% of traffic).[2]

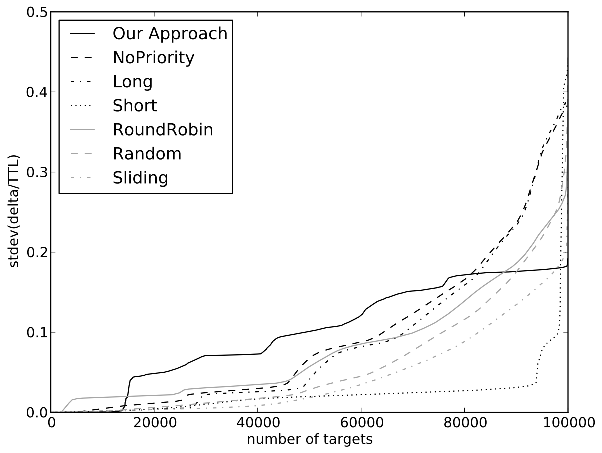
A. Comparative Schemes

To evaluate the effectiveness of our approach, we compare it with six alternative strategies for accomplishing the same goal of issuing 50 probes to each target during the target's useful interval. The strategies follow the same restrictions as we do (i.e., maximum probing rate, number of targets, periodicity, etc.), but are naïve in that they do not take useful intervals or other diurnal traffic patterns into account when scheduling their probes.

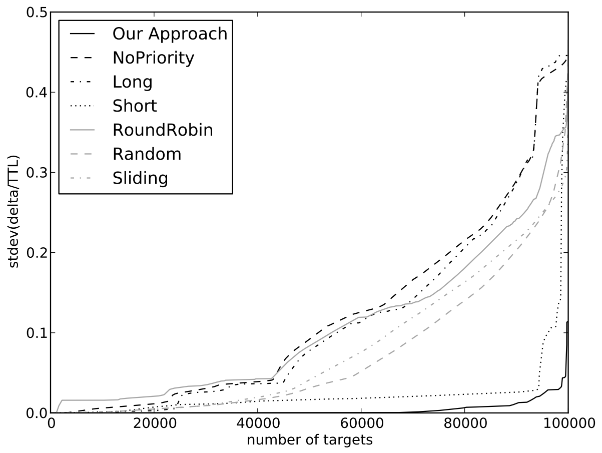
- SHORTESTPERIODSFIRST: prioritize targets with the shortest periodicities. This approach might be useful in the contexts of several measurement studies in the literature [6], [22], to collect as much measurement data as possible early in the experiment.
- LONGESTPERIODSFIRST: prioritize targets with the longest periodicities in order to complete the experiment in as few days as possible. Note that with variable probing periodicities, the overall length required to complete the experiment will be determined by the "long tail" caused by late probing of a few targets with long periodicities. Just like the above approach, shortening this long tail would also be desirable in many studies [6], [22].
- SLIDINGWINDOW: begin probing a random subset of targets on day 1, adding another subset on day 2, and so on until all targets have been added. This strategy is inspired by the approach of Keys *et al.* [13] to allow certain subsets of targets to overlap in probing.
- ROUNDROBIN: prioritize targets that have been probed least, so that each day the minimum fraction of probes sent across all targets is as high as possible.
- RANDOM: prioritize targets according to a predetermined random order.
- NOPRIORITY: use a simple first-in-first-out (FIFO) queue instead of a priority queue. Newly available probes and probes following those issued are placed at the end of the queue, without prioritization.

B. Analysis

Our results suggest that, if a probing rate limit must be observed, our approach will yield more accurate and efficient data collection than naïve probing techniques.



(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m

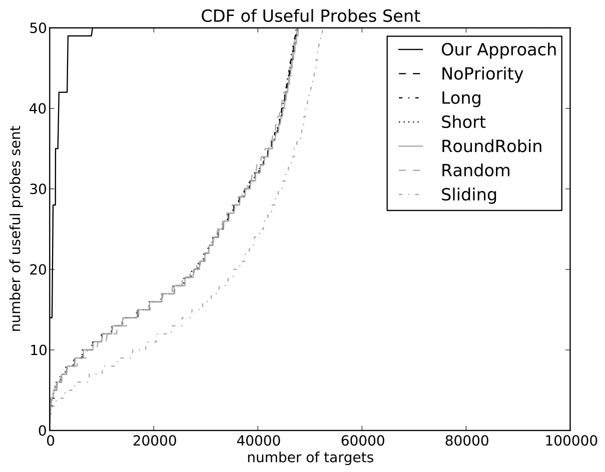


(b) $\mu \pm 2\sigma$, mean useful interval is 19h

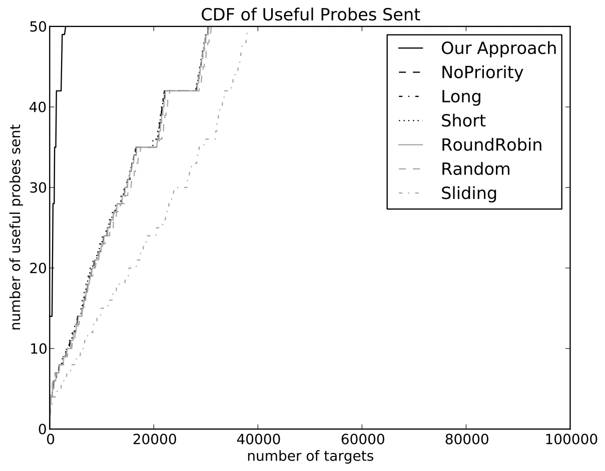
Fig. 3: CDF of the standard deviation of inter-probe delay. Many of the naïve strategies probe their targets at intervals that deviate significantly in length, which would lead to poor measurement results for several applications.

In this simulated probing experiment, probes are issued at intervals of $TTL_j + \delta$, for some small δ. Empirically, the mean of the δ values is always near 0, but we present in Figure 3 a CDF of the deviation in the δ values for each of the 100,000 targets. Figure 3 thus illustrates that our approach issues probes at more regular intervals than the naïve strategies, which will lead to more accurate data collection.

Our strategy also allows us to probe more targets—and issue more probes to those targets we do hit—than the naïve strategies. Figure 4 is a CDF of the total number of probes issued during useful intervals ("useful probes") for each of the targets, as of the end of the experiment. It shows that we are able to completely finish probing 92–97% (all but 8,228–3,070) of our targets by the end of the week, compared to about 50–70% of targets completed by the naïve strategies. Moreover, we manage to send at least 14 of 50 useful probes to all targets, while the naïve strategies send only 3–5 useful

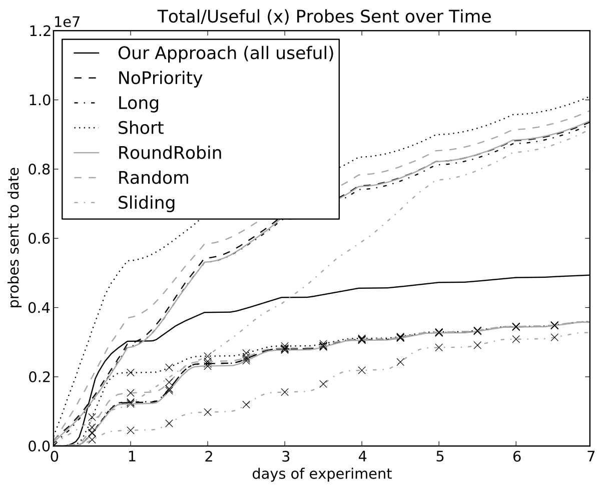


(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m

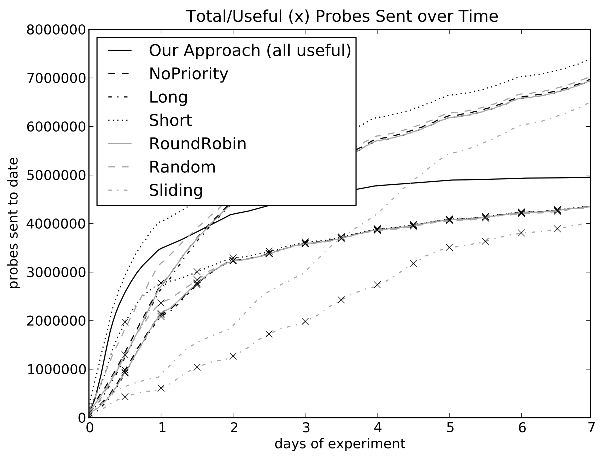


(b) $\mu \pm 2\sigma$, mean useful interval is 19h

Fig. 4: CDF of the number of useful probes sent per strategy. Depending on the useful interval length ((a) vs. (b)), we finish probing 92–97% of our targets, compared to 50–70% of targets finished by the other strategies.



(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m



(b) $\mu \pm 2\sigma$, mean useful interval is 19h

Fig. 5: Total vs useful number of probes sent over time. Total probes are indicated by unmarked lines; useful probes by criss-crossed lines. Notice that we send 4.9m probes, compared to the 7–10m probes issued by other strategies.

probes to all of the targets by the end of the week. As Figure 4b shows, lengthening the useful intervals does not significantly affect these results.

Notice as well that we are able to send more probes during useful intervals while sending fewer probes overall. Figure 5a illustrates our conservative probing behavior. Over the course of the week, we send a total of 4.9m probes, all during useful intervals—compared to the over 9.6–10m probes sent by naïve strategies, only 40% of which can be considered useful.

Two further observations may be made from Figure 3: first, with respect to the periodicity standard deviation being measured, our approach improves when the useful intervals are lengthened, while the naïve strategies perform worse. Second, the naïve strategies of SHORTESTPERIODSFIRST and SLIDINGWINDOW perform relatively well based on their inter-probe delay. For SHORTESTPERIODSFIRST, this performance is likely due to the ease with which the short periodicity jobs interleave with one another, as well as how easily they can be deferred if the rate limit is reached. However, notice that in Figure 5, the SHORTESTPERIODSFIRST strategy also issues the most probes overall. The SLIDINGWINDOW strategy limits inter-probe delay by spreading the workload evenly over the course of the experiment. That said, by naïvely staggering the targets, SLIDINGWINDOW fails to send as many useful probes to each target as the other strategies do (Figure 4).

Our advantage over the other strategies stems largely from the fact that we probe only during the useful intervals and that the intervals themselves are often not closely aligned with one another. For example, targets in various timezones will likely have useful intervals staggered throughout the day, providing a small subset of targets to probe during each hour of the day. Additionally, while we do not assume that probes in some part of a useful interval might be more useful than in other parts of the same interval (i.e., similar to the notion of "weighted tardiness scheduling" [14]), it should be clear that our approach could be augmented to work more effectively in such contexts since, by design, we have the flexibility to constrain the useful intervals as much as is necessary, and we can assign different scheduling priorities to probes depending upon where they fall within the intervals. Finally, our application of the EDF scheduling heuristic improves the precision of our periodic probing more than naïve prioritizing or a FIFO queue could do.

We believe the results show that our scheduling approach can be used to conduct active measurements on the Internet in a more responsible fashion. In particular, we argue that it offers a good solution for experimenters interested in distributing their workloads over time and across limited resources while keeping probing restrictions in mind — yet still maintaining their over-arching goal of finishing their experiments expeditiously.

V. Conclusion and Future Work

In this work we have introduced a technique for scheduling periodic network probes for Internet measurement, towards providing a means for researchers to collect measurement data while probing responsibly. By leveraging diurnal traffic data for our targets, we demonstrated that our technique could be used to collect more useful data (while sending fewer probes overall) than naïve strategies in the context of a simulated 7-day large-scale measurement experiment that observed a probing rate limit at all times.

As part of future work, we are exploring specific enhancements to our framework to accommodate subsets of targets whose probe sequences need to overlap. In doing so, we would extend our approach to be of significant practical benefit to some IP alias resolution applications [2] whose efficient linear probing complexity depends on probe sequences to targets that overlap in time. We are also interested in performing a large-scale empirical evaluation of the real-world benefits of using our scheduling strategy to coordinate probes from different measurement platforms (e.g., Archipelago [10] and FLAME [25]).

VI. Acknowledgements

We thank the anonymous reviewers for their comments and suggestions for improving the paper. This work was supported in part by the National Science Foundation under award number 0831245.

Notes

[1]Our intention is not to call attention to these particular studies, as they are by no means exceptions to the norm; they do, however, illustrate the sheer volume of probes sent in many active Internet measurements today.

[2]Note that due to limited data we estimated one useful interval for each website w, not one interval for each DNS server-website pair (v, w).

References

[1] Amotz Bar-Noy, Vladimir Dreizin, and Boaz Patt-Shamir. Efficient algorithms for periodic scheduling. *Computer Networks*, 45(2):155–173, 2004.

[2] Adam Bender, Rob Sherwood, and Neil Spring. Fixing Ally's growing pains with velocity modeling. In *Proceedings of the* 8^{th} *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 337–342, October 2008.

[3] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping Internet sensors with probe response attacks. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 13–29, 2005.

[4] Prasad Calyam, Chang-Gun Lee, Phani Kumar Arava, and Dima Krymskiy. Enhanced EDF scheduling algorithms for orchestrating network-wide active measurements. In *Proceedings of the* 26^{th} *IEEE International Real-Time Systems Symposium*, pages 123–132, 2005.

[5] Crawdad: A Community Resource for Archiving Wireless Data at Dartmouth. http://crawdad.cs.dartmouth.edu.

[6] David Dagon, Niels Provos, Christopher Lee, and Wenke Lee. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Proceedings of the* 15^{th} *Network and Distributed Systems Security Symposium*, 2008.

[7] David Dagon, Cliff Zou, and Wenke Lee. Modeling botnet propagation using time zones. In *Proceedings of the* 13^{th} *Network and Distributed Systems Security Symposium*, February 2006.

[8] M. Fraiwan and G. Manimaran. Scheduling algorithms for conducting conflict-free measurements in overlay networks. *Computer Networks*, 52(15):2819–2830, 2008.

[9] Luis Grangeia. DNS Cache Snooping, or, Snooping the Cache for Fun and Profit, February 2004.

[10] Young Hyun. The Archipelago Measurement Infrastructure. In 7^{th} *CAIDA-WIDE Workshop*, 2006.

[11] Tomas Isdal, Micheal Piatek, Arvind Krishnamurthy, and Thomas Anderson. Leveraging BitTorrent for end host measurements. In *Proceedings of the* 8^{th} *Passive and Active Measurement Conference*, pages 32–41, 2007.

[12] Jaeyeon Jung, Arthur W. Berger, and Hari Balakrishnan. Modeling TTL-based Internet caches. In *Proceedings of IEEE INFOCOM*, pages 417–426, April 2003.

[13] Ken Keys, Young Hyun, and Mathew Luckie. Internet-scale alias resolution with MIDAR. www.caida.org/tools/measurement/midar/, 2010.

[14] Joseph Y-T. Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman Hall/CRC, 2004.

[15] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20:46–61, 1973.

[16] A. Munier. The complexity of a cyclic scheduling problem with identical machines and precedence constraints. *European Journal of Operational Research*, 91(3):471–480, 1996.

[17] Christos Papadopoulos and John Heidemann. Toward best practices for active network measurement. In *Workshop on Active Internet Measurements*, 2009.

[18] Zhen Qin, Roberto Rojas-Cessa, and Nirwan Ansari. Task-execution scheduling schemes for network measurement and monitoring. *Computer Communication*, 33(2):124–135, 2010.

[19] Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos. Peeking through the cloud: DNS-based estimation and its applications. In *Proceedings of the* 6^{th} *Conference on Applied Cryptography and Network Security*, pages 21–38, 2008.

[20] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the* 6^{th} *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 41–52, October 2006.

[21] Rob Sherwood and Neil Spring. Touring the Internet in a TCP sidecar. In *Proceedings of the* 6^{th} *ACM SIGCOMM Conference on Internet Measurement*, pages 339–344, 2006.

[22] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[23] Paul Vixie. DNS complexity. *ACM Queue*, 5(3):24–29, April 2007.

[24] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. Inferring relative popularity of Internet applications by actively querying DNS caches. In *Proceedings of the* 3^{rd} *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 78–90, November 2003.

[25] A. Ziviani, A. T. A. Gomes, M. L. Kirszenblatt, and T. B. Cardozo. FLAME: Flexible lightweight active measurement environment. In *Proceedings of the* 6^{th} *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2010.

High Performance Internet Connection Filtering Through an In-Kernel Architecture

Naser Ezzati Jivan, Alireza Shameli Sendi, Michel Dagenais
Computer and Software Engineering
École Polytechnique de Montréal
Montreal, Canada
e-mail: {n.ezzati, alireza.shameli-sendi, michel.dagenais}@polymtl.ca

Naser Nematbakhsh
Computer Engineering
University of Isfahan
Isfahan, Iran
e-mail: nemat@eng.ui.ac.ir

***Abstract*—A firewall is a tool that protects users and applications from unauthorized accesses and network attacks, and secures network connections and resources. It rejects unauthorized access while permitting authorized connections based upon network security rules and policies. Although the importance of a firewall in securing a network is vital, a poor architecture and inefficient mechanism for inspecting network traffic may lead to reduced network performance. Therefore, the performance of a firewall is considered as one of its main characteristics. Several methods have been proposed to increase firewall performance. In this paper, an in-kernel architecture has been proposed. It changes the structure of application proxies and moves a portion of their functionalities to the operating system kernel level. This kernel proxy inspects and filters the connections passing through the firewall with the help of a user daemon. Tests under different loads show that the performance of the firewall increases with the proposed architecture. The main reasons are the reduction of context switches and elimination of extra copies between kernel and user space. The Kernel proxy supports the HTTP, FTP and TELNET protocols although a better performance could be reached using a kernel URL filter.**

Keywords-firewall; proxy; content filter; kernel proxy; performance.

I. INTRODUCTION

Connecting a local private network to the global public networks facilitates the communication between internal staff and outside clients and suppliers. However, some remote users may attempt to gain access to computers on the local private network for purposes such as stealing or destroying valuable company information, vandalism or even extortion.

A simple solution for local network security is to run a firewall, protecting it from unauthorized Internet accesses. However, the associated level of protection obtained depends on the chosen architecture and type of the firewall. Firewalls can be divided into two general types: packet filters and proxies.

Packet filter firewalls filter packets based on examining the source and destination addresses and ports. They examine the packet's IP (and/or TCP) headers and accept or reject the packet according to the firewall filtering policy. In this organization, packets are inspected at the network layer and then sent to the destination.

Proxy firewalls are protocol-aware firewalls. They use the packet application layer content in order to decide to accept or reject the packet, providing a more precise control [6].

A Proxy firewall may have several application proxies. In order to protect the local network, connections are usually made indirectly through these proxies. They play the role of a mediator to transfer data between the external and internal networks [6].

In an application proxy, each packet must come up to the application layer, be analyzed on that layer and ultimately it is rejected or sent to the destination. There is a risk for the application proxy to slow down the transfer of data and cause a bottleneck in the network. Therefore, the design of an application proxy should be such that it is highly efficient and has a reduced impact on the speed of data transfer between the local and global networks. The purpose of this paper is to discuss a novel method for increasing the efficiency of application proxies by moving some parts of the functionalities to the kernel level. The idea is based on splitting up the firewall responsibilities between the user and kernel levels. The kernel level tasks include some of the standard proxy tasks - authentication, rule management, and state management. The application level tasks include rule-based, detailed data and packet level analysis. The main benefits of the revised firewall architecture come from savings in the number of context switches needed to process each packet sent and a reduction in the number of copies between user and kernel space.

The paper continues in the next section with a discussion of related work for increasing the firewall performance. This is followed by a section describing the architecture of in-kernel proxies and differences with the Linux Netfilter and other application layer proxies.

Subsequently, we will discuss our method in detail and will show how it can increase the performance of a firewall. Finally, the paper concludes by identifying the main features of the method and possibilities for future work.

II. RELATED WORK

Different methods have been introduced for improving the performance of firewalls. The first and oldest method to increase the efficiency of a firewall is called state-full filtering [3]. In a simple stateless firewall, the inspection of a packet is performed separately from other packets. Thus, whenever a packet comes to the firewall, regardless of whether it belongs to a previously setup TCP connection or not, or the state of that connection or other packets, the firewall will analyze the packet according to the rules, and reach a decision for it. The analysis and investigation of all the packets causes a severe performance cost to the firewall. Therefore, keeping track of the state of the connections can speed up the analysis of the packets and thus increase its efficiency. In this way, only the first packet of a connection is verified against the firewall rules, and little verification is required for the remaining packets [3].

Another method to increase the efficiency of a firewall is the use of caching in the application layer. With this mechanism, the results of recent user requests are saved. In the case of a repeated request (e.g. Web page or DNS entry), instead of creating a new connection to the server, the saved pages in cache memory are used. This will decrease the response time and use of the network bandwidth. This mechanism has been used in the Squid application proxy [5].

Fall [13] introduced another method to speed up the copying between two sockets and two files. Hence, this method can increase the data transfer speed and efficiency, in comparison with application proxies which only copy and transfer data.

In 1998, at the IBM research center, Bhaqwat [14] introduced another method called connection link. This method is based on the separation of control and pass-through of the proxies and proposes a fast route to transfer the data in pass-through mode. The main idea of connection link is that one should determine when a proxy moves from control to pass-through and then link the two separate TCP connections as a single connection [14]. Studies [1], [2], [4] and [7] suggest different kernel methods to increase the performance and efficiency. Gopinath [9] discusses kernel support for firewalls. Knobbe et al. [6] propose high performance architecture for network firewalls. Most of the aforementioned methods are based on separating the control and pass-through modes which is not cost effective and can't be accomplished completely. This paper introduces a method that increases the performance of firewalls to an acceptable level, without having to isolate the control and pass-through modes.

III. IN-KERNEL PROXY

When two clients of a network are connected to each other through a proxy, this proxy mediates the connection and controls data transfers between these two clients. The proxy checks the authorization of these clients and decides whether these two clients can be connected to each other or not, and if the connection is permitted, the data being transferred between them is controlled by the proxy [10].

Generally, proxies act in two following modes [7]:

- Control mode
- Pass-through mode

In control mode, the proxy performs some analysis on the data before it is accepted and transferred. When the control phase is over, the proxy moves to the pass-through mode in which, the proxy only passes the data. After the data is transferred, the proxy may turn back to the control state. For example, a TELNET proxy starts in control state considers and analyses a TELNET request and whether it is permitted or not. When it is permitted and the connection is made, the proxy changes to pass-through mode and copies the data between the two ends of the connection. Distinguishing these two modes and moving from one mode to another is the most important task in the improvement of the proxies' performance [4].

The functionalities of various proxies are different in the control mode. These vary from simple control, at the connection starting time, to continuous analysis of data being transferred during the connection. The proxies can be categorized into four groups according to the amount of processing that they do.

The first group of proxies performs a very little control. They are in control mode only at the time of connection start, and then remain in pass-through mode until the end of the connection. An FTP proxy is an example of this type. In the FTP protocol there are two kinds of connection, control connection and data connection. The FTP proxy processes an FTP request in control mode and connects the two clients through a new data connection. Then, the proxy processes the data connection, in pass-through mode. It keeps this mode until the end of the connection. The control connection stays in control mode in order to process the subsequent requests and commands [3].

The second group of proxies performs a large amount of control. These proxies authenticate the user, and the connections remain in control mode for all the data being transferred in both directions. One example of these proxies is the HTTP proxy that lets the user have access to the HTTP servers on the Internet. HTTP proxies can refine and modify the access of external agents or users by filtering and constraining permitted agents or users, and also it can modify and refine the response to internal requests through deleting unsecured applets (as JAVA applets and other potentially malicious codes) [7].

A firewall may combine different proxies, each of which controls a specific aspect of the transfer or exchange of data between two networks or clients. Typically, a proxy receives a connection, authenticates the client or user, and possibly after having refined and modified the request, passes the data to another network or client. The firewall either uses its own IP address to be an intermediary between the two ends of the connection, in which case it is called classical proxy, or it is hidden from both ends of the connection, in which case it is called a transparent proxy.

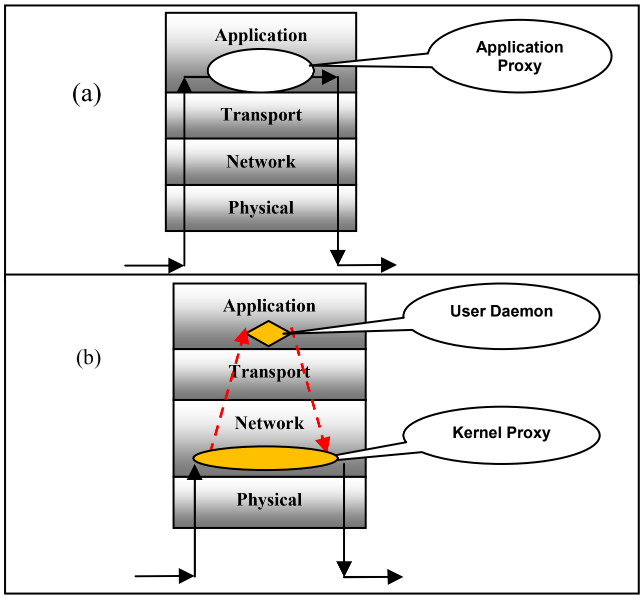


Figure 1. Path of Packets: (a) in a normal proxy (b) in the proposed in-kernel proxy.

An important point is that in many applications only a small part of packet (usually in the header) is controlled, and the proxy is mostly in pass-through mode. However, here a packet should also move from the kernel layer to the application layer and return to the kernel layer again before being transmitted to the destination. Therefore, the transfer of a considerable percentage of data from kernel layer to the application layer seems inevitable but wasteful [8] (Figure 1 (a)).

However, it is possible to design a new in-kernel structure for proxies that can solve the problem of unnecessary copying of packets from kernel layer to application layer, decreasing the number of context switches, and potentially greatly improving the efficiency. In this method, the proxies will have the responsibility of controlling the packets of the connection in the kernel network layer, and will avoid the transfer of data to the application layer. Thus, many cases of copying between kernel layer and application layer will be avoided and the transfer and efficiency of the firewall will be increased. Figure 1 (b) depicts this organization.

Of course, completely moving the proxy functionalities to the kernel may complexify the proxy structure. As a consequence, this could decrease of efficiency of the firewall. A solution is only moving the necessary and convenient parts of proxies to the kernel, keeping the remaining part in the application layer. This user-level part is shown in figure 1(b) as user daemon.

IV. . DESIGN OF THE IN-KERNEL PROXY

The in-kernel proxy is composed of two parts: one central part in the IP layer and one part in the application layer. The location of the in-kernel proxy in relation to the packet filter (Linux Netfilter) and IP layer is shown in figure 2.

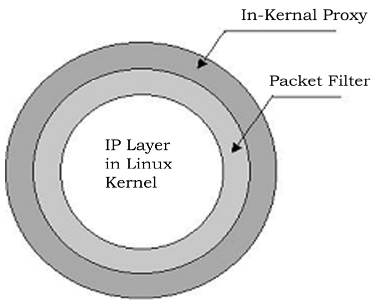


Figure 2. The location of the in-kernel proxy

As shown, whenever the network layer receives a packet, it is first analyzed in the packet filter of the firewall and then given to the in-kernel proxy.

Similar to the Linux Netfilter, the in-kernel proxy implementation is divided into two sections, a kernel module and a user daemon that interfaces with users and creates the rules and interactions.

However, the in-kernel proxy is a content filter which means that it has knowledge about the higher level protocols and inspects the packet based on its data part as well as its TCP/IP headers, while Netfilter is a packet filter and can just filter the network packets based on the TCP/IP header of a packet and not the data content [15].

Unlike the proxies in the application layer, where there is a specific proxy for each protocol, in the in-kernel proxy, a common proxy is set for all protocols. Of course it will have to distinguish the protocols in some parts of this proxy. However, all proxies are designed within this single framework. The proposed in-kernel proxy supports the HTTP, FTP, and TELNET protocols. However, due to memory and resource limitations of the kernel, the main usage of the in-kernel proxy is for URL filtering. This is an important difference between a complete and comprehensive proxy like SQUID and this in-kernel proxy.

With respect to the responsibilities of the proxies, and also the characteristics of the operating system kernel, the kernel-based proxy is composed of the following modules:

- Authentication
- Rule Manager
- Connection Manager
- Connection Filter
- Log manager

Each module has specific responsibilities and duties, being described in detail in the following sections. The relations among these modules are shown in Figure 3.

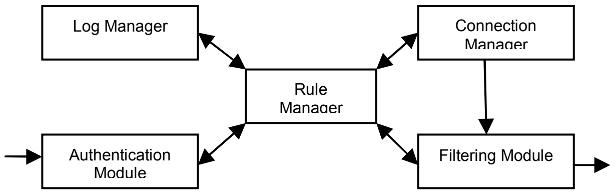


Figure 3. Architecture Elements

A. Authentication Module

In the kernel-based proxy, the authentication of the client is used to authenticate the users and clients. In this module, the client or user is identified once in order to check its permission to use the proxy and then all the connections and delivered packets are permitted for a predefined time duration. For authentication, users must connect to a specific port in the firewall and authenticate themselves by entering their username and password.

B. Rule manager module

This module is responsible for management and maintenance of inspection rules. The rule manager is one of the most important and active parts of the kernel proxy. This module receives the rules from a user interface, stores them to become available to other parts such the authentication, log manager, connection filter module. This module is also responsible for receiving upgrades of these rules and keeping the consistency. The rule manager is thus somewhat related to all the modules as a database in which all modules get their operation instructions and orders.

In the kernel proxy, the rule manager is composed of two parts: application layer and kernel layer. This module receives the rules from the system administrator through a user interface. Administrator writes the rules in synthetic forms, and the user interface, having received these rules, parses them to a canonical form saved in the data structures and files. The application layer part of the rule manager checks these rules and after deleting possible errors, copies the rules to the kernel layer.

C. Connection Manager Module

The responsibility of this module is to identify the protocols, classify the packets in the kernel, and then create and manage the table of different states of the proxies. In other words, this module identifies the protocols type of the packets, and accordingly reads the content of the packet and adds it as a new entry in the states table. For example, for HTTP packets, this module extracts the HTTP request and response contents and puts them in the proxy state table. All the proxies use the same state table. The data stored in the state table are as follows:

- Requested URL or address
- Client and Server address and port
- Protocol type (HTTP, FTP, TELNET)
- Connection state (connected, known ID, unknown ID, filtered, waiting, …)
- The list of the rules taken before for the connection packets
- Connection timeout value
- Bytes sent and received
- The policy applicable to the packets
- Display of the whole saved packets

D. Filtering Module

The duty of this module is to filter the data, if required- by different filtering rules for the packets passing through. Its responsibilities are as follows:

- Receiving the packets from the connection manager module: in the previous section it is mentioned that the connection manager module reads the packets, determines their protocol and then classifies them in accordingly. The filter module receives its own input data and packets from the connection and consults with the manager module to perform the required type of filtering.
- Inspecting the connection packets through the rule manager module: this module having received the data of the connection, according to the required filter, sends a request to the rule manager module concerning the characteristics of the connection and the related data. The rule manager having analyzed the request responds to the filter based on its database of rules. Moreover, the rule manager extracts the type of logging needed for that request and performs logging.
- Making decision about the packets: the filtering module decides according to the response from the rule manager to reject or accept the packet and to continue the processing. If the response is to wait, the filter saves the packet until the response from the rule manager is available. The rule manager may also consult with the daemon in the application layer for the decision.

Three types of filters to be used by the filtering module:

1) Command filter: this filter is used for HTTP and FTP. Here, the commands are received from the packets in the connection, and compared to the rules inside the kernel. The commands in the FTP proxy are “get”, “put”, “dir”, and “pass”, while in the HTTP proxy they are “get”, “head”, “post”, “connect” and etc.

2) URL filter: the filter is used for the HTTP and FTP protocols. To use this filter, the extracted URL from a connection is comparedwith rules inside the kernel. The database for inspecting the URLs is located in the user level agent but there is a small URL database cache in kernel which helps the kernel proxy to filter the most used URLs. However, for the URLs out of this cache, the kernel asks the user level agent to get the correct decision.

3) Content filter: considering the importance of the transferred files and data, it is possible that the rule manager force the proxy inside the kernel to analyze the content of some connections. Therefore, the files and data extracted from the packets are filtered and refined by the rule manager through the use of the content filter placed inside the kernel (and sometimes by connecting to the more elaborate content filter in the application layer).

E. The logging module

This module reports important events to the system log. There are three log methods (summarized, full log and no logging) from which the administrator can select the default method:

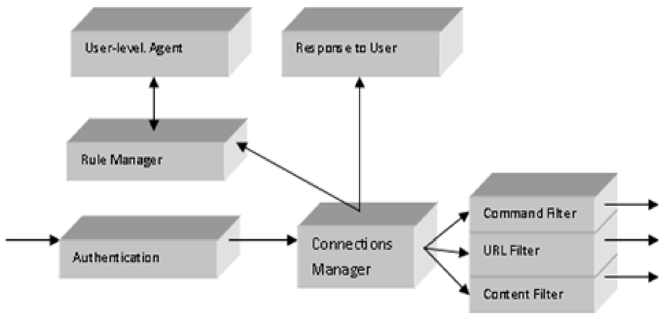


Figure 4. Execuition scenario of a HTTP kernel proxy based on the proposed architecture.

F. *Architecture of the In-kernel proxy*

In this part, the execution scenario of the HTTP kernel-based proxy is analyzed and reviewed in terms of proposed architecture. In general, the practical scenario of the HTTP proxy can be as follows:

- Analyzing user's authentication status: this is the first step in inspecting the connection. If the user has been authorized already, then the remaining packets are accepted. Otherwise, an "Unauthorized Access Message" will be sent to the user and the connection will be closed.
- Receiving the user's request and URL from the packet.
- Writing the new connection in the state table.
- Connecting and consulting with the rule manager and filtering the command and/or the URL and/or the content.
- Recording correspond events in the log.
- If the connection is valid based on the filtering rules, then a connection will be made with the distant host.
- Otherwise, a suitable response will be sent to the client and the connection will be terminated.

Figure 4 shows this process.

V. EVALUATION

The proposed kernel proxy architecture has been implemented in Linux Red Hat version 7.2 with kernel version 2.4.1. The network, hardware and software configuration of the evaluation setup are first presented. Then the influence of different parameters on the efficiency of the system will be analyzed. Finally, the results of the evaluation for the proposed system will be compared to the efficiency of an existing user-level proxy implementation.

A. *Configuration*

Table 1 show the configuration of the networks, clients, server and firewall machine (FwTest). In this configuration, two local networks with 100 Mbps bandwidth have been used.

TABLE I. THE CONFIGURATION USED FOR TESTING THE KERNEL PROXY

Host	CPU	RAM	HDD	NIC
Client	Intel PIII 800 MHz	256	Quantom 60G	1*3c905c 10/100
Server	Intel PIV 1000 MHz	256	Quantom 20G	1 *3c905c 10/100
FWTest	Intel PIII 800 MHz	128	Quantom 15G	2 *3c905c 10/100

In this configuration, proxies are transparent, the log method is summarized log and NAT (network address translation) is disabled. The number of rules in the kernel is 30 rules and the tests exercise the HTTP proxy. No cache mechanism is used in clients, server, or firewall.

Fire Bench [12] is used to generate network traffic and monitor the firewall performance. It measures the connections per second and the average response time:

- Connections per second: This test counts the number of the connections per second that are supported by the firewall. The clients make a connection to the server and immediately terminate it and start with a new connection. The number of clients increases gradually up to the point where a maximum is reached. SPECWeb2009 [11] is used to generate the needed connections and loads. Figure 5 depicts the comparison based on this test.
- The average response time test: This test is another criterion to evaluate the firewall performance; it is based on the average time needed to respond to each connection. Figure 6 depicts the comparison based on this test.

The evaluation is started by the execution of a load managing application, the manager, in the first client. This application reads the configuration file and produces the workload file. Then, it creates a TCP/IP socket to other clients and sends the workload and configuration files. At this point, each client waits for a message from the manager. When clients get the START message from the manager, they start to send their request to the server (through the firewall). After finishing the test, the manager gets the test results from the clients. This process runs 3 times and subsequently, the manager calculates the results and generates the final report.

B. *Analysis of the results*

Each normally terminated TCP connection is composed of at least seven packets [1]. In the kernel proxy, the first packet and also the packets that contain application layer content are analyzed and inspected. If they are sent to the application layer daemon, they need re-analysis by the user level rule manager. Therefore, the number of communications with the daemon and the number of rules, have a direct effect on the firewall performance.

The FireWall ToolKit (FWTK) from TIS is a popular user level application proxy [16]. The comparison between the in-kernel proxy and FWTK shows that the transfer of the firewall input traffic to the application layer causes a severe decline in the efficiency of the firewall (the performance drop in the last part of the yellow line in figure 5).

In the case of the kernel HTTP proxy, only the packet that contains the URL is sent to the application layer daemon, while the remaining packets will pass through the kernel and be sent directly to the destination. Therefore the process is much simpler for many TCP control packets. FWTK on the other hand sends all the input data traffic to the application layer, thereby decreasing the efficiency.

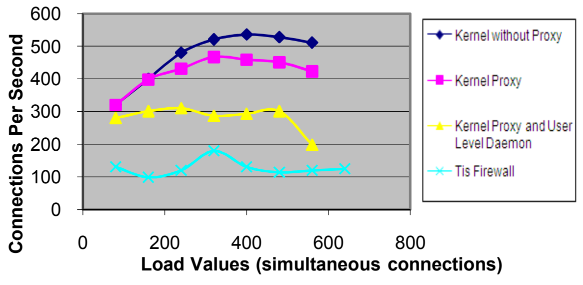


Figure 5. Comparison of the results (Connections per Second)

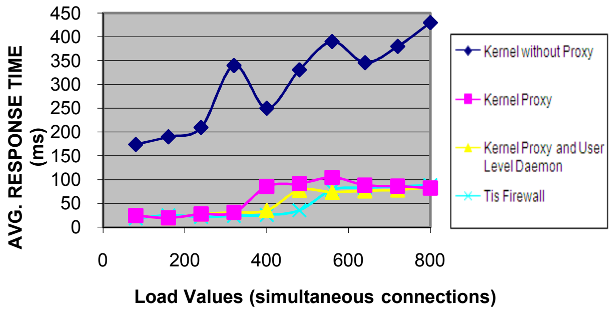


Figure 6. Comparison of the results (Average Response Time)

In the in-kernel proxy, a high efficiency is reached because of the pass-through of most TCP control packets.

Of course, the number of rules in the in-kernel proxy also affects the efficiency of the firewall directly. However, the influence of the number of rules is much less than that of the extra copies between kernel and user-level and associated the context switches. Figure 6 compares the average response time of kernel proxies (with/without user level daemon) and FWTK application proxy.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, the architecture and implementation model of an in-kernel proxy to increase the efficiency of the firewalls is presented. Results of the evaluation show that a proper division of labor between the application and kernel levels could yield substantial savings in terms of reduction of OS system calls and copied data. The efficiency increase compared to a kernel without any proxy and also compared to the efficiency of FWTK application layer proxy is shown. The main reason for the increase in efficiency of the proxy is due to the decreased number of contexts switches between the kernel and application layer and also the reduction of unnecessary copies among the different layers.

A complete URL filter proxy requires an application layer URL categorizer. However, in this project, a basic URL categorizer has been implemented; the implementation of a kernel URL categorizer could be a future extension.

The most important future enhancement will be implementing of a high performance kernel level packet modifier which inspects and modifies packets content, removing potentially malicious content (e.g. viruses and other malwares) from the packets. The important issues here are controlling the packets sequence number and the TCP sliding windows. This would lead to the creation of a light packet content filtering TCP daemon in the IP layer.

Finally, the kernel proxy should provide an application-layer interface to aid users and administrators in the configuration, rule editing and policy management of the firewall.

VII. REFERENCES

[1] Y. Zhang, Z. M. Mao, and J. Wang, "A Firewall for Routers: Protecting against Routing Misbehavior," Proc. 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007.

[2] J. Lee, P. S. Jean, T. Rick, M. G. Jack, and B. A. Bavier, "Network Integrated Transparent TCP Accelerator," Proc. 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 10), 2010.

[3] A. Rousskov, "On Performance of Caching Proxies," North Dakota State University Fargo, 1999.

[4] R. Jain and T. J. Ott, "Design and implementation of split tcp in the linux kernel," Doctoral Dissertation, 2007.

[5] S. Sahu, "Design Considerations for Integrated Proxy Servers," Department of Computer Science, University of Massachusetts, 1999

[6] R. Knobbe, A. Purtell, and S. Schwab, "Advanced Security Proxies: an Architecture and Implementation for High Performance Network Firewalls," Proc. DARPA Information Survivability Conference and Exposition 2000

[7] S. K. Adhya and S. Ganguly, "Asymmetric TCP Splice: A Kernel Mechanism to Increase the Flexibility of TCP Splice," master thesis, Department of Computer Science & Engineering Indian Institute of Technology, April 2001.

[8] S. E. Schechte and J. Sutaria, "A Study of the Effects of Context Switching and Caching on HTTP Server Performance," Available: http://www.eecs.harvard.edu/stuart/Tarantula/FirstPaper.html, [Oct. 29, 2010].

[9] K. N. Gopinath, "Kernel support for building network firewalls," M.S. thesis Department of Computer Science & Engineering Indian Institute of Technology, April 1997.

[10] W. Shroder, Firewall and Internet Security, Prentice Hall, 1994.

[11] Standard Performance Evaluation Corporation (SPEC): SPECweb2009 Release 1.10, Oct 2009.

[12] KeyLabs Corporation: Test Final Report, Firewall Shootout Networld+Interop, 1998.

[13] K. Fall, "A peer to peer I/O system in support of I/O intensive workloads," Ph.D. thesis, 1993, Department of Computer Science U.C.San Diego.

[14] S. D. Purkayastha, "Symmetric TCP Splice: A Kernel Mechanism For High Performance Relaying," Department of Computer Science & Engineering Indian Institute of Technology, April 2001.

[15] The Netfilter Project site, Available: http://www.netfilter.org/ [Oct. 29, 2010].

[16] TIS FWTK Firewall site, Available: http://www.fwtk.org [Oct. 29, 2010].

An Empirical Study of Web and Distributed Software Risks from Three Perspectives: Project, Process and Product

Ayad Ali Keshlaf
School of Computing Science
Newcastle University
Newcastle, UK
a.a.a.keshlaf@ncl.ac.uk

Steve Riddle
School of Computing Science
Newcastle University
Newcastle, UK
steve.riddle@ncl.ac.uk

***Abstract*—Web and Distributed software development is vulnerable to risks, which may apply to any of three perspectives: Project, Process and Product. However, existing software risk management approaches are mainly concentrating on the Project and only very few of them have touched the Process perspective. Our *WeDRisk* approach has, as one of its main objectives, coverage of Web and Distributed risks from all three perspectives. The work presented in this paper is a result of an experiment to evaluate some aspects of *WeDRisk*. This paper is mainly focused on the evaluation of a clustering of the risks from the three perspectives, and the criteria used for the clustering. The result of the experiment illustrated the importance and usefulness of clustering and considering of the risks from the three perspectives as a way of reducing the effort and time in managing the risks and then increasing the efficiency of risk management in web and distributed developments.**

Keywords- Web and Distributed risk perspectives; Software risk mangement; Clustering of risks

I. Introduction

The development of the Web and Distributed (W-D) software industry is sharply accelerating over the last five years. This high rate of growth is due to the incremental demand on software applications in all today's activities and technologies as well as the ubiquity of the Internet, which has increased the deployment and development over it [1]. However, with these developments come new problems: a higher management complexity, new challenges and risks such as: Insufficient competence; Wage and cost inflation; Inadequate informal communications; Lack of trust; Culture differences (e.g., different language, different corporate culture and different developers' background); Time-zone difference (leading to ineffective synchronous communication); Development process differences; Knowledge management challenges (most of the existing management approaches are designed for co-located teams); Security issues (Ensuring electronic transmissions confidentiality and privacy) [2][3][4].

The above-mentioned challenges and risks attack all perspectives (project, process and product, hereafter called "3P") of W-D software industry. However, there are diverse definitions for the 3P perspectives and it is difficult to find a clear and unique definition for any of them. The following definitions are used for the purpose of this study [5][6]:

Project perspective concerns aspects such as budgets, plans, goals, responsibilities and schedules.

Process perspective concerns the methods, tasks and activities of producing the software.

Product perspective concerns the final product aspects such as its functionality, maintenance, market competence and security.

Looking at these perspectives it is expected that each one of them includes, or could be affected by, different types of risks. Risk management is, therefore, an important issue from these three perspectives [5][6].

The paper gives a background on related work (Section II), problem and approach (Section III), and then it describes the experiment (Section IV), presenting and analyzing the result of the experiment (Section V). The paper discusses the experiment results in Section VI and then presents the conclusions and suggested future work in Section VII.

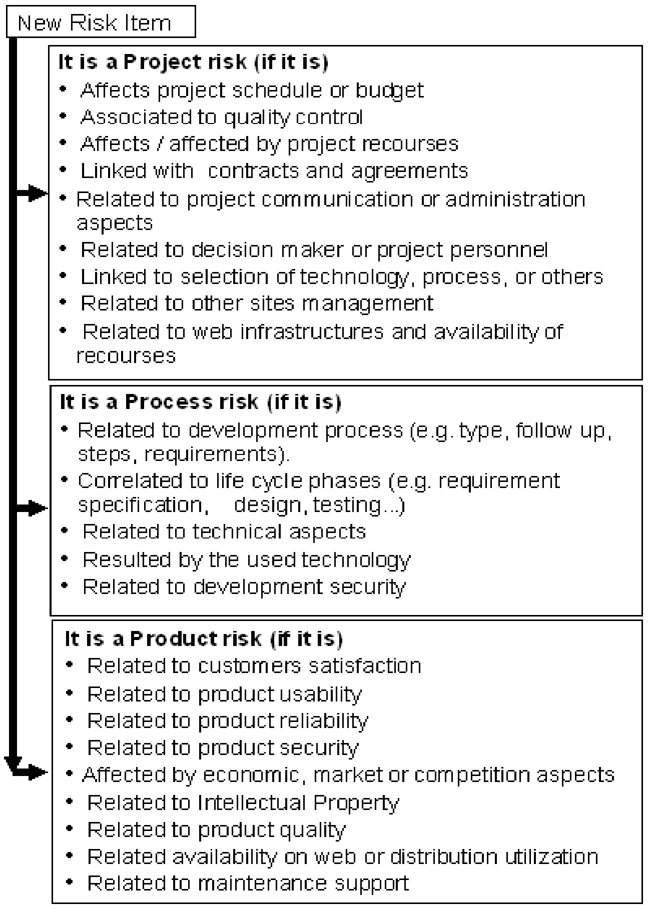
II. Related Work

Many software risk management approaches exist such as WinWin Spiral model [5]; GRisk-Mode and tool [7]; GSRM model [8]; Riskit method [9]; GLM Model[10]; GDPS RM Framework [11]. However, the software risk management issue has got only scant attentions in distributed development [1]. A review of software risk management for selection of best tools and techniques, which has been concentrated on recommended approaches (*SEI*, *SER*, *SoftRisk*, *TRM*, *ARMOR*, *Riskit*) has concluded that no one tool or technique alone can be considered as a perfect for managing risks in software development [12]. Gorski and Miler in [13] have introduced a concept (*DS-RM-Concept*) and a tool called Risk-Guide for risk management in distributed software development projects, with emphasis on the role of open communication. Kuni and Bhushan [14] introduced the Wipro Offshore Outsourcing Methodology (WOOW), which takes the risks in the account through a model called Risk Management Model.

Keshlaf and Riddle [3] reviewed the existing approaches and highlighted a number of weaknesses in them, especially in managing W-D development risks. One of the weaknesses is "*... the existing approaches concentrate on project perspective of software development and they do not pay enough attention to other perspectives (Process and Product)*".

III. Problem and Approach

In order to tackle the weaknesses of the existing approaches in managing W-D development risks we introduced a new approach called *WeDRisk* [15]. It consists of five layers (Project, Stakeholder, Risk Management Customization, Implementation, and Evaluation & Auditing) and two supporter components (Communication & Plug-In Controller and Evolution Regulator). The layers consist of components, which contain steps, techniques and guidelines [15]. *WeDRisk* maps risks dependencies during the risk management operation in order to reduce undesired consequences.


New Risk Item

It is a Project risk (if it is)
- Affects project schedule or budget
- Associated to quality control
- Affects / affected by project recourses
- Linked with contracts and agreements
- Related to project communication or administration aspects
- Related to decision maker or project personnel
- Linked to selection of technology, process, or others
- Related to other sites management
- Related to web infrastructures and availability of recourses

It is a Process risk (if it is)
- Related to development process (e.g. type, follow up, steps, requirements).
- Correlated to life cycle phases (e.g. requirement specification, design, testing...)
- Related to technical aspects
- Resulted by the used technology
- Related to development security

It is a Product risk (if it is)
- Related to customers satisfaction
- Related to product usability
- Related to product reliability
- Related to product security
- Affected by economic, market or competition aspects
- Related to Intellectual Property
- Related to product quality
- Related availability on web or distribution utilization
- Related to maintenance support

Figure 1: Perspectives Clustering Criteria

WeDRisk includes several concepts, which could help in tackling some of the identified weaknesses. One of these concepts is the consideration of the risks from the 3P perspectives. This concept depends on a clustering strategy (using special criteria factors) to deal with the risk from these three perspectives.

The clustering strategy is intended to save time and effort. It locates fewer resources at each perspective as the managing of risks will focus on the relevant perspective risks each time. *WeDRisk* suggests some factors that could help for clustering the risks from the 3P perspectives. These factors are shown in Figure 1.

The suggested criteria groups the risks based on some characteristics and nature of the perspectives. The proposed criteria are identified based on experience, available literature and previous research results. Following sections describe controlled experiment, which was used to evaluate the clustering strategy.

IV. The Experiment

This experiment is a part of PhD research at Newcastle University, UK which aims to build a software risk management approach to manage W-D development risks. The approach is called *WeDRisk* and it is currently under development. The aim of this experiment is to test some hypotheses, which are prepared in order to validate the significance of a list of proposed W-D risks and the usefulness of clustering them from the 3P perspectives (project, process and product). The experiment is also used to examine W-D vulnerability to atypical risks and the usefulness of absorbing their side effects.

The experiment has been designed to test four hypotheses (H1-H4). H1 evaluates the importance of potential risks to W-D development. H2, H3 evaluate the consideration of the 3P perspectives and H4 evaluates the atypical risks absorbing strategy. This paper focuses on H2 and H3, which validate the clustering strategy (and our proposed clustering criteria) as a way of considering the risks from the 3P perspectives. The two hypotheses are:

<u>Hypothesis H2:</u> *"If the developers use the proposed clustering criteria then the clustering time of W-D risks from three perspectives will be shorter and the effort will be saved"*
<u>Hypothesis H3:</u> "*Clustering the risks from three perspectives (project, process and product) saves time and effort"*

The choice of the controlled experiment to test these hypotheses was due to the following reasons:
- Difficulties of getting a suitable case study, as many software developers have high confidentiality restrictions for their projects data.
- Unavailability of suitable subjects who have the required experience or education.
- Emulating the real working environment conditions, which are significant to the study.
- It was difficult to ensure the same working environment for all the subjects (to avoid the difference in resources such as internet and computer

speeds, as time is one of the measurements as well as to observe the real effort).

- To avoid any outside influence on the participant, which could be different from one participant to another, as this could lead to some bias in the experiment.
- In such experiment the participants need some clarifications from time to time, which should be provided equally to all participants to avoid any bias.
- It is difficult to get all the participants at the same time in the same room and it is difficult to provide the same support and clarification to a group of subjects who are doing the same tasks at the same time. If so there will be some biases.
- To ensure the exact implementation sequence of the tasks during all experiment stages.

A. *Experiment Method*

Our experiment design was highly inspired by works in [16][17][18], especially in the way of structuring the experiment, preparing the hypotheses, avoiding bias, collecting, analyzing the data, discussing the result and describing the experiment. Before conducting the experiment we discussed the experiment design with expertise from Carnegie Mellon University and other researches in Newcastle University, who provided us with valuable comments. Based on the provided comments we made some modification to improve the experiment. The modifications included changing the method of recording the time during the experiment and giving more freedom to subjects, in order to reduce the time pressure on them. One other modification related to the arrangement and sequence of handling the experiment material. This also helped in estimating the required time for each participant to perform the experiment: we found 30-35 minutes suitable. We made the required improvement on the experiment material and measurement then we started the real experiment.

In order to give a chance for more replication of this experiment we provide hereafter full details of the experiment so it can be replicated easily for any research reasons.

B. *Subjects (Participants)*

We recruited 30 participants (male and female) for this experiment. They were PhD students, researchers and MSc students at School of Computing Science, Newcastle University-UK. The majority of them were PhD students or researchers. All of them either have experience with software development or at least participated in software projects in their studies. The subjects were recruited by emails. We sent email to all MSc students, PhD students and researchers at the school and we got a positive response from about 35 of them, but we have chosen only 30 subjects based on specific experience and education criteria.

This set of participants has been selected as we expected that they had enough knowledge or experience with software development and many of them had participated in software development projects as part of their courses. We compensated the participants who performed the experiment with £10 Amazon vouchers for their time. Instead of using the participants' real names or numbers we assigned a special reference number so that it can be used anonymously for future research after this experiment.

C. *Introductory*

At the beginning of the experiment each participant was asked to fill in and sign a consent form. Then the participants were briefed with the necessary information (e.g., description of the experiment, software risks and proposed list of risks, W-D development, software risk management, software perspectives). After that their assigned tasks in the experiment were explained to them. Each subject was told by the experimenter that he has the right to stop at any time if he feels not happy to continue for any reason and he has to try to be accurate as possible. Participants were told that they have the right to ask any question related to the experiment at any time if need be.

Printed versions of all experiment related information, tasks and instructions were supplied to support the participants' understandings.

In some stages of the experiment the participants were randomly divided into two groups (control and experimental group) based on the nature of the task and needed measurements.

D. *Apparatus & Instrumentation*

The apparatus, which were used in the experiment include computer for data entry and office environment, normal stationery, hard copies of the experiment material and forms and sport watch (on a mobile).

E. *Subjects (Participants) Tasks*

The participants' tasks can be summarized as follows:

- Understanding their roles in the experiment.
- Performing the assigned roles.
- Clustering the W-D risks from three perspectives (project, process and product). For this task the participants are divided into groups (control and experimental groups): Control group members perform the clustering operation based on their own knowledge whereas, the experimental group use a specific criteria for clustering the risk from the three perspectives.
- Searching twice for certain perspectives risks before and after the clustering

All the participants were told that they had the right to ask for any clarifications during the experiment and they could stop at any stage of the experiment

F. *Avoiding Bias*

Experiments are very sensitive to errors. Many errors could arise due to bias in the experiment. In order to avoid bias the required information and instructions were provided to all of the participants as hard copies. However, the criteria factors were provided to experimental group members only as it is used by them only. Moreover, the dividing of the participants into control and experimental groups was on a random basis. This is also to avoid contradiction in the experiment result.

On all the data documents we used only the participants' reference numbers rather than the names or numbers. This anonymity makes the analysis of the data more reliable and saves the privacy of the participants.

Bias is also avoided at the result analysis. This is achieved by sending the gathered data to a third party to help us with the analysis without giving him any information about the subjects.

G. *Data Confidentiality*

The collected data are strictly confidential to the experimenter and his supervisor. It is only used for research purposes and not for other intention. The participants bibliographic data (e.g., subjects name and number) were only used for providing the free Amazon vouchers (through the school administration) and were discarded afterwards.

V. RESULT AND ANALYSIS

The collected data from the experiment were in the form of tables and question answers, to test the research hypothesis. Several tasks were designed to test each hypothesis. For this reason, we introduce the results and analysis of the data arranged in order of the hypotheses. In this section each hypothesis is stated and followed by the related result and analysis.

A. *Hypothesis H2*

In order to test Hypothesis H2 we have divided the subjects into two "control" and "experimental" groups (15 in each). In order to avoid bias and contradictions we used a randomization strategy. The randomization in this case was done on the subjects, when they were divided into the two groups (control and experimental).

The subjects of both control and experimental groups have been asked to cluster the risks in Table I from the 3P perspectives. The control group completed their task without giving them any clustering criteria; whereas the experimental group has been given clustering criteria.

There is a significant difference between the time taken by the two groups shown by the Mann-Whitney U statistical test at (**p-value = 0.0079, U = 168.0**).

Generally the total time used by control group participants for the clustering (**56 minutes**) of the risks was less than the time that was used by the experimental group (**108 minutes**). This could be for reasons such as:

- Actual time for reading the criteria, poor design of the criteria design or it is hard to understand the criteria.
- The speed of answering from the control group could be because there is no restriction on their clustering. This raises questions about the accuracy of their clustering.
- Without the criteria the selections and answers could be different from one cycle to another and from one developer to another. This also applies to the required time for the clustering.

In fact using the criteria first time may take some time from the developer for understanding and reading but it is expected that the next cycles will take a shorter time. More training and improvement on the criteria will make it much easier to understand and use the criteria.

TABLE I. NON CLUSTERED POTENTIAL W-D RISKS

Risk No.	Risk Name
1	Unfamiliarity with international and foreign contract law
2	Inadequate customer requirement (see and change strategy)
3	Poor documentation
4	Low visibility of project process
5	Inadequate process development
6	Not enough measurement and estimations
7	Lack of security precautions
8	Weaknesses in protection procedures for Intellectual Property rights
9	Vendor feasibility
10	Insufficient competence
11	Communication failures
12	Poor sites management control
13	Failure to manage user expectations
14	Insufficient project stakeholder involvement
15	Process instability
16	Poor performance
17	Poor UI
18	Insecure of communication channels
19	Lack of requirement specification
20	Inadequate user involvement
21	Difficulties in ongoing support and maintenance
22	Unrealistic estimation of the number of users
23	Differences in the development methodologies and processes
24	Weak or inadequate contracts
25	Complicated development dependencies between project sites
26	A Cross cultural differences / influence
27	Poor product functionality
28	Market fluctuations
29	Scalability limitations
30	Poor availability
31	Lack of top management commitment
32	Instability in other project sites
33	Lack of Face-To-Face meetings
34	Lack of Management availability and efficiency
35	Unfamiliarity with customer type
36	Constraints due to time zone differences

B. *Hypothesis H3*

To test Hypothesis H3, data was collected from different tasks (2, 3, 7, 8, 3, 4 and 9), which are stated below. The test uses the difference in time and effort between using clustered and non-clustered risks to test this hypothesis. The used time was obtained from tasks 2, 3, 7 and 8 whereas answers for some questions in task 4 (Q2 and Q3) and task 9 (Q1) were used to evaluate the effort. In order to avoid any influence or bias the tasks 7 and 8 were performed separately from tasks 2 and 3 (in both time and sequence). The following sections describe how the used time and effort are obtained and estimated:

1) Used Time:

Used time can give a preliminary indication of whether the task is easy, difficult or complicated. Task 2 and task 7 were the same, except that task 2 was on non-clustered risks and task 7 was on pre-clustered risks. In these two tasks the subjects have been asked to s***pecify two risks for each one of the three perspectives*.**

Task 3 and task 8 were also the same but task 3 was on non- clustered risks and task 8 was on pre-clustered risks. In these two tasks the subjects have been asked to ***specify the perspectives for three pre-ticked risks by the experimenter***.

While the subject was implementing the tasks (2, 3, 7 and 8) the experimenter was monitoring and recording the time. Table II shows the total of the used time during the tasks 2, 3, 7 and 8. The illustrated values in Table II are for the time used by the subjects for both non-clustered and pre-clustered risks. As shown on Table II the subjects spent less time with the pre-clustered risks compared with non-clustered risks for the above tasks, suggesting that clustering from three perspectives reduces the required time for dealing with the risks.

TABLE II. TOTALS OF USED TIME FOR TASKS 2, 3, 7 AND 8

Non Clustered Risks		**Pre-Clustered Risks**	
Task No.	∑ Used Time	Task No.	∑ Used Time
T 2	128.92	**T 7**	42.13
T 3	26.76	**T 8**	8.29

2) Effort:

It is not easy to evaluate effort. We used a set of questions, which were distributed among the tasks in a specific order to gather subjects' feedback and opinions about the usefulness of clustering of the risks from the three perspectives and the spent effort. Moreover, the experimenter monitored the subjects while they were performing the tasks. For this purpose Q2 and Q 3 in task 4 and Q1 in task 9 were designed and asked to the participants. The questions and the answers are shown below:

Task 4/Q2: ***Was it easy for you to specify the risks or perspectives? (Yes / No)***

This question was answered by **29** participants, **13** of them answered ***Yes*** with percentage **44.8 %** and **16** of them answered ***No*** with percentage **55.1 %**.

Task 4/Q3: ***Do you agree with the idea that the above tasks will be much easier and the time and effort can be saved if risks were clustered from the three perspectives?***

Strongly Agree	***Agree***	***Neutral***	***Disagree***	***Strongly Disagree***
1	***2***	***3***	***4***	***5***

As shown in Figure 2 the number of subjects who voted to ***Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree*** are **4, 22, 3, 0, 0** respectively. This means that the majority of the subjects are in agreement (or strong agreement) in their answers for this question.

Task 9/ Q1: ***To what extent do you agree with the idea statement that "concentrating only on the risks of the appointed perspective saves time and effort"***

Strongly Agree	***Agree***	***Neutral***	***Disagree***	***Strongly Disagree***
1	***2***	***3***	***4***	***5***

Figure 3 illustrates the subjects' answers for Q1 in task 9, as shown on the figure the number of answers on the "agree" side (**Strongly Agree and Agree = 20)** is higher than the "disagree" side (**Disagree and Strongly Disagree = 2),** with **6** subjects answering Neutral. This means that the idea of "concentrating only the risks of the appointed perspective to save time and effort" has strong support from the subjects in the experiment.

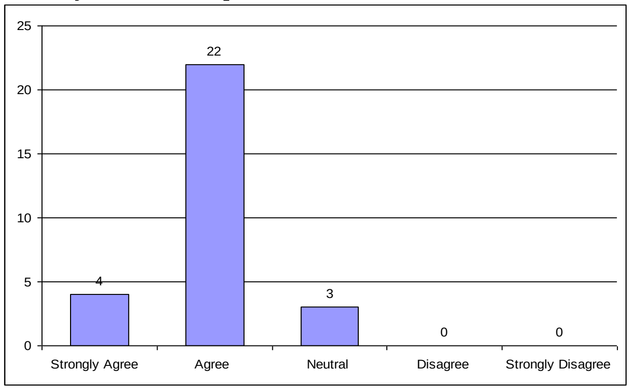


Figure 2: Voting for used effort (clustered and none clustered)

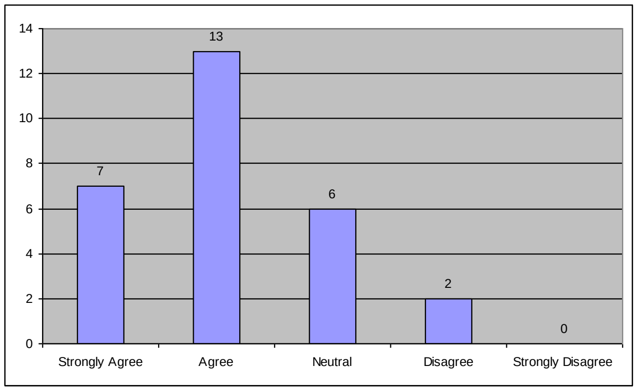


Figure 3: Concentrating on appointed perspective risks saves time & effort

From the above result following points can be remarked:

1- From tasks 2,3,7 and 8 as it can be seen on Table II the total used time for the tasks, which were performed on non-clustered risks was higher than the one, which were performed on pre-clustered risks.

2- After performing tasks 2 and 3 on non-clustered risks the participants were asked "*Was it easy for them to specify the risks or perspectives?*" More than **55%** of them answered **No** to this question. They were asked another question after performing these two tasks, "*Do you agree with the idea that the above tasks will be much easier and the time and effort can be saved if risks were clustered from the three perspectives?*" with 5 options (*Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree*) to select from them. As can be seen in Figure 2 the majority of participants agreed or strongly agreed.

3- After performing tasks 7 and 8 the subjects were asked the same questions but in a different way and in only one question: *To what extent do you agree with the idea statement that "concentrating only on the risks of the appointed perspective saves time and effort" with the same options to answer this question* (*Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree*). Only two subjects disagreed. The rest of subjects agreed (including those who strongly agreed), see Figure 3.

C. *The Support of Our Pre-Clustered list*

As a side product of the experiment we compared our clustering of the W-D potential list of risks with the clustering of the two groups (control and experimental). The result of that will be used to improve the pre-clustered risks. Table III summarizes the result.

We found that some participants have categorized some risks under more than one perspective.

TABLE III. SUPPORT TO OUR CLUSTERING

Supported By: / **Cluster**	**Both Groups**	**Control Group only**	**Experimental Group only**	**No One**
Project Risks	10/13	-	1/13	2/13
Process Risks	4/9	-	1/9	4/9
Product Risks	10/14	-	0/14	4/14
Totals	24/36	0	2/36	10/36

As can be seen in Table III, our clustering of risks has higher support from both control and experimental groups together. In total 66.6 % of our clustering of risks from the 3P perspectives were supported and only 27.7 % were not supported. This is understandable from the numbers shown in Table III. Our clustering for the project and product risks has stronger support from the groups. For the clustering project risks only 2 risks out of 36 were not supported by the groups. Our clustering for the product risk was also strongly supported by the groups as only the clustering of 4 risks out 14 were not supported. By contrast, the support of process risks was medium as the clustering of 4 risks out of 9 was not supported.

VI. DISCUSSION

A. *Study Limitation*

It would have been preferred if the experiment had been undertaken at one of the software development houses or projects, but this was not feasible as most software companies have restrictions with data security. Several local web development companies were conducted but they were not able to participate.

B. *Study Reflection*

The following three points summarize the experiment result:

- The result of the experiment was against the Hypothesis H2, the opposite of what was expected. This was clear as the used time by the experimental group was higher than in the case of control groups. The experiment result showed the need for revising and modifying the criteria in order to improve and rectify them.
- Hypothesis H3 is strongly supported by the experiment result.
- The experiment results show the clustering has a significant impact by reducing the time and effort. This result supports the concept of managing the risks from the 3P perspectives in W-D development, because the risks are distributed between the three perspectives and none of them can be ignored. Previous approaches have considered all the risks from the project perspective or, in the best cases (very rare), they might see them from the process perspectives [3]. This wastes developers' time, effort and leads to them locating more resources for one perspective's risks and ignoring others.
- Our pre-clustered risks list of the risks, which is used in the experiment has gained significant support from control and experimental groups. In general the support came from both groups and only in two cases did it come from the experimental group only. However, the clustering of a few risks were not supported by the groups.
- Some risks could affect more than one perspective.

VII. CONCLUSION AND FUTURE WORK

Since it is difficult to find a suitable project and wait for the risks to actually happen, we emulate such a situation pro-actively in an experimental based setting. We have presented the results of the experiment, which is designed to validate some aspects of our research into the *WeDRisk* approach. The results that were presented in this paper cover usefulness of clustering risks from the three "3P" perspectives (project, process and product) and evaluate our criteria factors that can be used for the clustering. The experiment has taken place at School of Computing Science/ Newcastle University, UK. The recruited subjects were 30 participants (MSc, PhD and Post-doctoral researchers) who either have experience and worked in software development projects or at least have appropriate knowledge with software engineering and software development. The result of the experiment shows the following:

- The clustering of the risks from the three perspectives has got a high degree of support from the subjects.
- It seems that using the criteria for clustering takes more time than clustering without the criteria but:
  - It supports standardization.
  - In the long run, using criteria will take shorter time when the developer becomes familiar with it.
  -Using criteria avoids subjective judgments, which could be different from one practitioner to another.

Without the criteria it is very difficult to decide, which perspective the risk could apply to, particularly if the developer does not have enough experience. In real applications it is expected that developers may have some training on how to use the criteria. The following contributions are made by this work:

- Risks should be considered from the 3P perspectives.
- Clustering identified risks is effective in saving time and effort.

The result of the experiment confirm the importance and usefulness of clustering, considering of the risks from the 3P perspectives as a way for reducing the effort and time in managing the risks and then increasing the efficiency of risk management in W-D developments.

The experiment result also raised the need for updating and improving our proposed clustering criteria to make it more understandable and less time consuming for developers and managers. The result of this experiment will be used to revise and rectify our pre-clustered risks list. Finally, some risks could affect more than one perspective. The experiment has been designed to be ready for any replications in the future if need be.

VIII. References

[1] Mishra, D. and A. Mishra, *Distributed Information System Development: Review of Some Management Issues*, in Proceedings of the Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems, Springer-Verlag:, 2009, pp. 282 - 291.

[2] Jim´enez, M., M. Piattini, and A. Vizca´ıno, *Challenges and Improvements in Distributed Software Development: A Systematic Review.* Hindawi Publishing Corporation Advances in Software Engineering Volume 2009, 2009, pp. 1-14.

[3] Keshlaf, A. and S. Riddle, *Risk Management for Web and Distributed Software Development Projects*, in 2010 Fifth International Conference on Internet Monitoring and Protection ICIMP2010, IEEE Computer Society, 2010, pp. 22-28.

[4] Ebert, C., B. Murthy, and N. Jha, *Managing Risks in Global Software Engineering: Principles and Practices*, IEEE International Conference on global software engineering ICGSE'08, 2008, pp. 131-140.

[5] Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, *Using the WinWin Spiral Model: A Case Stud*, IEEE Computer, 1998, pp. 33 - 44.

[6] Misra, S., U. Kumar, V.Kumar, and M. Shareef, *Risk Management Models in Software Engineering.* International Journal of Process Management and Benchmarking (IJPMB), 2007. **2**(1): pp. 59-70.

[7] Kirner, T. and L. Goncalves, *Software Risk Management: a Process Model and a Tool*, in *Software Engineering Techniques: Design for Quality*, Springer Boston, 2007, pp. 149-154.

[8] Islam, S., *Software Development Risk Management Model – A Goal Driven Approach*, in ESEC/FSE Doctoral Symposium'09 2009, ACM: Amsterdam - The Netherlands.

[9] Kontio, J., *The Riskit Method for Software Risk Management*, Version 1.00 CS-TR-3782 / UMIACS-TR- 97-38, in CS-TR-3782 / UMIACS-TR- 97-38, Computer Science Technical Reports, C.-T.-U.-T.-. 97-38, Editor. 1997, University of Maryland: Maryland.

[10] Fewster, R. and E. Mendes, *Measurement, Prediction and Risk Analysis for Web Application*, in Proceedings of 7th International Software Metrics Symposium, METRICS-2001, IEEE Computer Society, 2001, pp. 338 - 348.

[11] Presson, J., L. Mathiassen, J. Boeg, T. Madsen, and F, Steinson., *Managing Risks in Distributed Software Projects: An Integrative Framework.* IEEE Transaction on Software Engineering, 2009. **56**(2): pp. 508-532.

[12] Rabbi, M. and K. Mannan, *A Review of Software Risk Management for Selection of Best Tools and Techniques*, in Proceedings of 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel / Distributed Computing, IEEE Computer Society, 2008, pp. 773 - 778.

[13] Gorski, J. and J. Miler, *Towards an Integrated Environment for Risk Management in Distributed Software Projects*, in Proceedings of 7th European Conference on Software Quality ECSQ2002. 2002.

[14] Kuni, R. and N. Bhushan, *IT Application Assessment Model for Global Software Development*, in *IEEE International Conference on Global Software Engineering (ICGSE'06)*. 2006, IEEE Computer Society, 2006, pp. 92 - 100.

[15] Keshlaf, A. and S. Riddle, *Web and Distributed Software Development Risks Management: WeDRisk Approach.* International Journal On Advances in Software, vol. 3&4 2010. **Unpublished**.

[16] Dunphy, P., A. Heiner, and N. Asokan, *A closer Look at Recognition-Based Graphical Passwords on Mobile Devices*, in Proceedings of the Sixth Symposium on Usable Privacy and Security SOUPS'10, ACM, 2010, pp. 1-12.

[17] Maxion, R. and K. Killourhy, *Keystroke Biometrics with Number-Pad Input*, In IEEE/IFIP International Conference on Dependable Systems & Networks (DSN-10), 2010, pp. 201-210.

[18] Kitchenham, B., S. Pfleeger, L. Pickard, P. Jones, K. Elemam, and J. Rosenberg., *Preliminary Guidelines for Empirical Research in Software Engineering.* IEEE Transactions on Software Engineering, 2002. **28**(8): pp. 721 - 734.

New Probabilistic-Based Broadcasting Algorithm for Mobile Ad Hoc Networks

Muneer Bani Yassein
Department of Computer Science
Jordan University of Science and Technology
Irbid 22110, Jordan
masadeh@just.edu.jo

Lina Abu-Fadaleh
Department of Computer Science
Jordan University of Science and Technology
Irbid 22110, Jordan
Lanooo86@hotmail.com

***Abstract*- Mobile Ad-hoc Networks (MANETs) have a lot of features, like the autonomous terminal that means each node can function as both host and router. Also, the operations on a MANET are distributed because of the absence of infrastructure and central control of the network. The topology of such network changes dynamically because of the mobility of the nodes. Nodes use Multi-hop routing to guarantee the delivery of messages. A MANET suffers from a set of problems. Most of them arise from the nature of the network itself, since it use wireless communication which is already suffer from the high bit error rate and hence the data exchanged through MANET is more likely to interference, fading, and subjected to noise. In this paper, a new scheme for reactive routing protocols is proposed to decrease the effects of the broadcast storm problem and to discover a more stable route to maximize the throughput of the network and minimize the average delay and the routing overhead.**

Keywords- MANET; Probabilistic-Based; AODV; NPBA.

I. INTRODUCTION

Mobile Ad Hoc Networks (MANETs) are networks of mobile nodes that are connected through Multi-hop wireless links without any infrastructure. MANETs received more attention and became one of the significant areas in network world, because of the wide spread of new technologies such as laptops, and mobile phones [1][9]. MANETs are simply built without the need of any infrastructure, it consists of a number of nodes distributed over a geographical area that dynamically change their locations, they are rapidly deployed, self configured and the nodes are connected through wireless links. Due to the nodes mobility, the topology of such networks is rapidly changes [2][10]. MANETs could be stand alone network or connected to external networks (e.g., Internet).

Ad Hoc network has special characteristics such as highly dynamic environment that make the conventional routing protocols not appropriate choice for these networks. The routing process is one of the most challenging aspects in MANET because of its limited resources, distributed operations, dynamic features and instable wireless links. It needs a routing process that constructs and maintains up-to-date routes with minimum overhead and resources consumption [3][7]. The remainder of this paper is organized as follows: Section 2 reviews broadcast in MANETs. In Section 3, we present our proposed protocol. Then in Section 4, we analyze the broadcast based probabilistic scheme with different system parameters. Section 5 accommodates the simulation environment and analyze of our results, whereas Section 6 concludes this study.

II. RELATED WORK

Flooding [11] is the simplest static routing protocol. It does not need any information about the network topology to deliver packets from the source to the destination. When a node wants to send a packet, it transmits this packet to all its neighbors. Then each node that receives this packet for the first time retransmits it to all neighbors except the neighbor from which it was received. This process is continued until the packet reaches all nodes in the network. Each packet has a unique identifier that consists of the source address, a special sequence number used to prevent sending duplicate packets from the same node, and the destination address [12]. The main disadvantage of flooding is the consumption of the network resources because of the high traffic load it generates. On the other hand, it ensures that the packet reaches to the desired destination and gives a high packet delivery ratio. Such routing protocol is still used as a building block for other enhanced protocols, such as DSR [2], and AODV [12].

Distance Vector (DV) routing algorithms [8] are based on Bellman-Ford formula. Its concern is determining the cost to any node in the network. Each node maintains its routing table which contains information about best routes to every node in the network [11]. Different metrics used to calculate the cost between nodes, such as hop count, queue length, and delay. Nodes flood the cost information to neighbors periodically to update their routing tables. Then each neighbor uses this information to recalculate the costs by applying Bellman-Ford formula and comparing it with its local routes to choose the next hop with minimum cost to each destination. The process is repeated every time new distance vector is received from any of the neighbors that cause a change in the node distance vector. The slow convergence of the routing information is the major drawback of DV algorithms [4].

In the Link State (LS) algorithm, each node has a complete view of all links in the network. A shortest path algorithm is used to determine the best path or route. The best route then is selected based on some metrics like link speed, number of hops, or traffic congestion. Upon topology change, a notification message is flooded to the whole network and each node updates its links state with the new information. The efficiency of LS algorithms is decreased when the size of the network is increased. LS has a highly space complexity because each node stores information for all network elements.

III. NEW PROBABILISTIC-BASED BROADCASTING ALGORITHM

The New Probabilistic-Based Broadcasting Algorithm (NPBA) is an on-demand, broadcast-based; Ad Hoc route discovery protocol that is designed for MANETs. The main goal of this scheme is to minimize redundant broadcasts, and to increase the overall routing performance.
NPBA solves the problem of probabilistic based protocol in sparse networks, where, in such networks nodes do not receive all broadcasts unless probability parameter is high. When probability is 1, this scheme is identical to flooding. So, NPBA adjusts the sending probability of broadcast packet according to certain parameters such as, network density.

IV. DESIGN OF NPBA

NPBA aims to find the best route with lowest cost while preserving network resources. NPBA modifies the route discovery phase specifically the propagation of RREQ packets of the original reactive protocols. Other phases are the same as the original ones. To implement this scheme, a list of neighbors is required to keep track of the current neighbors of the node.
Depending on the network topology, each node in the network is assigned a probability to rebroadcast the upcoming messages. When source node receives a broadcast message, it runs a broadcast procedure to decide whether to continue the broadcast process or to drop it.
According to network topology, network is divided into dense and sparse areas. In dense area, nodes that are located in such area have a low sending probability to incoming broadcast messages. That is clearly minimizes the number of rebroadcast messages. This minimizes the opportunity to reach new sources in the network. But according to this scheme, nodes in dens area check neighbors periodically for any changes of neighborhoods information to ensure that the message is delivered correctly. This can be achieved by adjusting the sending probability.

In Spars area, nodes that are located in such area have high sending probability to incoming broadcast messages. However, if the nodes sense the possibility to reach more nodes it adjusts the sending probability.

The propose algorithm is outlined in figure 1. NPBA consist 5 steps as follows: in step1 and periodically, each node broadcasts a HELLO message containing its address and list of neighbors. In step 2 and upon receiving the HELLO message, a node updates its routing tables and list of neighbors. At any time, the list of neighbor for a particular recipient (*Y*) will contain the addresses of all (*Y*'s) 1-hop neighbors. Then source nodes will run a small procedure for comparing list of neighbors to adjust the forwarding status of the node. After that, in step 3 and 4, each node now adjust its sending probability according to the result of step 2.

When a source node *S* wishes to communicate with a destination *D*, and there is no known route to this destination, it prepares a RREQ message.
Upon receiving the RREQ message in step 5, nodes will propagate the RREQ according to their probability values, where nodes that in their forwarding status will have higher opportunity to propagate the upcoming messages.

```
Algorithm:  New Probabilistic-Based Broadcasting Algorithm
  Input:      Ad Hoc network with n nodes
  Output:    route between nodes with minimum cost
    1.  Periodically, every HELLO-INTERVAL broadcast
        a HELLO message, which is already attached
        with list of friend.
    2.  On Receiving a HELLO message:
          1.  Update list of neighbor, so that it will contain
              1-hop neighbor address for all neighbors.
          2.  Compare lists of neighbors, to find new
              destinations.
          3.  Update forwarding status
                If there are new destination could be
                reached
                    Forwarding status = true;
                Else
                    Forwarding status = false;
    3.  Determine the probability of sending according to
        network topology, where dense area has low
        probability, and sparse area has high probability.
    4.  Check Forwarding status with the probability and
        adjust sending probability according to neighbor's
        information.
    5.  Upon receiving an RREQ message, the following
        actions take place:
                If the recipient node is the destination
                   Done
```

Figure 1. An outline of the new probabilistic-based algorithm

V. PERFORMANCE EVALUATION

Simulation experiments are carried out on T6400-2 GHz computer with Intel Core 2 Duo processor, and 4 GB RAM. The operating system is Fedora 10. Network Simulator (NS) version 2.29 was used [13].

NS is discrete even simulator targeted at networking research, it provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks, it is heavily used in Ad Hoc networking, it is widely used in a academia due to its open source model.

NS supports many protocols for wireless network such AODV, DSR, and others. The proposed scheme is compared with AODV. AODV [6] was chosen due to the high delivery ratio it scores and low overhead comparing to other routing protocols.

Each run for the simulation lasts for 900 seconds, each simulation scenario within same experiment is repeated 30 times, and their average value is taken to increase accuracy. To study the effects of network density, we used 50, 75, 100, and 125 nodes that are randomly distributed in 600 X 600 m^2 simulation area. And for deep measurement and judgment of our scheme we try different node speeds (5, 10, and 20 m/s). Also different pause times were used (0, 2, and 4 seconds).

The simulations conducted have revealed that a typical value for a sparse region in a network size of 50 nodes contains Ns =4 nodes while a medium region contains Na=16 nodes and a dense region contains Nd =30 nodes. As a consequence, there are, on average, 16 rebroadcasts in highly adjusted probabilistic flooding when the rebroadcast probability, for example, is set at p_1=0.7, p_2=0.35 and $p3$=0.25, respectively.

To study the performance of NPBA and compare it against AODV, two different types of simulation scenarios are conducted:

- **Density Scenarios**: is to study the effect of change in node density on the performance metrics for different protocols.
- **Mobility Scenarios**: is done by varying the maximum speed of the nodes to see how it affects the behaviors of the protocols in terms of some measured metrics.

Our first scenario is to experiment 50, 75,100, and 125 nodes, with speed =5m/s, packet rate equal to 4 packets/second, and pause time= 0.

The simulation results presented in Figures 2, 3, 4, and 5 illustrates the routing packets, packet delivery ratio, normalized load, and average-end-to-end delay.

From Figure 2, it is clear that our protocols achieve major enhancement in terms of reducing the routing overhead for all speed values. This is due to the fact that our protocol tends to control flooding by selecting only a subset of nodes to retransmit packets. This reduction of retransmission saves many control packets from being sent, and this reduces the overall routing overhead. Figure 3 shows also that as the number of nodes increases, the overhead encountered by AODV increases as well. This is because the large number of nodes, the more control packets need in order to manage the whole networks. For small number of nodes, NPBA outperform AODV at most about 32.57 %. The enhancement becomes less significant when number of nodes is large, which reaches about 14%.

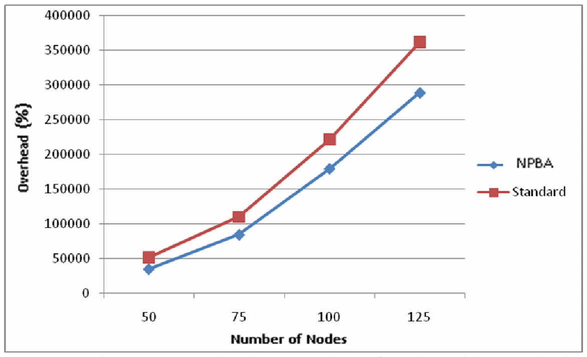


Figure 2. Overhead vs. Number of nodes, with speed =5

The overhead reduction achieved in Figure 2 is reflected positively in the network normalized load as can be seen in Figure 2. For low density networks, the performance of AODV is close to that of NPBA. However, for high density networks, NPBA outperform AODV, but in general as network density increased, the normalize load also increased, due to the huge number of control packets need to be exchanged, and the contention and collisions of these packets.

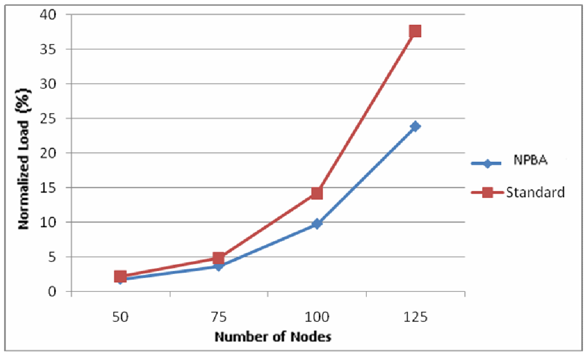


Figure 3. Normalized load vs. Number of nodes, with speed =5

Figure 4 displays the packet delivery ratio for the two protocols and shows the superiority of our prtocol for all numbers of node. This is an expected result since the network overhead decreased and hence number of collisions is decreased, which maximize the chance of delivering packets to its destination.

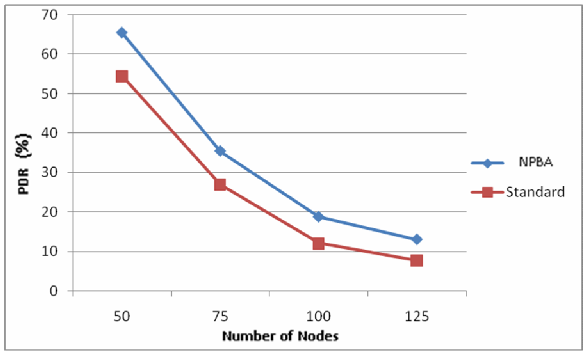


Figure 4. Packet delivery ratio vs. Number of nodes, with speed =5

Figure 5 depicts the average end-to-end delay achieved by our protocol in comparison with that achieved by AODV. As the Figure shows clearly, our protocol outperforms AODV, which is expected as the end-to-end delay metric includes delays caused by route discovery, queuing and retransmissions at the MAC level. Due to the fact that the routing overhead of our protocol is low and minimized, the packets are no longer needed for a long period of time, in addition, since the number of rebroadcasts is reduced, this will reduce the average end-to-end delay.

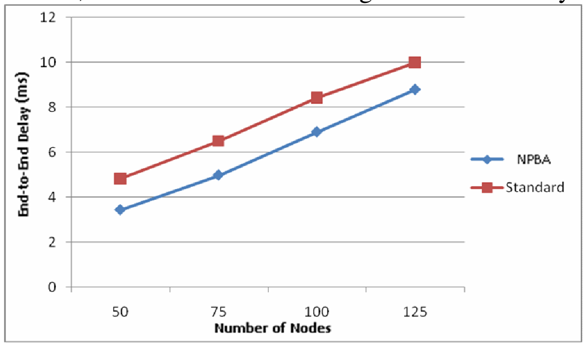


Figure 5. End-to-End Delay vs. Number of nodes, with speed =5

The simulation results presented in Figures, 6and 7 illustrates the routing packets, packet delivery ratio, normalized load, and average-end-to-end delay.

Figure 6 shows that with increasing number of nodes, the overhead of AODV result from number of routing packets increases significantly. It is obviously that NPBA outperforms AODV moderately in all number of nodes with average of enhancement equal to 25.94%.

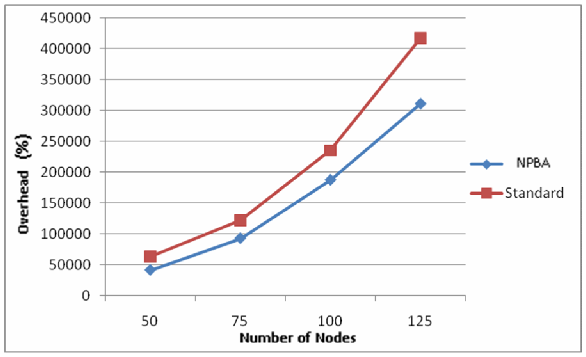


Figure 6. Overhead vs. Number of nodes, with speed =10

In Figure 7 it can be noticed that at low density, our protocol and the AODV have almost identical normalized load values. However for high density values, our protocol become evidently superior with 46% enhancement score.

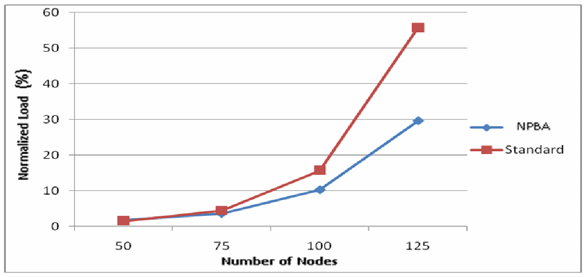


Figure 7. Normalized load vs. Number of nodes, with speed =10

Figure 8 shows that for different network density and as the nodes density increases in the Ad Hoc network, the packets delivery ratio decreased. In addition, the Figure shows that increasing the number of nodes results in lower delivery ratio for all protocols compared.

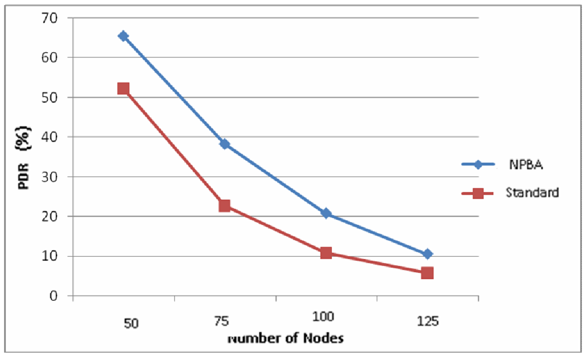


Figure 8. Packet delivery ratio vs. Number of nodes, with speed =10

VI. CONCLUSION AND FUTURE WORKS

In this paper, a new scheme of reactive routing protocols is proposed to decrease the effects of the broadcast storm problem and to discover best route with minimum cost to maximize the throughput and minimize routing overhead and the average end-to-end delay. The major contributions of our protocol are The NPBA is reliable broadcast-based protocols that avoid the negative attitude of simple flooding, which causes a very high overhead. We minimize number of redundant broadcast message, contention and collision by allowing only specific nodes to participate on broadcast propagation. For future works, it would be interesting to compare the performance of our proposed protocol with a dynamic probabilistic algorithm on a Dynamic source Routing protocol (DSR).

REFERENCES

[1] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multi-hop Mobile Ad Hoc NETWORK. ICDCS '01, in proceedings of the 21st International Conference on Distributed Computing Systems; 2001.

[2] S. Corson and J. Macker. Mobile Ad Hoc Networking (MANET". IETF Internet Draft, http://tools.ietf.org/id/draft-ietf-manet-issues-02.txt; 1999. [Online] [Accessed 2008 June].

[3] K. Timo, A. Christian, E. Stephan, S. Christoph, and S. Markus. The Scalability Problem of Vehicular Ad Hoc Networks and How to Solve it. In IEEE Wireless Communications Magazine; 2006.

[4] Muneer Bani Yassein, Saher S. Manaseer and Abdallh .Al-Turani, "A Performance Probabilistic Broadcasting of Ad hoc Distance vector(AODV) using Fibonacci Increment Backoff Algorithm" , 25th UK Performance Engineering Workshop, Leeds, UK , July, 6-7, 2009.

[5] E. Royer and C. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. IEEE Personal Communications; 1999; 6(2): 46-55.

[6] C. Chiang, H. Kuang, W. Liu, and M. Gerla. Routing in Clustered Multi-hop Mobile Wireless networks with Fading Channel. IEEE; 1997.

[7] G. Pei, M. Gerla, and X. Hong. 1ANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility. In MobiHoc '00, Proceedings of the 1st ACM international symposium on Mobile Ad Hoc networking & computing; 2000; 11-18.

[8] K. Gorantala. Routing Protocols in Mobile Ad Hoc Networks. M.S. Thesis, Umeå University, Sweden; 2006.

[9] A. Pirzada, M. Portmann, and J, indulska. Hybrid Mesh Ad Hoc On-demand Distance Vector Routing Protocol. Australian Computer Science Conference (ACSC), Ballarat; 2007.

[10] D. Johnson, D. Maltz, and Y. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). IETF Internet Draft, draft-ietf-manet-dsr-10.txt; 2004.

[11] Muneer Bani Yassein, Sanabel Nimer, and Ahmad AL-Dubai. The Effects of Network Density of a New Counter-based Broadcasting Scheme in Mobile Ad Hoc Networks. Proceeding of the 10th IEEE International Conference on Computer and Information Technology (CIT2010), 29 June – 01 July, 2010, Bradford, UK

[12] S. Das, C. Perkins, and E. Royer. Ad Hoc On Demand Distance Vector (AODV) routing. IETF Internet Draft, http://tools.ietf.org/id/draft-ietf-manet-aodv-13.txt; 2003. [Online] [Accessed 2008 October]

[13] The Network Simulator Site. [Online] [Accessed 2011 January].Available from URL, http://www.isi.edu/nsnam/ns.

A New Approach in a Multifactor Authentication and Location-based Authorization

David Jaros, Petr Bednar, Kuchta Radek
Department of Microelectronics
Brno University of Technology, FEEC
Brno, Czech Republic
jarosd|kuchtar@feec.vutbr.cz, xbedna06@stud.feec.vutbr.cz

Abstract — **This paper is focused on location-based authentication and authorization in a network environment. We propose a new approach, where the user's biometric credential and the user's location are used. A basic framework for the MALBA (Multifactor Authentication and Location-based Authorization) is defined in the article. We describe processes of initial binding, authentication and authorization. Finally, the MAD I (Multifactor Authentication Device) is introduced. The MAD I provides user's credentials for authentication and authorization processes. The user will get roles in the system dependent on his or her position.**

Keywords-location-based authentication; GPS; AES; multi-factor authentication; RBAC, embedded system

I. Introduction

Different contents and resources in the network environment require different security level. The security level is hard related to the user's authentication process. Protected services or resources that need higher security level adopt more effective authentication techniques. For example, a simply email client requires either login-password authentication (secret information). A user accessing to his or her bank account, where multifactor authentication technique is used, is a different case. The most often authentication techniques can be divided into three main groups. The first group is based on the knowledge of secret information, a password [1]. These authentication techniques are commonly used for authentication in the web services. The second group is formed by authentication techniques based on ownership of subject (token) that is unique in the system framework [2]. A token can be represented by a hardware key that is used for protection of computer program against illegal copying. The last group includes authentication techniques, which verify some of human user's biometric property such as fingerprint [3].

A piece of information about the user's current position is an additional factor that can be exploited in an authentication process, as refers [4]. The importance of the location-based authentication (LBA) is increasing especially for mobile users [5]. The advantage of LBA can be found in hospital sector as well. A doctor shouldn't handle with patient's privacy information out of hospitals. If is needed, he/she can define with cooperation with the administration desk the new safe area (his/her home). Also when the user wants to get access to his or her bank account, LBA can be used. The advantages of LBA are furthermore discussed in [5, 6].

The systems for access management are commonly called AAA systems (Authentication Authorization and Accounting) due to its processes [7]. The user's position information can be addressed into each of them. For a user's identity evaluating, rights dependent on his or her position can be assigned to the user's identity and finally a payment rate for services can dependent on his or her position.

The user's position is very sensitive information that can be abused in many cases. User's position can be also exploited for the position-targeted spam. For these reasons it should be operated very carefully with position information over whole its lifecycle. Position information should be anonymous as much as possible. The level of anonymity is dependent on required accuracy of position information. For instance, if the service requires position information for country determination, the position information shouldn't be interpreted in accuracy with a few meters.

In this article, we propose a new approach of mobile user authentication and authorization called MALBA that connects multifactor authentication and authorization. In our network framework we introduce embedded terminal MAD I (Multifactor Authentication Device) that performs user's fingerprint, user's position information and stores encryption keys.

The rest of the article is organized as follows. MALBA's application scenario, processes of the initial binding between MAD I and domain controller and authentication and authorization are described in the Section II. Next section introduces the MAD I. The final section is focused on conclusion and future work.

II. Application scenario

We assumed the application scenario as shown in Figure 1. The user wants to get access to protected domain content as are resources, services clients. MAD I is connected to the user's terminal. The request for protected content from the user is redirected to domain controller which performs access management. The user is challenged for giving in its credentials. If the user has connected MAD I it provides position information and fingerprint. Methods for fingerprint processing generally product same hash for the same fingerprint, otherwise a fingerprint reader cannot be use in the identity verification. Position information and fingerprint are encrypted by AES (Advantage Encryption System) [8]. The user adds its login and data are sent to a domain controller. The domain controller will solve user's authentication dependent on receipt credentials. If identity is verified, the user's roles in the domain are defined. For the system the RBAC (Role Based Access Control) is used [9].

An area management presents a database, which stores definition of user's areas. The areas are defined by two ways. A simpler way is to define one point and the distance from it (radius). Then we get a circle from where the user will get the access. The definition of net of triangles is more complex (leads to convex combination). This way is more difficult as for definition, storing and evaluating but gives us an advantage in definition area of any shape. Defined areas are stored within IDs and can be used by any users. The defined area can mean different roles (rights) for different users. The user can cooperate with the administration desk to define new area. A pairs area's ID - roles are stored in a user's profile in Active Directory. Dependent Appropriate areas are requested by domain controller from areas management. Domain controller contains an API for evaluating position information (if a user is or is not in evaluated position). The order in which area's IDs are stored in user‘s profiles defines areas priority. The last added ID in the list has the highest priority. This right solves the overlapping problem.

API in the domain controller evaluates mutual position between user's position and areas defined for its identity.

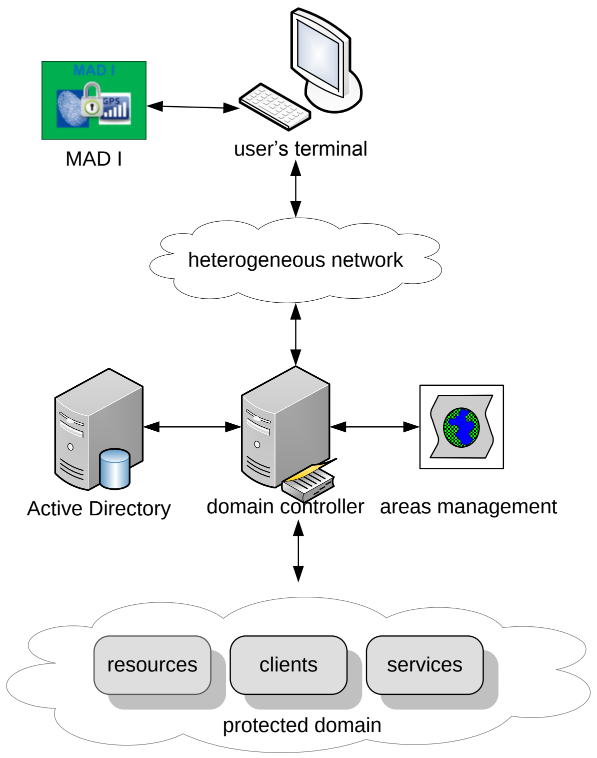


Figure 1. MALBA's application framework

A. Initial binding

Before the first user is authenticated mutual binding has to be done. Initial binding has to be executed at the system administration desk over local bus (MAD I is USB enabled). Binding process performs AES key exchange between MAD I and domain controller resp. Active Directory, where the key is stored during binding. A hash of user's fingerprint is also stored on the server side. This process can also cause MAD I can be assigned to exact user. Initial binding is described in Figure 2 in following steps.

1. At first, a secured channel should be established. This is done by Diffie-Hellmann key exchange [10]. Two unknown sides can derive the secret key. This technique is often used for exchange of symmetrical encryption key.
2. When the secured channel is established, domain controller generates encryption key for AES. Length of the key is 256 bytes.
3. The key is sent over the secured channel created in the first step.
4. The MAD I stores the key in secured memory after receipt.
5. The user is requested to swipe his or her finger on the fingerprint reader on the MAD I.
6. Hash of the user's fingerprint is sent to domain controller.
7. User's fingerprint hash is stored in the user's profile in the Active Directory.

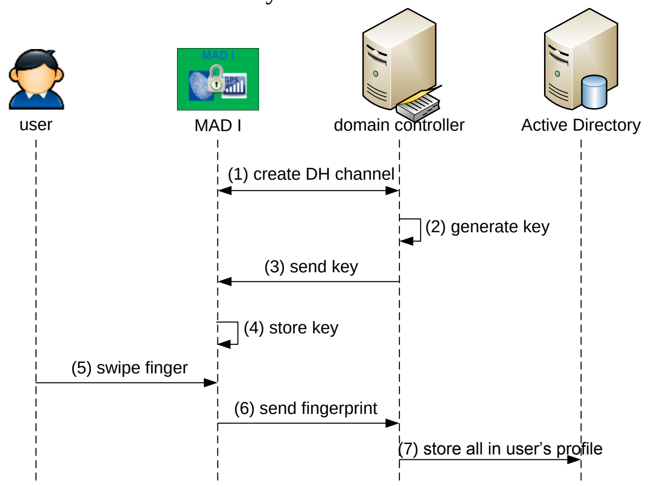


Figure 2. Initial binding

The areas are defined over GUI on the administration desk. The defined areas have to be stored for the current user in the area management and their IDs have to be stored with assigned roles in the Active Directory.

As written above we can define two kinds of areas. The first one is a circle defined by the center and its radius. To define general form we added the second type to the administration interface of area management that lies in the sequence formed by adding points. This feature exploits math method of the smallest triangles and convex combination is used for evaluating.

B. *User's authentication*

When initial binding is done, both sides share the same encryption key and server side has stored the user's fingerprint hash. From this point server side is able to examine user's credentials as fingers. In next steps authentication and authorization processes are commonly described. We assume that the MAD I will be used in areas with free view on the sky. For indoor using, controlled areas should be covered by signal from signal repeater GPS. The situation is illustrated in Figure 3. The description starts after server side's request for credentials.

1. During the first part of the whole process user's credentials should be collected. Therefore the user swipes his or her finger.
2. The GPS (Global Position System) receiver gets position coordinates. The MAD I has to wait after power on for the position evaluating dependent on signal conditions.
3. The position *LOC* is encrypted by AES and user's finger print hash *UHFP* is used as key. The product of this step is cipher *EL*.
4. The second step of the encryption on the client side is provided by encryption cipher *EL* from the second point by symmetrical key *KEY*. The product is cipher *EAD*.
5. The encrypted credentials are sent to the user's terminal.
6. The user is requested to type his or her login.
7. The login with the user's credentials is sent to the domain controller.
8. The domain controller requests from the Active Directory needed data form user's that is related to received login. Data contains shared KEY, user's fingerprint hash and for user defined pairs (area ID – roles).
9. The domain controller receives requested data from the Active Directory.
10. The domain controller decrypts received cipher *EAD* by *KEY* from the active directory. The product of this step is *EL'*.
11. The domain controller tries to decrypt EL' by user's fingerprint hash CHFP from the Active Directory. If the result is an understandable position information, the user's identity is authenticated.
12. The domain controller sends a request to the area management, defining areas for the user.
13. The domain controller gets the IDs dependent on requested area from the areas management.
14. User's position is evaluated in relation to defined areas on the domain controller.
15. Dependent on the results from the previous step, the user has assigned roles for the current domain.

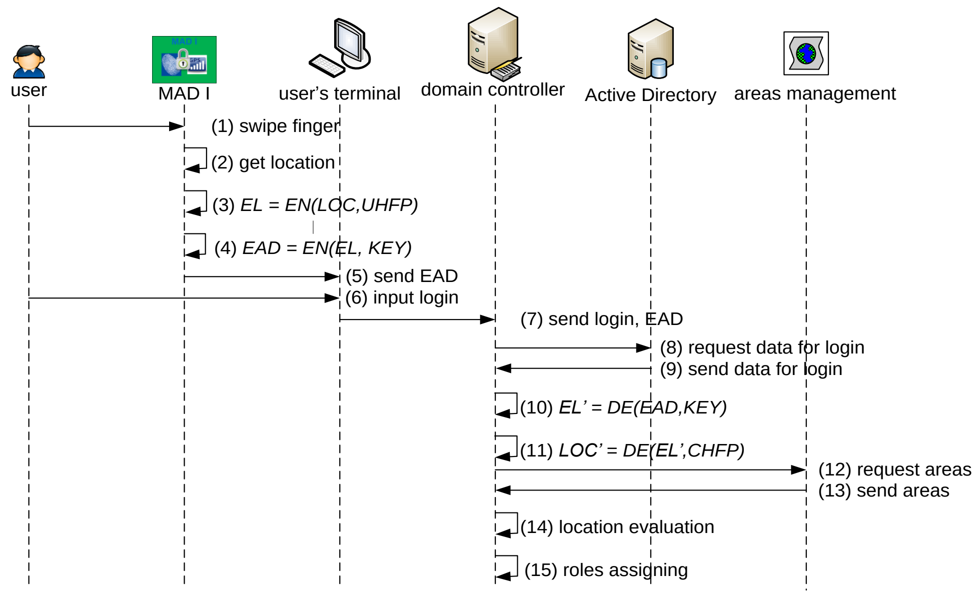


Figure 3. Authentication a authorization processes

III. THE MULTIFACTOR AUTHENTICATION DEVICE I

Multifactor authentication device MAD I was developed for the MALBA's framework.

The MAD I collects principally three authentication factors as ownership of certain device, fingerprint and user's position, where the user's position is used in the authorization as described in the second section.

The MAD I is connected to user's terminal via USB (Universal Serial Bus). The device is designed as a pocket device. The block diagram of the MAD I is in Figure 4.

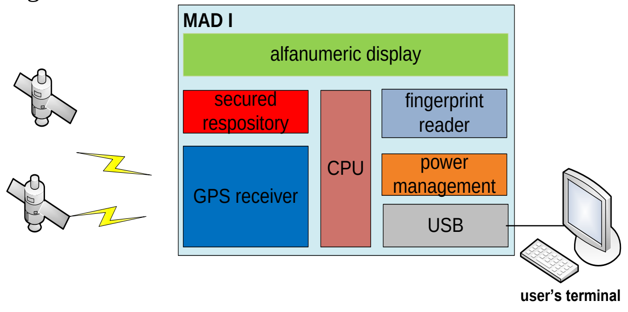


Figure 4. The MAD I block diagram

The position information is provided by the receiver GPS. The assembled GPS received is ready to Galileo for future use with European GNSS (Global Navigation Satellite System). As described above, the fingerprint reader is used for the user's authentication. For the security reasons the symmetrical encryption key is stored in the secured data repository. The secured data repository has special features that protect stored data against to unauthorized reading or writing. Alphanumeric display is assembled for communication between the user and MAD I.

The MAD I is a battery-powered pocket device. The power management contains circuits for adjustments power voltages for the other blocks and circuits for battery charging over USB.

IV. Conclusion and Future work

The importance and possible applications of the user's position information in the access management is discussed in this article. We introduced the newly designed technique MALBA which is addressed for authentication and authorization of mobile user in the WLAN (Wireless Local Area Network) environment. We described the process of initial binding between device MAD I and domain controller. Next, the authentication and authorization processes are described. The device MAD I is described in the final section.

The hardware implementation of the MAD I is already done. The future work will be focused on software implementation of the MAD I. Setting up a test bed for testing proposed technique in real conditions has to be worked on also. We will test proposed technique in the ordinary network environment. The results from testing will be used as input data for next development and will be published.

Acknowledgement

This research has been supported by the Czech Ministry of Education, Youth and Sports in the frame of MSM 0021630503 *MIKROSYN New Trends in Microelectronic Systems and Nanotechnologies* Research Project, partly supported by 2C08002 Research Project *KAAPS Research of Universal and Complex Authentication and Authorization for Fixed and Mobile Computer Networks* in the frame of the National Program of Research II, ARTEMIS JU in Project No. 100205 *Process Oriented Electronic Control Units for Electric Vehicles Developed on a multi-system real-time embedded platform*, by ENIAC JU in Project No. 120001 *Nanoelectronics for an Energy Efficient Electrical Car*, partly by the Czech Ministry of Industry and Trade in projects FR-TI1/057 *Automatic stocktaking system* and FR-TI1/058 *Intelligent house-open platform*.

References

[1] H. Jiang, "Strong password authentication protocols," in *Distance Learning and Education (ICDLE), 2010 4th International Conference on*, 2010, pp. 50-52.

[2] H. K. Lu and A. Ali, "Communication Security between a Computer and a Hardware Token," in Systems, 2008. ICONS 08. Third International Conference on, 2008, pp. 220-225.

[3] E. Sano, et al., "Fingerprint Authentication Using Optical Characteristics in a Finger," in SICE-ICASE, 2006. International Joint Conference, 2006, pp. 1774-1777.

[4] E. Bertino, et al., "Location-Aware Authentication and Access Control - Concepts and Issues," in 2009 International Conference on Advanced Information Networking and Applications, ed, 2009, pp. 10-15.

[5] D. E. Denning and P. F. MacDoran, "Location-based authentication: Grounding cyberspace for better security," *Computer Fraud & Security*, vol. 1996, pp. 12-16, 1996.

[6] G. Lenzini, *et al.*, "Trust-enhanced Security in Location-based Adaptive Authentication," *Electronic Notes in Theoretical Computer Science*, vol. 197, pp. 105-119, 2008.

[7] H. Rui, *et al.*, "A novel service-oriented AAA architecture," in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, 2003, pp. 2833-2837 vol.3.

[8] L. Chi-Feng, *et al.*, "Fast implementation of AES cryptographic algorithms in smart cards," in *Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on*, 2003, pp. 573-579.

[9] M. L. Damiani, *et al.*, "GEO-RBAC: A spatially aware RBAC," *Acm Transactions on Information and System Security*, vol. 10, Feb 2007.

[10] Y. Eun-Jun and Y. Kee-Young, "An Efficient Diffie-Hellman-MAC Key Exchange Scheme," in *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, 2009, pp. 398-400.

Secure Access Node: an FPGA-based Security Architecture for Access Networks

Jens Rohrbeck, Vlado Altmann, Stefan Pfeiffer, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
Rostock, Germany
{*jens.rohrbeck;va031;sp385;dirk.timmermann*}*@uni-rostock.de*

Matthias Ninnemann, Maik Rönnau
Nokia Siemens Networks GmbH & Co. KG,
Broadband Access Division
Greifswald, Germany
{*matthias.ninnemann;maik.ronnau*}*@nsn.com*

***Abstract*—Providing network security is one of the most important tasks in today's Internet. Unfortunately, many users are not able to protect themselves and their networks. Therefore, we present a novel security concept to protect users by providing security measures at the Internet Service Provider (ISP) level. Already now, ISP are using different security measures, e.g. Virtual Local Area Network tags, MAC limitation, or MAC address translation. Our approach extends these security measures by a packet filter firewall and a deep packet inspection engine. A firewall and a deep packet inspection system, at the ingress of the network, offers security measures to all connected users, especially to users with limited IT expert knowledge. Adjustments can be made only by the ISP administrator. Consequently, our security system itself is secured against attacks from users and from the network side. Our approach includes a powerful Packet Classification Engine, a high speed Rule Set Engine without using Content Addressable Memory and control stages in reconfigurable hardware. Our goal is to be able to control network traffic at wire speed.**

***Keywords*-Access Network, Hardware Firewall, Intrusion Detection, Web Filter**

I. Introduction

Firewalls and anti-virus programs provide basic protection for Internet-enabled devices. Normally, these security measures are installed on computers of users. But installing security measures at the users' side has two serious drawbacks. Firstly, the threat detection is done on the target machine. Secondly, the users must install, upgrade, and maintain these security measures without professional support. Other measures such as a Web filter and a deep packet inspection engine like snort are often not installed and require additional maintenance. In addition, the majority of Internet users is missing the necessary expertise to configure their security software so that it provides optimal protection. Furthermore, because of negative experiences like phishing attacks targeting online banking, many users have lost their confidence in online services and the Internet itself. Therefore, it is mandatory to disburden respectively to support users in issues of Internet security.

A trustworthy place for the placement of security measures is the ingress of the network — the access network. Each user, referred to as subscriber by Internet Service Providers (ISPs), is connected to the Internet through the access network. The access network itself consists of access nodes (AN).As ANs are transparent for subscribers, these components are safe from, e.g., Denial of Service Attacks. To reestablish the subscribers' confidence into the Internet and moreover, to even protect the Internet itself, it is useful to establish additional security services at ANs. With these additional security features, two objectives can be achieved. On the one hand, the subscriber is offered a higher security service without the need to care about security measures himself. On the other hand, outgoing traffic from subscribers can be verified. Thus, the network is protected as well.

However, although an ISP can take up new security measures in its portfolio, various challenges have to be addressed. On an AN, higher traffic rates (e.g., 1 Gbit/s or higher) have to be processed than in single Internet connections — both in up- and downstream. Furthermore, rules for up to 32k connections should be supported. Due to hardware restrictions, we dispense with connection tracking and the control of protocols' communication sequences. Our approach referred to as Secure Access Node (SecAN) extends the currently available security measures on an AN by a packet filtering firewall, Web filtering, and intrusion detection system. Thereby, these functionality moves from the subscriber to the ISP.

To fulfill these tasks under the conditions described, a very powerful packet classification [1] and packet processing are required. Due to these requirements, pure software solutions are not applicable. Therefore, we use a hardware/software solution on a XILINX evaluation board with a FX70T Field Programmable Gate Array (FPGA). In our solution, we do not use CAM memory. Already for 224 connections (these approximates ca. 0.7% of all connections), over 90% of available block ram ressources or 23% of slice register would be needed. Without using CAM, our solution is able to control traffic at wire speed. Briefly summarized, the main contributions of this paper are the following:

- We present a novel hardware/software approach of a packet filter, Web filter, and an intrusion detection system placed onto an access network.
- Our solution is able to control traffic in up- and downstream direction simultaneously. Thus, we can protect the connected subscribers and the network itself.

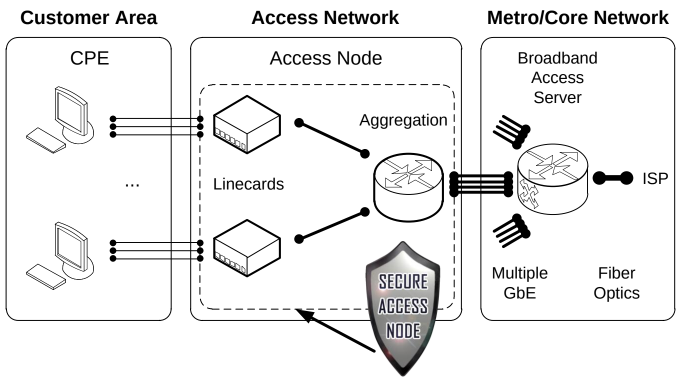


Fig. 1. Access Network containing ANs. The Secure Access Node is an extension of an AN.

- We aspire an individual classification for each connection, e.g., we do not want to limit the number of configurable rules as described in [1].
- As target platform, we use a XILINX evaluation board with an FX70T FPGA. As we do not using CAM, we are able to control traffic with 2 Gbit/s in wire speed, e.g., we control each packet without packet loss.

The remainder of this paper is organized as follows: Section II describes security measures available in the access network today. In Section III, our hardware design is presented. Here the various modules and their functions are explained. Before the paper concludes in Section V, we introduce our software solution for flexible configuration in Section IV.

II. Security Measures In The Access Network

Each subscriber achieves access to the Internet through the access network. Access networks comprise subscriber premise equipments (CPE) and access nodes such as DSLAMs. The latter usually consists of linecards and aggregation cards as shown in Figure 1. While aggregation cards provide high-bandwidth interfaces towards metro or core networks, linecards aggregate the various subscriber lines.

Although the network ingress is transparent to the traffic from and to subscribers, ISPs have to protect the access network. Today, security measures mainly include passive measures on OSI layers II and III [2], [3]. For example, ISPs are using security measures like:

- Port isolation - subscriber may not communicate via an AN
- MAC antispoofing - a Source MAC address is allowed only at one port at a time
- MAC address limitation - to limit the number of MAC addresses per port
- MAC address translation - subscribers MAC address is translated to an ISP MAC address
- VLAN tags - to separate subscriber and services
- IP antispoofing - only the IP address - assigned by the ISP - in combination with the requested MAC address, is allowed pair Source IP and Source MAC at a special port

To ensure a minimum necessary level of security when connecting to the Internet, the already introduced security measures must be integrated into the access area by means of the Secure Access Node.

III. SecAN - Architecture

A. Hardware Overview

To emulate the SecAN on an AN, we use the XILINX ML507 evaluation board with an FX70T FPGA [4]. Thereby the FPGA is the main component. We also utilize the 1MB Static Random Access Memory (SRAM) and the 512MB large Double Data Rate Synchronous Dynamic Random Access Memory (DDR2-SDRAM). To control traffic in upstream and downstream direction, we use two 1 Gigabit Ethernet transceivers.

B. The System In General

Each Ethernet transceiver of the evaluation board is able to process data with 1 GBit/s. If we want to process all data from both directions, we have to process 16 bits/cycle. To avoid the discarding of any uncontrolled data frame and due to the internal delay during frame processing, we have decided to increase the internal bandwidth to 32 Bit/cycle.

The basic components of the SecAN system are the packet classification engine (PCE), rule set engine (RSE), and packet processing engine (PPE) (see Figure 2).

C. The Frame Configuration and Processing In General

- Before the system can process traffic, it must be configured. The components that need to be configured are the PCE, RSE, Web filter, and the DPI control stage. All configuration data is solely written to the hardware and read from it by the ISP. The configuration flow is shown by dashed arrows in Figure 2.
- After configuration, frames reach the inner system. The frame multiplexer chooses the next frame to be processed by the PCE. The PCE separates flow data from the frame and requests the individual rule set from the RSE. The rule set is an individual collection of rules, which are necessary to evaluate a frame. After identifying the right rule set, it has to be forwarded to the PCE. If the rule set reaches the PCE, the rule set, the data frame, and collected frame parameters have to be sent in the direction of the PPE - to the control stages. In the control stages, the rules from the rule set are applied. The control stages are able to discard or forward frames or replace frame values like IP addresses. If a frame is not discarded, it leaves the PPE and is forwarded to the right output interface.

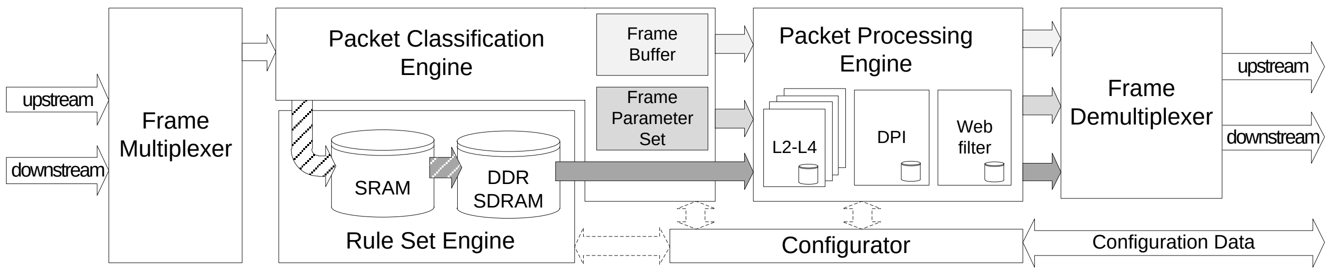


Fig. 2. Block diagram of the Secure Access Node

D. Configuration Of The Hardware

Before the hardware components are not configured, no frames traverse the SecAN. During configuration, all internal processes are stopped. Configuration data is provided to the appropriate modules by the configurator. This data has a type-length-value layout.

- Type is an 8 bit field and determines the component of the hardware to be addressed. Each component has two valid type values: one for writing configuration data and one for reading configured data.
- Length is an 8 bit field and represents the number of configuration data bytes. A maximum of 256 bytes plus configuration header can be configured.
- The actual configuration data is contained in the value part. All components are assigned specific configuration values.

E. The Frame Processing Flow

The frame processing flow is shown by shaded arrows in Figure 2. If the PCE is not busy, it has to receive and classify the frame. The frame multiplexer selects a frame from the internal buffer with the highest fill level. After frame classification is finished, the RSE searches an individual rule set for each frame. Rules of the rule set are applied to the frame. If the frame is not discarded by the PPE due to the rules, it is sent to the correct output interface by the frame demultiplexer.

1) Packet Classification Engine: Often packets are classified by five packet header fields: Both IP addresses and port numbers, and transport layer protocol [5], [6], [7]. Upon agreement with our co-operation partner, which develops end products for ISPs, we want to support a higher degree of flexibility. Thus, we extend this set by both MAC addresses, up to 2 VLAN tags, and the Ether type field to a set of 10 frame parameters. During the configuration phase, the PCE has received two so-called flow id triggers (up-/down stream trigger). These triggers describe, which of the 10 frame parameters are necessary to classify a frame. It is possible to set a new trigger by reconfiguration on the fly.

In addition to the frame, the frame multiplexer delivers information of the receiving direction of frames. Depending on the receiving direction, the flow id trigger for upstream or downstream is selected. After this, the corresponding frame parameters are combined to a unique flow id, which identifies the frame bijectively. Furthermore, all described frame parameter are extracted and stored, and the receiving frame is buffered.

During composing the flow id, we calculate an address for the rule set by CRC32 on the fly. Similar projects like [5], [6], [8] have very short flow IDs and use CAM or bloom filter approaches to increase the lookup performance as suggested in [1]. A CAM would require a disproportionate number of hardware resources and a bloom filter approach is not able to calculate an address for a rule set. We avoid these both solutions and use a two-stage approach in the RSE.

After the flow id is completely composed, a request for the individual rule set is performed by the RSE. If one of the frame parameter is not available in the frame, the flow id is not fully completed. In this case, a standard rule set is requested. If the individual rule set is received from the RSE, the frame, rule set, and parameter set is sent towards the PPE. Because only the parameter set is available in the PPE with the first cycle, a comparison with rule parameter can be done before the proper frame data reaches the PPE.

2) Rule Set Engine: For the Rule Set search, we use a two-stage approach with a hardware-gentle compression method. First, the mapping between flow id and the rule set is done in a sufficiently large SRAM memory. Second, the very large rule sets have to be stored in DDR2-SDRAM. To increase speed when reading and writing memory information, we use self-developed memory controllers. The rule set is sent towards the PCE and forwarded together with the frame and frame parameter set to the PPE.

3) Packet Processing Engine: The PPE is responsible for control and evaluation of the data stream and consists of three central components. In addition to a classic packet filtering, we have implemented a signature recognition and a Web filter. Each of the three components aims at protecting subscribers from unauthorized access from the network side and suppresses attacks from subscribers on the network.

Packet Filtering

The packet filter is divided into several control stages (CS). It controls and evaluates Ethernet frames on OSI layer 2 until

4. Therefore, CS use the first rule in the rule set. Each rule has a type-length-value layout similar to configuration data. If the type of the rule is unequal to the CS' ID, the frame, rule set, and parameter set are forwarded to the next stage. Otherwise, the rule is processed by the CS and removed from the rule set. So, the next CS is able to look at the first position of the rule set. Each CS compares the data from the rule with the data of the parameter set. Because the whole parameter set is available in the first cycle of new data, the lookup increases the processing speed, especially for OSI layer 4 values. In case of a match, the rule action has to be executed. That is, the frame, rule set, and parameter set can be discarded or forwarded. Thereby, it is possible to suppress, e.g., local network shares for the Internet.

Signature Recognition

The signature detection starts after the header of the transport layer. As a basis, we use the snort database. Additionally, we support ISP administrator defined rules. To realize a high speed signature detection, we use bloom filters [1]. Bloom filter are space-efficient probabilistic data structures, which allow for the detection of special signatures in a set of known signatures in a very short time.

Web Filtering

Web filters are a very sensitive issue and have been poorly discussed in the research community. Some countries such as China, the United States, and Great Britain [9] already use Web filtering. These Web filters use external services to compare tagged domain names like the database from Internet Watch Foundation. Our developed solution works in 2 steps. First, we hashes detected domains by CRC64 and search the hash value in a preconfigured binary tree. Second, we verify matches by the onboard DDR2 memory. Thereby, we are able to control traffic in wire speed.

Matches by bloom filters or the web filter can be false positives due to different domains resulting in the same hash value. In the improbable case of a false positive (0.001 %), a match analyzer verifies the possible match at wire speed.

IV. Configuration Software

Via a web interface, customers can set their own filtering rules. Before these rules are applied, they are verified by the ISP. The configuration of the hardware is done by platform independent software developed with QT. The graphical user interface (GUI) consists of a framework, which is able to include so called plugins. Each plugin offers a GUI to configure a separate hardware component of the Secure Access Node. When starting the GUI, the software searches in a special directory for available plugins. All plugins are loaded and appear in the software as a tab. By means of the plugins, ISP provided rule can be generated and customer rules are applied. Furthermore, the configuration software is able to interrupt the hardware processing flow for updating the hardware configuration.

V. Conclusion

Because many subscribers do not have the necessary knowledge to maintain their own security measures, it is important to include security features at the ingress of the network. Therefore, we have designed a software/hardware co-design consisting of a packet filter firewall, a signature detection, and a Web filter module. The implementation results show a reachable speed of 142.9 MHz corresponding to 4.57 GBit/s. Furthermore, subscribers are protected by the Secure Access Node and do not need to care about their own security. Especially for the large number of customers with minor technical knowledge, this is an important feature. Because of the applied methods, the bandwidth of customers is not influenced. Furthermore, no attacker has access to the hardware. Only an ISP administrator is able to update the security mechanism. Moreover, it is possible to update the system during operation. Prospectively, a functional test with real traffic data is intented.

Acknowledgment

We would like to thank the Broadband Access Division of Nokia Siemens Networks in Greifswald, Germany for their inspiration and continued support in this project. This work is partly granted by Nokia Siemens Networks.

References

[1] D. Taylor and J. Turner, "Scalable packet classification using distributed crossproducting of field labels," *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, pp. 269–280, 2004.

[2] A. Guruprasad, P. Pandey, and B. Prashant, "Security features in ethernet switches for access networks, TENCON 2003, Conference on Convergent Technologies for Asia-Pacific Region ," pp. 1211–1214, 2003.

[3] N. S. Networks, "SURPASS hiX 5622/25/30/35 R3.7M System Description IP-DSLAM," *System*, no. 3.

[4] XILINX, "Platform User Guide," *Evaluation*, vol. 347, pp. 1–60, 2009.

[5] G. S. Jedhe, A. Ramamoorthy, and K. Varghese, "A Scalable High Throughput Firewall in FPGA," *16th International Symposium on Field-Programmable Custom Computing Machines*, pp. 43–52, Apr. 2008.

[6] W. Jiang and V. K. Prasanna, "A FPGA-based Parallel Architecture for Scalable High-Speed Packet Classification," *20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 24–31, Jul. 2009.

[7] M. Dixit, B. V. Barbadekar, and A. B. Barbadekar, "Packet classification algorithms," *IEEE International Symposium on Industrial Electronics*, no. ISlE, pp. 1407–1412, Jul. 2009.

[8] A. Kayssi, L. Harik, R. Ferzli, and M. Fawaz, "FPGA-based Internet protocol firewall chip," *ICECS*, pp. 316–319, 2000.

[9] R. Clayton, "Anonymity and traceability in cyberspace," *ACM SIGACT News*, vol. 36, no. 653, pp. 115–148, Nov. 2005.