Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 24, 2015

H. Naderi
B. Carpenter, Ed.
Univ. of Auckland

April 22, 2015

Experience with IPv6 path probing
draft-naderi-ipv6-probing-01

Abstract

This document reports on experience and simulations of dynamic probing of alternate paths between two IPv6 hosts when network failures occur. Two models for such probing were investigated: the SHIM6 REAchability Protocol (REAP) and the Multipath Transmission Control Protocol (MPTCP). The motivation for this document is to identify some aspects of path probing at large or very large scale that may be broadly relevant to future protocol design.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 24, 2015.

Table of Contents

1. Introduction

A common situation in the Internet today is that a host trying to contact another host has a choice of IP addresses for one or both ends of the communication. Multiple addresses are expected to be quite common for IPv6 hosts [RFC2460]. Some approaches to this situation envisage either switching paths during the course of the communication or using multiple paths in parallel. Examples include "Happy Eyeballs" [RFC6555] which tries alternative paths at the start, SHIM6 [RFC5533] and Stream Control Transmission Protocol (SCTP) [RFC4960] which change paths when there is a failure, and Multipath TCP (MPTCP) [RFC6824] which shares the paths dynamically.

Some of these methods involve active path probing to choose the best one. SHIM6 probes all available paths using the REAchability Protocol (REAP) [RFC5534] when the current path fails, and MPTCP effectively probes all paths continuously, and shifts load according to the results. In this document we summarise results and observations from SHIM6 and MPTCP operated or simulated at large scale. These observations may be of help in designing future path probing mechanisms. In particular, we are interested in minimising both the time taken to recover to the maximum possible throughput after a path failure, and the amount of overhead traffic caused by the probing process.

In summary, we ran a series of SHIM6 experiments, each including 250 path failures, between Auckland and Dublin, measuring the time and overhead traffic for each instance of path probing and recovery.

Then we repeated essentially the same experiment in the laboratory in Auckland (i.e., with negligible RTT instead of round-the-world RTT). Then we built a Stochastic Activity Network (SAN) simulation model of the same scenarios, and validated it by comparison with the experimental results. Finally we used this model to simulate path failure and recovery using REAP at very large scale (10,000 simultaneous sessions on a single site experiencing path failure). Both TCP and DCCP [RFC4340] were used for the transport layer, with a simple application sending meaningless data in one direction only.

This was followed by roughly equivalent simulations of recovery from path failure for MPTCP sessions. In this case we validated the SAN model by comparison with a completely different MPTCP simulator developed elsewhere [Wischik10].

One advantage of the SAN model is that there are SAN analysis software tools which allow very large scale simulations. Another is that it makes it relatively easy to experiment with variations of the protocol itself, so we did test the impact of certain protocol changes. However, unlike conventional network simulation tools, the user has to program a complete protocol behaviour model. We used the Moebius tool [Moebius].

Details of the experiments and results have been described in two papers [Naderi10] [Naderi14b] and in H. Naderi's thesis [Naderi14a]. This document limits itself to outlining the results and their implications for the design of path probing mechanisms in the Internet.

2. Results for SHIM6 and REAP

2.1. Experiments over the Internet

We set up a test environment which enabled us to run a set of experiments over the Internet with the LinShim6 implementation of SHIM6 [Barre08]. We have used two SHIM6-enabled multi- addressed hosts, located in the University of Auckland (New Zealand) and Waterford Institute of Technology (Dublin, Ireland). Each host was equipped with two network interface cards and configured with two prefixes from two different providers. The SHIM6 host in Auckland was connected to a router which was a Linux machine and was configured as an IPv6 router. This router simulated link failures for the experiments.

Source Address Dependent Routing (SADR) is necessary for effective use of SHIM6. Hosts decide what source and destination address to use when host-centric solutions, like SHIM6, are used. Without SADR, or similar mechanism for routing, packets might be forwarded to the

wrong address providers and dropped because of ingress filtering according to BCP 38 [RFC2827] [RFC3704]. Unfortunately, we could not convince the university network administrators to enable SADR on the Auckland University edge router. To run the experiments, they agreed to add static routes to the edge router's routing table, to forward packets destined to the host in Dublin through different providers according to their destination addresses. Therefore, only two address pairs out of four possible address pairs could work. To resolve this issue, we have changed LinShim6 to shuffle the list of address pairs before starting the exploration process in order to put the working address pair in a random location in the list. As a result, the working address pair could appear in any location in the list and thus create different recovery cases.

This configuration enabled us to run experiments with four address pairs over the Internet. For each experiment, we artificially created 250 failures and for each case measured the REAP exploration time (EP), number of sent (SP) and received probes (RP) and application recovery time (ART).

Comparing results from experiments with TCP and DCCP shows that when DCCP is employed, EP, SP and RP are bigger than when TCP is used. The main reason for this is that DCCP employs delayed acknowledgement. It sends ACKs every RTT (300 ms), while in case of TCP, they are sent more frequently (less than 100 ms apart). Since the RTT is long, the communications look different from REAP's view point although the behaviour of the application is the same in both experiments. Since TCP sends ACKs faster, REAP treats it more like a bi-directional communication while DCCP communication is treated more like uni-directional. As a result, in the DCCP experiment, the sender always detects the failure first and then reports it to the receiver, while in the TCP experiment both sides detect failure and start exploration almost at the same time. In other words, in case of TCP, exploration is performed in parallel on both sides and takes less time and generates less traffic. This result also shows that the efficiency of the solutions, like SHIM6, which are implemented inside the protocol stack may be affected by the behaviour of the other layers of the protocol stack as well.

We also observed some signs of probe loss in the results. Probe losses can affect EP, SP, RP and ART. When a probe is lost, it might cause the exploration process to go to a second round, and then an exponential backoff algorithm causes the exploration process to take longer and generate more traffic.

2.2. Lab Experiments

We repeated similar experiments in the lab. The main difference was RTT which was much smaller (0.3 ms) than in the Internet experiments. We setup two SHIM6 hosts in the lab, each equipped with four network interfaces. Thus, in addition to experiments with four address pairs (similar to the Internet experiments), we could run experiments with 9 and 16 address pairs as well.

In the lab, we got similar results from the TCP and DCCP experiments. Since RTT is small, DCCP sends ACKs faster, and therefore there is no difference from REAP's viewpoint.

Probe losses are observable in the lab experiments too. Probe loss causes REAP to go to the second round for scanning the list of address pairs, which leads to sending more probes and also longer exploration time.

Experiments with 16 address pairs fail when the working address pair is located at or close to the end of the list of address pairs. REAP employs exponential backoff after sending its initial probes, to avoid generating large bursts of traffic during exploration. For 16 address pairs, this delay sometimes causes the connection to time out and stop the experiment. In some cases, SHIM6 removes the context without finding the new address pair. In such cases it seems that packet losses cause the exploration process to go to the second round of exploration and the resulting longer delays cause SHIM6 to actually stop exploration and remove the context.

2.3. Large scale simulation

To study the behaviour of REAP in a very large scale network (e.g., an enterprise network), we built a simulation model of REAP and conducted some experiments which simulated a link failure event in a network with 10,000 simultaneously active SHIM6-monitored communications. The aim of the experiments was to see how REAP reacts to path failures in a large SHIM6-enabled multihomed network. In our practical tests, nine address pairs seems to be the limit but we have included larger numbers in our simulations to obtain a clearer view of REAP's behaviour.

We focused on REAP recovery time and probe traffic as two important performance parameters. REAP recovery time is the time that REAP takes to detect the failure and find a new working address pair. REAP traffic is the traffic which is generated by REAP itself during its exploration process.

We measured average and total REAP recovery time for different numbers of address pairs for 10,000 instances of REAP. We define total REAP recovery time as the recovery time for the whole site, i.e., the time between failure occurrence and recovering the last context. In other words, it shows the recovery time for the last context that is recovered. The average recovery time is calculated by dividing the sum of recovery times for REAP instances by the number of REAP instances. It should be noted that recovery time includes failure detection and address exploration times.

A typical average recovery time for 4 address pairs is 10 to 12 seconds. The results show that the average and maximum recovery time increase when the number of address pairs is increased. The correlation is not linear because REAP uses an exponential backoff algorithm for increasing the time interval between probes. As a result, REAP shows poor performance when the number of address pairs exceeds 9, for example exceeding 100 seconds to recover with 16 address pairs.

We also measured the average and total number of probes sent during the address exploration process in the experiments. The results show that there is a linear correlation between number of address pairs and number of sent probes. They also show that a large quantity of probes is sent at the start of exploration. For example, in the case of four address pairs, 93% of the probes, and in the case of 25 address pairs 34% of probes, are sent during the first 10 seconds. The reason is that all contexts detect failure within 10 seconds and start exploration by sending initial probes (the first four probes, which are sent in two seconds). After that, there are some intervals when very few probes are sent. This can be seen more clearly in the experiments with more address pairs, e.g. 16 or 25 address pairs. This means that for some SHIM6 contexts the time interval between probes is large, because of the exponential backoff, so REAP instances have to wait for a long time before probing the next address pair. Some connections might be dropped by the transport or application layer before REAP can recover them. For example, in case of 25 address pairs, 50% of contexts need more than five minutes to recover.

Although the peak of the REAP traffic is generated in the first 10 seconds (before employing the exponential backoff algorithm), our results show that this traffic is small compared to normal traffic for a large network, and cannot cause a major problem. For example, in the case of 25 address pairs, about 4800 probes per second are sent during the first 10 seconds of the exploration process, which is the peak of the traffic. Every probe in the first 10 seconds carries at most seven address pairs; four initial address pairs and three more after employing exponential backoff. Thus, the average probe

size in the first 10 seconds is 232 bytes; each probe needs 72 bytes for the fixed part and 40 bytes for each address pair. As a result, a load of 4800 probes per second does not occupy more than one MB/s of the site's available link capacity. Large sites usually have high bandwidth links to the Internet and this amount of traffic does not cause a significant problem for them. In any case this traffic will occur at a time when normal traffic from the same sessions has been interrupted.

We also tried two changes to REAP to improve recovery time: Increasing the number of initial probes, and sending initial probes in parallel. In both cases, we also measured the probe traffic. The results showed that those modifications improved recovery time while their effect on the traffic were not big. For example, in case of nine address pairs, increasing the number of initial probes from four to five caused about 6.5% increase in traffic in the first 10 seconds of the recovery process, 22% decrease in average recovery time and 34% decrease in maximum recovery time. Sending initial probes in parallel, in the case of nine address pairs, caused an 11% decrease in average recovery time, 4.5% decrease in maximum recovery time, and 8.2% increase in traffic. In both cases, these modifications increased traffic but not to the level that could not be handled in a large network.

3. Results for MPTCP

MPTCP does not use any specific mechanism for probing paths. In fact, every subflow runs as a TCP flow and it is the TCP congestion control mechanism which monitors the used path. When congestion is detected, the load from the congested path is transferred to other available paths, if they present less congestion. The MPCTP congestion control algorithm, known as SEMICOUPLED, reacts to congestion reports from subflows and adjusts the load on the used paths to achieve performance and fairness. TCP never sets the congestion window for a subflow to less than 1. Therefore, even on a highly congested path or a broken path, it performs the equivalent of probing by setting the congestion window size to 1, so that any improvements in the path can be detected. Expiration of the TCP retransmission timer for the subflow on a broken path triggers sending a segment once in a while, acting as a probe, to ensure a recovery in the path can be detected. How fast this mechanism can detect an improvement in a broken path depends on the value of the time-out for this timer (RTO). The minimum value is usually set to 1 second and consequent expirations, the case for a broken path, back off the timer value and multiplies RTO by 2. The traffic generated by this mechanism in this case is low and may be handled easily, even in a large network.

We simulated MPTCP with up to 8 paths and with RTTs between 80 and 150 ms, observing the expected behaviour, with the load in the steady state spread across the paths. When the loss rate of a path is higher, the throughput of that path is lower. For a given loss rate, a smaller RTT increases throughput on that path. However, total throughput increases sublinearly with more paths, due to the way SEMICOUPLED links the congestion windows of the various subflows. For example, we simulated a scenario in which the steady state throughput for 8 paths was only about 25% greater than for a single path (Figure 5.10 in [Naderi14a]). This suggests that a scenario with as many as 8 paths is of limited value in a reasonably reliable network.

We simulated a permanent failure of a single path in a scenario with four paths in operation. As may be deduced from the previous point, the throughput recovered in the steady state to within a small percentage of its previous value. This recovery took about 6 seconds (Figure 5.15 in [Naderi14a]), which is significantly faster than observed with SHIM6 due to MPTCP's effectively continuous probing. Simulations of temporary path failures showed that returning to the original steady state using all paths took a similar time.

Finally we simulated the effect of variable loss rates on MPTCP performance with two paths operating. We observed that for loss rates varying randomly in the range up to 1%, MPTCP effectively maintains its steady state throughput.

4. Operational issues

Many if not most site border firewalls today drop packets containing the SHIM6 extension header. In our Internet experiments we had to bypass the site firewall at both ends. This issue is discussed in [RFC7045].

Source Address Dependent Routing (SADR) is necessary for effective use of multiple paths. Without it, packets may be sent to the wrong exit router, or to an ISP that will immediately discard them due to ingress filtering. With ingress filtering in place, packets with a given source address may only be sent via an ISP that accepts packets from that source address. If this is not taken correctly into account by the source host and by the local routing configuration, the host will waste resources trying to explore paths that are certain to fail.

5. Implications for future designs

We suggest several conclusions from the above results that should be relevant to the design of any probing mechanism for exploiting alternative paths between two hosts:

- The interaction between round-trip time, the transport layer acknowledgement mechanism, and the failure detection mechanism is quite subtle and significantly affects the time taken to start recovery after a failure.

- When probing is linked to congestion control, packet loss rates may also affect recovery times.

- Probe traffic is unlikely to cause overload, especially since normal traffic stops during recovery from failure.

- Exponential backoff leads to significantly slower recovery time, and (due to the previous point) is probably unnecessary.

- Probing all alternative paths in parallel leads to significantly faster recovery times with only a minor increase in the intensity of probe traffic, although this does occur on the paths that are still carrying normal traffic. However, full sized probe packets (as used by MPTCP, because they are normal data packets) have more impact than short probe packets (as used by SHIM6).

- The probe packets should resemble normal data packets as much as possible, in order to avoid being treated specially or dropped by middleboxes such as firewalls or load balancers.

- If Source Address Dependent Routing (SADR) is unavailable, it is better to avoid probing address pairs that will fail as a result. (Probing all paths in parallel would in fact mask this problem.)

- There is little to be gained by having more than two or three alternative paths.

6. Security Considerations

Apart from the need for SHIM6 to bypass firewalls, no security issues were identified during this work.

7. IANA Considerations

This document requests no action by IANA.

8. Acknowledgements

This document was produced using the xml2rfc tool [RFC2629].

Some text was adapted from [Naderi14a].

John Ronan from the Telecommunications Software and Systems Group, Waterford Institute of Technology, and the University of Auckland Information Technology Services (ITS) helped to run the SHIM6 experiments over the Internet between Auckland and Dublin.

9. Change log [RFC Editor: Please remove]

draft-naderi-ipv6-probing-01: editorial improvements, 2015-04-22.

draft-naderi-ipv6-probing-00: original version, 2014-10-21.

10. Informative References

[Barre08] Barre, S., "LinShim6 - implementation of the Shim6 protocol", Technical Report, Universite catholique de Louvain , February 2008.

[Moebius] Deavours, D., Clark, G., Courtney, T., Daly, D., Derisavi, S., Doyle, J., Sanders, W., and P. Webster, "The Moebius framework and its implementation", IEEE Transactions on Software Engineering 28(10):956-969, October 2002.

[Naderi10] Naderi, H. and B. Carpenter, "A Performance Study on REAchability Protocol in Large Scale IPv6 Networks", Second International Conference on Computer and Network Technology (ICCNT 2010), Bangkok 28-32, April 2010.

[Naderi14a] Naderi, H., "Evaluating and Improving SHIM6 and MPTCP: Two Solutions for IPv6 Multihoming", Ph.D. Thesis, The University of Auckland , July 2014.

[Naderi14b] Naderi, H. and B. Carpenter, "Putting SHIM6 into Practice", Australasian Telecommunication Networks and Applications Conference (ATNAC 2014), Melbourne , November 2014.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

[RFC5534] Arkko, J. and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", RFC 5534, June 2009.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, December 2013.

[Wischik10] Wischik, D., Raiciu, C., and M. Handley, "Balancing resource pooling and equipoise in multipath transport", 8th USENIX Symposium on Networked Systems Design and Implementation, San Jose , April 2010.

Authors' Addresses

Habib Naderi
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: habib@cs.auckland.ac.nz

Brian Carpenter (editor)
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com