Putting SHIM6 into Practice

Habib Naderi
Department of Computer Science
University of Auckland
Auckland, New Zealand
e-mail: hnad002@aucklanduni.ac.nz

Brian E. Carpenter
Department of Computer Science
University of Auckland
Auckland, New Zealand
e-mail: brian@cs.auckland.ac.nz

***Abstract*—SHIM6 is a host-centric solution for multihoming in IPv6 networks. It is fully implemented in hosts and does not affect the Internet routing system. We present the results of experiments with SHIM6 in the lab and over the Internet. The aim is to evaluate the behaviour of SHIM6 in case of link failure. We tried two transport protocols, TCP and DCCP, measuring traffic and recovery time, two important performance metrics. In addition, the experiments revealed some operational issues which must be addressed before SHIM6 can be deployed world-wide.**

Keywords*— *IPv6; Multihoming; SHIM6; Performance

I. Introduction

A host or site is multihomed if it has more than one Internet Service Provider (ISP). Multihoming offers a range of benefits, such as failure resistance and load balancing. This is a widely used functionality that enables businesses to tolerate failures and provide continuous service.

Multihoming is provided for current IPv4 Internet users by using Border Gateway Protocol (BGP) features. This is not considered a suitable technique for the future Internet since it cannot scale properly, given the growing demand for multihoming and the huge address space provided by IPv6 [1].

A wide range of solutions has been proposed for this problem. One of those solutions is SHIM6 [2]; it is fully implemented in hosts and does not affect other components of the Internet routing system. SHIM6 allows hosts to be multi-addressed, so a multihomed SHIM6-enabled site does not need to have its own provider independent address block; instead it can use address blocks assigned by its ISPs. This makes address aggregation possible and resolves the scalability problem mentioned above.

SHIM6 inserts a layer between IP and the transport layer to monitor long-lived sessions. Information about each session is stored in a *context* data structure. This is created when a long-lived session is established between two hosts. All upper layer sessions between two hosts may share the same context.

SHIM6 employs an identifier/locator separation scheme. The IP address used to establish a session is called the Upper Layer ID (ULID). A host's other IP addresses play the role of locators. If a path failure occurs, SHIM6 seeks a new path using both hosts' locators. When a new path is found, the SHIM6 layer is activated. It moves the session to the new path by rewriting source and destination addresses in the IP header. All incoming and outgoing packets are inspected and addresses are rewritten, if required. The ULID remains unchanged, which makes the address switching process transparent to the upper layers.

To detect failures and explore new paths, the REAchability Protocol (REAP) is employed [3]. It uses Force Bidirectional Detection (FBD) to detect failures. A valid path requires traffic in both directions, so REAP sends Keep-alive messages when there is nothing to be sent in response to incoming traffic. Thus, outgoing traffic without incoming traffic means that the path is broken. REAP employs timers to implement FBD. When an outgoing packet leaves the host, *send timer* is started and *keep-alive* timer is stopped. When an incoming packet arrives, *send timer* is stopped and *keep-alive* timer is started. If *keep-alive* timer expires, a Keep-alive message is sent. When *send timer* expires, it is considered a path failure and an *Exploration Process* is started to find a new path. First, a list of *address pairs* is created. Each address pair consists of one local and one remote IP address. As a result, this list contains *number of local IP addresses* × *number of remote IP addresses* entries. For example, if each host has 3 addresses, the list will have 9 entries. Then the candidate list is ordered and pruned using address selection rules [4]. To verify reachability of the listed address pairs, REAP starts sending *probe* messages, one probe for each address pair. The list is traversed sequentially. Four *initial probes* are sent 500 ms apart. Then an exponential backoff algorithm increases the interval between probes. Therefore, *exploration time*, the time required to explore a new path, has a direct relation with the number of address pairs. When REAP receives a probe, it realizes that the current path has failed and the other end has started the exploration process, so it triggers the converse exploration process. This finishes when both hosts find a new working path. Then the corresponding SHIM6 context is updated so that future packets use the new path. This process is concealed from the transport layer, which just sees apparent severe congestion.

In this paper, we present the results of an effort to characterize how SHIM6 and REAP perform in real conditions. We conducted a set of experiments in the lab and over the Internet between two multiaddressed SHIM6-enabled hosts, using two transport layer protocols. We chose TCP [5][6] because it is a widely used transport protocol in the Internet. To extend the scope of our study, we chose DCCP [7], a connectionless protocol like UDP, configured with TCP Friendly Rate Control (TFRC) [8] for congestion control. To the best of our knowledge, our experiments over the Internet and using DCCP and TFRC in the experiments are the first attempted using SHIM6 in the IPv6 Internet.

TFRC [8] is a rate-based congestion control mechanism which minimizes quick changes in transfer rate. It reacts more smoothly to bandwidth changes than TCP. It is suitable for unicast flows needing smooth throughput, such as media streaming with a buffering mechanism. In contrast to TCP-like congestion control mechanisms, TFRC gradually reduces its sending rate when congestion is detected. We used DCCP with Congestion Control ID 3 (CCID3) [14], implemented in the Linux kernel, in our experiments. Most media streaming applications follow a DCCP half-connection (HC) scenario, where HC-Sender sends application data and HC-receiver returns acknowledgements at least once per round-trip time. Although DCCP is a connectionless protocol, its header includes a sequence number so the receiver can detect lost packets. Also, the receiver notifies the sender of recent loss intervals. Using this feedback, the sender tries to keep its sending rate suitable for the receiver.

The structure of this paper is as follows: Section II summarizes other work in this area. Section III summarises the configuration of our testbeds in the lab and over the Internet. Results are presented and discussed in section IV. Section V briefly describes the operational issues experienced in the experiments. We conclude our work in section VI.

II. Related Work

S. Barre et al. evaluated REAP performance in the lab [9]. J. Ronan et al. made an empirical evaluation of LinShim6 [15]. A. de la Oliva et al. [10] simulated SHIM6 to study REAP performance and the effect of different values of *send timer* on TCP and UDP application recovery time. L. Xie et al. [11] proposed an improved address exploration method which maintains address switching time at a constant value. H. Naderi et al. performed a performance study on REAP by simulating its behaviour in large scale IPv6 networks [12]. A. de la Oliva et al. presented the results of an analytical study on REAP performance in [13].

III. Configuration of test beds

We set up test environments to run a set of experiments with LinShim6 [17] in the lab and over the Internet. LinShim6 is an implementation of SHIM6, developed at the Université Catholique de Louvain, for Linux.

A. Internet Experiments

Fig. 1 shows configuration of the testbed for Internet experiments. Two SHIM6-enabled multiaddressed hosts in Auckland and Dublin[1] are used. Each host is equipped with two network interface cards and configured with two prefixes from two different ISPs. The SHIM6 host in Auckland is connected to a lab router which is a Linux machine configured as an IPv6 router that simulates link failures. To simulate a failure, all paths except for one are blocked on the lab router, to push SHIM6 to switch the active session to the open path. We developed a client/server program (*failure scheduler*) to schedule failures on the router. The server side of this program runs on the lab router and the client side runs on the SHIM6 host in Auckland. The failure scheduler decides when a failure should happen and which paths should be blocked.

Source Address Dependent Routing (SADR) is necessary for SHIM6. SADR is a mechanism for forwarding packets according to their source addresses. Normally, ISPs expect all packets received from each customer to carry source addresses belonging to its assigned address prefix. Without SADR, packets might be forwarded to the wrong ISP and dropped [16]. Unfortunately, network administrators would not enable SADR on the university edge router. To simulate SADR, the network administrators added static routes to the edge router to forward packets for the host in Dublin through different ISPs according to their destination addresses. Fig. 1 shows the two paths used in the experiments. As a result, only two out of four possible address pairs could work. Packets using the other address pairs would be dropped by the ISP's ingress filters [16]. To minimize the effect of this on our experiments, we changed LinShim6 to shuffle address pairs before starting the exploration process, putting the working address pair in a random location in the list of address pairs. Therefore, the working address pair could appear anywhere in the list and create different cases for the exploration process. The SHIM6 host in Dublin was directly connected to two IPv6 service providers with SADR enabled. This configuration enabled us to run experiments with four address pairs over the Internet.

We developed a client/server program which runs on SHIM6 hosts and is able to generate a stream of 1000 pkt/s of unidirectional traffic using TCP and DCCP protocols. Each experiment starts by running the server side of the application on one SHIM6 host and the client side on the other. The client connects to the server on one of the available paths and sends 256 byte packets to the server in an infinite loop. The server receives and discards them. When communication becomes stable, the failure scheduler blocks the current path. REAP detects the failure and starts the exploration process to find a new working address pair. When a new path is found, the failure scheduler waits for the application to recover and everything to return to normal. Then a similar scenario is repeated with the new path. After each recovery, REAP exploration time (EXPL), the number of sent (SP) and received (RP) probes and application recovery time (ART) are logged. ART is defined as: *Arrival time of the first data packet on the new path - Arrival time of the last data packet on the old path.* Exploration time and probes are measured at the client side (sender) and application recovery time is measured at the server side (receiver). We repeated the above scenario 250 times for each experiment.

B. Lab Experiments

Fig. 3 shows the testbed for lab experiments. Host 1 and Host 2 are two SHIM6-enabled multiaddressed hosts, each equipped with four network interface cards. Two IPv6 routers, similar to the lab router for Internet experiments, connect these hosts. Using this configuration we could run experiments with 4, 9 and 16 address pairs for a clearer view of SHIM6 behaviour in a real environment. To simulate a failure, one

[1] Thanks to John Ronan from Telecommunications Software & Systems Group, Waterford Institute of Technology, who helped us in running these experiments between Auckland and Dublin. He setup a multihomed SHIM6 host in Dublin for these experiments.

instance of the failure scheduler server runs on each router. The client side schedules failures and then asks the server instances to block/unblock paths, using the same application program and scenario as in our Internet experiments.

IV. Results

A. Internet Experiments

Fig. 2 shows the results from two experiments with TCP and DCCP. In all graphs, the X-axis shows the experiment number. REAP exploration time and application recovery time are shown in milliseconds. The title of each graph shows the transport protocol and the measured parameter.

Results show that the number of received probes changes between 1 and 2 when TCP is used, while it is always 1 in case of DCCP. With DCCP, the client side (sender) always starts the exploration process but with TCP, either side might start the process. DCCP employs delayed acknowledgement so it is unlikely that the receiver can detect a failure before the sender. TCP sends regular acknowledgements, so there is an almost equal chance for either sender or receiver to detect the failure and start the exploration process first.

For the DCCP experiments, REAP exploration time, the number of sent probes and application recovery time are bigger than for TCP. The reason is again that for DCCP, the exploration process is started by the client. The server only starts exploration when it receives a probe *exploration* from the client. Therefore, the total exploration time is the sum of the time to reach the working address pair at the sender plus the equivalent time at the receiver. TCP acknowledgements cause the send timer to be activated at the receiver as well, so the failure might be detected by either side. However, the send timer at the other side will also expire soon, so the exploration process at both sides is started with a small time difference. In other words, the sender and receiver perform exploration in parallel. Thus, the exploration time is equal to the maximum exploration time at both sides. As a result, the exploration time, and therefore the number of probes and application recovery time, are bigger for DCCP. The same application can make REAP behave differently by using different transport protocols. In other words, not only the behaviour of the application but also that of the transport layer protocol can affect REAP behaviour, recovery time and generated traffic.

We also observed some signs of probe loss in the results. Any run with more than six probes, including four initial probes and one probe *operational* at the end and one more probe *exploration* that might be sent before receiving the response from the other side, can potentially be a sign of probe loss. DCCP results show only one instance of this, but more instances can be observed in TCP results. If packet loss causes the exploration process to go to the second round, it seriously affects recovery time because of the exponential backoff algorithm which is employed after the first run.

B. Lab Experiments

Fig. 4 and 5 show the results from similar experiments, with 4 and 9 address pairs, in the lab. The graphs are as before, except that the title of each graph is a triple in the form: <transport-protocol> - <measured-parameter> - <number-of-address-pairs>. We also tried experiments with 16 address pairs, but they failed when the working address pair was at or close to the end of the REAP candidate list. REAP employs exponential backoff after sending initial probes to avoid a signalling storm, which leads to a long exploration time when there is a long candidate list. For 16 address pairs, this delay causes the connection to time out and fail. In some cases, packet losses cause the exploration process to go to the second round of exploration which also causes long delays and makes SHIM6 decide to stop exploration and remove the context. Therefore, we only include results from experiments with 4 and 9 address pairs.

DCCP results show a different behaviour from the Internet experiments. In the lab, TCP and DCCP show a similar behaviour because RTT in the lab experiments is small (0.3 ms), so DCCP sends acknowledgements faster than in the Internet experiments. Therefore, employing delayed acknowledgement does not make a noticeable difference.

Probe losses can be observed in the lab experiments too. They are the main reason for differences between average values for DCCP and TCP. By removing runs with probe loss from the results and calculating the average values again, we get similar values for TCP and DCCP experiments in the lab.

V. Operational issues

Running experiments in the lab and over the Internet uncovered some issues which affect using SHIM6 in practice:

- *Firewall:* Some firewalls drop IPv6 packets if they do not recognize their extension headers. SHIM6 adds its own extension header to the packets, which seems to be unknown to some commercial firewall software. We experienced this problem in the Internet experiments.
- *Filter precedence:* This is an issue specific to LinShim6. iptables filters process packets before LinShim6. It means that iptables filters can only see ULIDs and not real source and destination addresses. Thus, when SHIM6 changes source/destination addresses in the packets, iptables filters are not able to recognize them.
- *New path employment:* When a new path is discovered after a failure, the transport layer is notified but in some cases it does not start using the new path for a reason we could not determine. It causes the application to stay in the wait state. Producing traffic on the new path, e.g. by sending ICMP packets, resolves the issue but it still has its negative impact on the application recovery time.
- *Source Address Dependent Routing (SADR)*: SADR is necessary for effective use of SHIM6. SADR needs routers to keep one routing table per source address prefix in memory. It also imposes some processing overhead on the routers. We found it difficult to convince site administrators to enable this on edge routers. Without SADR, packets are forwarded based on their destination address only and might go to the wrong ISP. This can seriously affect the REAP exploration process if REAP probes are forwarded to the wrong provider and dropped.

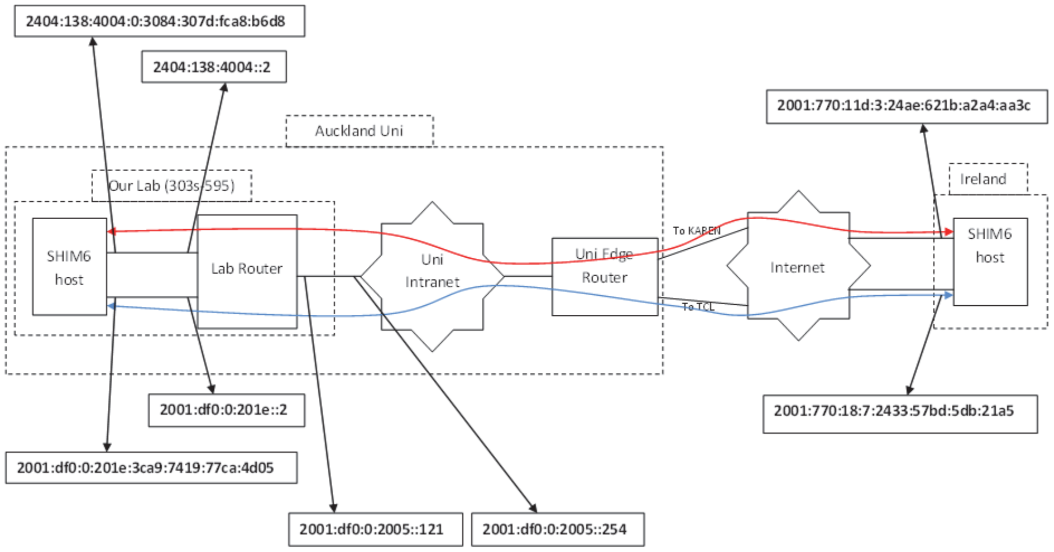

Fig. 1. Test bed for experiments with SHIM6 over the Internet between Auckland and Dublin. SADR is not enabled on the university edge router, so only two paths are available.

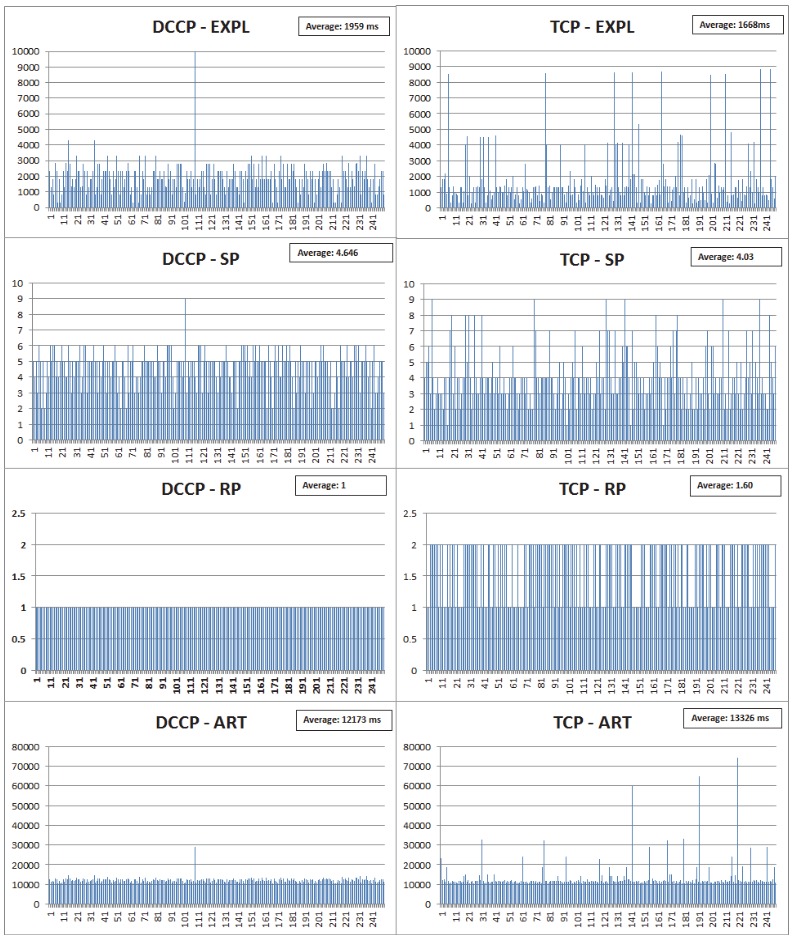

Fig. 2. Results from the experiments with DCCP and TCP over the Internet with four address pairs. X-axis shows the experiment number and Y-axis shows the value for the measured parameter. Times are in milliseconds.

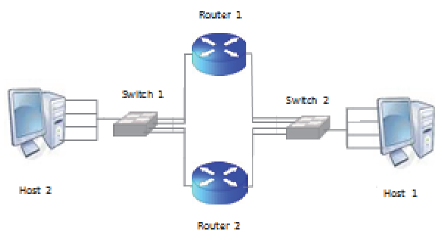

Fig. 3. Testbed for experiments with SHIM6 in the Lab with two SHIM6-enabled hosts each equipped with four network cards.

VI. Conclusion

This paper presents the results of a set of experiments in the lab and over the Internet with SHIM6. They showed that not only behaviour of the application but also behaviour of the transport layer protocol can affect the behaviour of REAP and, therefore, recovery time and generated traffic. They also uncovered some issues for practical use of SHIM6. Two main issues that affect deployment of SHIM6 are: 1) some firewalls do not recognize and, therefore, drop SHIM6 packets and 2) it is hard to convince network administrators to enable source address dependent routing on the site's edge routers, due to its performance penalties. The experiments also showed that the REAP exploration process is not able to handle failures effectively when there are more than 9 address pairs.

ACKNOWLEDGEMENT

Some material in this paper has been adapted from H. Naderi's doctoral thesis.

REFERENCES

[1] J. Abley, B. Black and V. Gill, Goals for IPv6 Site-Multihoming Architectures, Internet RFC 3582, 2003.

[2] E. Nordmark, M. Bagnulo, Shim6: level 3 Multihoming Shim Protocol for IPv6, Internet RFC 5533, June 2009.

[3] J. Arkko, I. van Beijnum, Failure Detection and Locator Pair Exploration Protocol for IPv6, Internet RFC 5534, June 2009.

[4] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default address selection for Internet Protocol Version 6 (IPv6). Internet RFC 6724, 2012.

[5] J. Postel, Transmission Control Protocol, Internet RFC 793, 1981.

[6] M. Allman, V. Paxson, E. Blanton, TCP Congestion Control, Internet RFC 5681, 2009.

[7] E.Kohler, M. Handley, S. Floyd, Datagram Congestion Control Protocol (DCCP), Internet RFC 4340, 2006.

[8] S. Floyd, M. Handley, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, Internet RFC 5348, 2008.

[9] S. Barre, O. Bonaventure, Improved Path Exploration in shim6-based Multihoming, SIGCOMM 2007 Workshop "IPv6 and the Future of the Internet", 2007.

[10] A. de la Oliva, M. Bagnulo, A. Garcia-Martinez, I. Soto, Performance Analysis of the REAchability Protocol for IPv6 Multihoming, NEW2AN '07 Proceedings of the 7th international conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, 2007.

[11] L. Xie, P. Hu, J. Bi, An Enhanced Shim6 Address Switching Method, Fifth International Conference on Computational Science and Applications, 2007.

[12] H. Naderi, B. Carpenter, A Performance Study on REAchability Protocol in Large Scale IPv6 Networks, 2nd International Conference on Computers and Networks (ICCNT 2010), 2010.

[13] A. de la Oliva, I.soto, A. Garcia-Martinez, M. Bagnulo, A. Azcorra, Analytical characterization of failure recovery in REAP, Computer Communications, Volume 33, Issue 4, pp. 485-499, 2010.

[14] S. Floyd, E. Kohler, J. Padhye, Profile for Datagram Congestion Control Protocol (DCCP) Congestion control ID 3: TCP Friendly Rate Control, Internet RFC 4342, 2008.

[15] J. Ronan, J. McLaughlin, An Empirical Evaluation of a Shim6 Implementation, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer Berlin Heidelberg, 2011.

[16] P. Ferguson, D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. Internet RFC 2827, 2000.

[17] S. Barre, J. Ronan, and O. Bonaventure. Implementation and evaluation of the shim6 protocol in the linux kernel. Computer Communications, 34(14):1685 –1695, 2011.

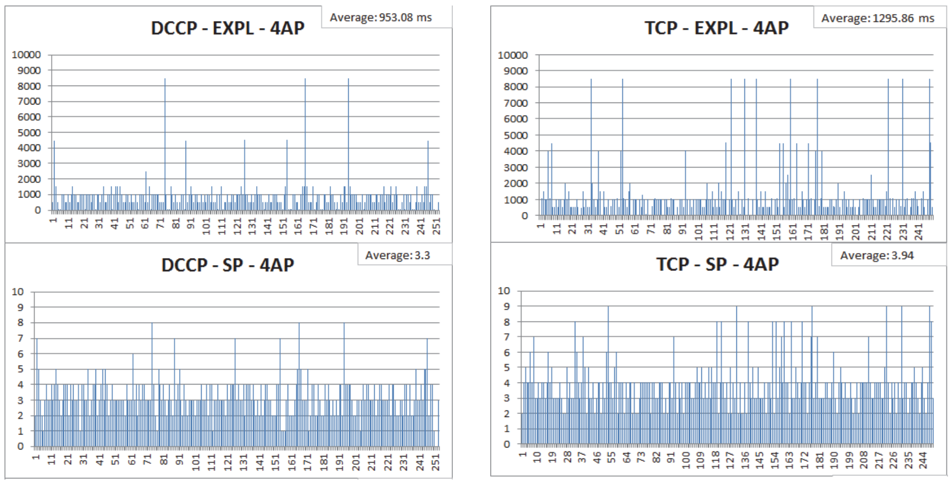

Fig. 4. Continued on next page

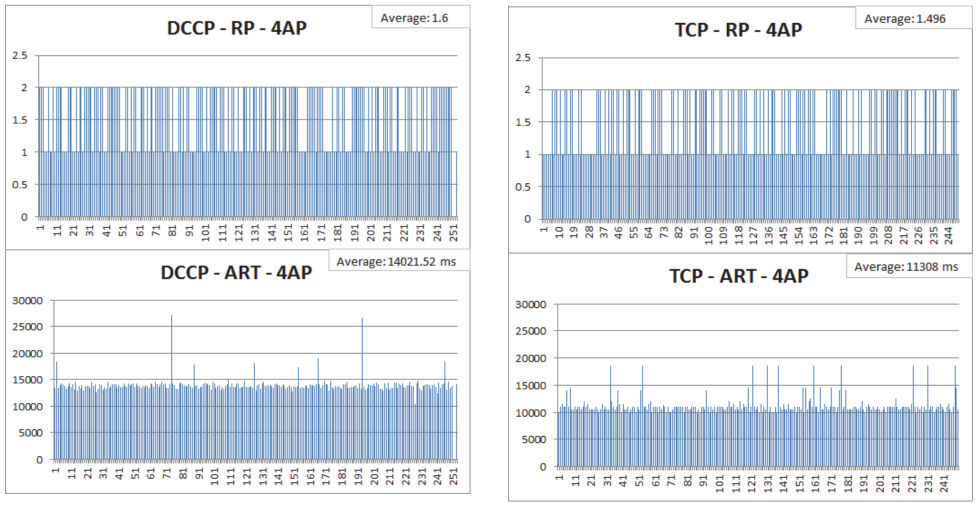

Fig. 4. Results from lab experiments for TCP and DCCP with four address pairs. X-axis shows the experiment number and Y-axis shows the value for the measured parameter. Times are in milliseconds.

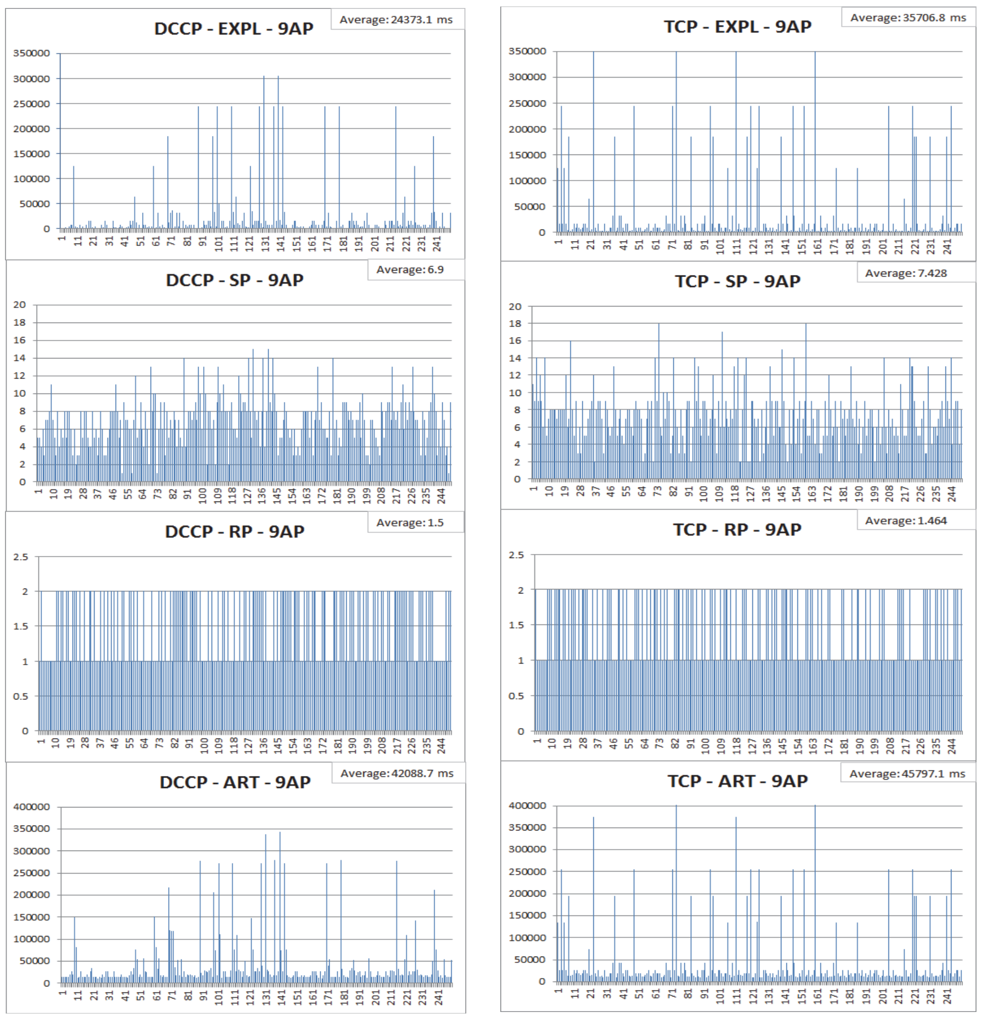

Fig. 5. Results from lab experiments for TCP and DCCP with nine address pairs. X-axis shows the experiment number and Y-axis shows the value for the measured parameter. Times are in milliseconds.