Revisiting Next-hop Selection in Multipath Networks

Simon van der Linden
simon.vanderlinden@uclouvain.be

Gregory Detal
gregory.detal@uclouvain.be

Olivier Bonaventure
olivier.bonaventure@uclouvain.be

ICTEAM Institute
Université catholique de Louvain
Louvain-la-Neuve, Belgium

ABSTRACT

Multipath routing strategies such as Equal-Cost MultiPath (ECMP) are widely used in IP and data-center networks. Most current methods to balance packets over the multiple next hops toward the destination base their decision on a hash computed over selected fields of the packet headers. Because of the non-invertible nature of hash functions, it is hard to determine the values of those fields so as to make the packet follow a specific path in the network. However, several applications might benefit from being able to choose such a path. Therefore, we propose a novel next-hop selection method based on an invertible function. By encoding the selection of successive routers into common fields of packet headers, the proposed method enables end hosts to force their packets to follow a specific path.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Routers*

General Terms

Algorithms, Management, Performance

Keywords

multipath, load balancing, path selection

1. INTRODUCTION

Load balancing allows to improve the performance and the scalability of the Internet by distributing the load evenly across network links, servers, or other resources, in order to maximize the throughput, achieve redundant connectivity, obtain an optimal resource utilization, or avoid congestion. Different forms of load balancing are deployed at various layers of the protocol stack. At the network layer, multipath routing strategies such as ECMP are widely deployed.

Most multipath routers use a hash computed over selected fields of the packet headers to select a next hop. Because of the avalanche effect hash functions exhibit, a fair distribution of the packets over the next hops is ensured even though the values of their header fields fed to the hash function might not vary widely[2]. By carefully selecting these fields, next-hop selection methods avoid distributing the packets of a transport-level flow over more than one next hop in order to prevent packet reordering at the destination.

Due to the design of hash functions, it is hard to find the values of these fields in order to force a next hop to be selected at a multipath router. It is even harder to find them to control the decisions of multiple routers in a row, i.e., to force a packet to follow a specific path.

Hash-based next-hop selection methods bring two challenges that could be relieved by letting end hosts possibly control the selection of next hops at multipath routers.

First, it is hard to monitor paths in multipath networks to test and verify the performance and quickly detect problems. The simplest monitoring tool, traceroute, gives erroneous measurements [1]. Paris traceroute fixes most of the problems traceroute experienced with multipath networks, and is able to discover various paths between the source and a destination. Unfortunately, its probabilistic approach causes it to send quite a large amount of probes and leaves undiscovered paths when path diversity increases.

Second, it is difficult for end hosts to use disjoint paths in the network to get better performance and resiliency. There has recently been a growing interest in being able to use multiple paths in transport protocols. While it was initially focused on multi-homed hosts, single-homed hosts could also benefit from multiple paths in the core network. To leverage on multipath with hash-based next-hop selection methods, end hosts can barely do better than creating enough flows in the hope that they will be bound to different paths [3].

For these applications, solutions like source routing or a shim header with routing information are impractical. Monitoring packets should not be handled differently than regular packets. Protocol changes should be avoided to keep compatibility with legacy routers on the path and allow for incremental deployment.

In this paper, we propose a novel next-hop selection method that allows to determine the ports to use for a TCP or UDP flow to choose the path the latter will follow. Compared to a modulo-N hash, the most deployed next-hop selection method, the proposed method mainly differs in two points. First, instead of using a hash function, we use an invertible function, e.g.., a block cipher. Second, we use a different part of the output of that function at each router. Notwithstanding, with ordinary traffic, modulo-N hashes and our method achieve equally-fair distributions.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

2. DESIGN

We propose to apply a function over selected fields of the packet header – the source and destination IP address, the protocol identifier, and the source and destination TCP or UDP ports – to pick one next hop among the multiple available at each router along the path to the destination. More precisely, an invertible function F is applied over the source and destination ports, while the rest is fed to a injective function H. F and H should exhibit the avalanche effect to keep the distribution of their output as uniform as possible even if the input is not. We use block ciphers such as RC5 and KATAN for F, and CRC, as in many implementations of hash-based methods, for H. The output of both functions is *xor*-ed to form a 32-bit value that we call path selector. Albeit the computed path selector is the same at each router on the path, routers pick a different part of it according to the Time-to-Live (TTL) of the packet. This part can serve as a finger into a table of next hops to select one to relay the packet. We call it the next-hop selector.

Using this method, the end host can determine the ports to use to make its flow follow a specific path. The next-hop selectors of all routers on the path are concatenated into a 32-bit value and sorted according to the initial TTL to form a path selector. The latter is then *xor*-ed with the output of H applied over the source and destination IP addresses, and the protocol identifier. The inverse function F^{-1} is applied over the resulting value. The latter value provides the source and destination ports to use.

If the end hosts do not have enough knowledge of the network topology to construct a useful path selector, the construction could be relegated to a dedicated service. Such a service could be offered to end hosts by a server that gathers information about the topology, e.g., by listening to link-state advertisements.

This next-hop selection method can be used to implement a monitoring tool that deterministically probes any particular path or all paths between a source and a destination. Another example enables hosts using a multipath-aware transport protocol like Multipath TCP [3] to spread flows over disjoint paths. Indeed, while the destination port of the first TCP flow can hardly be chosen, the ports of the subflows are not actually used. They can be freely set in order to make the subflows take different paths than the first one.

Albeit only the ports are controlled and the lattitude of choice their 32 bits offer is limited, our analysis of ISP and data-center networks has shown it should be enough to implement the aforementioned applications in these networks. We are investigating ways to get more bits to control and to use them more efficiently. We are also looking at ways to control the path beyond the network of the packet sender.

3. PRELIMINARY EVALUATION

We implemented the proposed method in a module for the Click modular router. We implemented the Skip32, RC5, and KATAN block ciphers as invertible functions. We also implemented a monitoring tool to validate the ability to control the path the proposed method offers. Due to space limitation, we are not going to evaluate the tool here.

We then analyze how the proposed method preserves a fair distribution of packets at each router, compared to a modulo-2 hash. Therefore, we replay a CAIDA trace [4] of 10 million of packets. Fig. 1 shows the difference in packet distribution over two next hops for the modulo-2 hash and the proposed method with RC5. We observe two things. First, as the maximum difference value never reaches 3%, the distribution is almost equal. Second, the variation around the optimum is in most cases smaller than 1%. Fig. 2 shows the packet distribuition over time over two next hops. Both the modulo-2 hash and the proposed method slightly fluctuate within the same tight interval $[31\%, 36\%]$ and their median is close to 33,3%. We observe that there is no strong difference between the two methods.

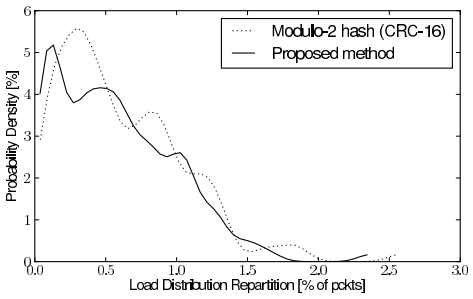
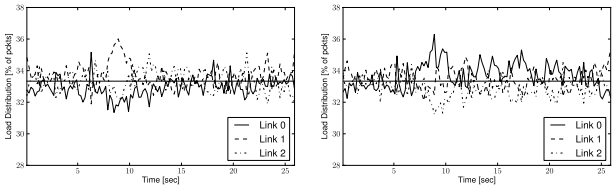


Figure 1: The difference in packet distribution over two next hops is comparable to the distribution offered by the modulo-2 hash.



(a) Modulo-3 hash (CRC-16) (b) Proposed method

Figure 2: The packet distribution over three next hops is similar over time too.

Thus, the ability to control path offered by the proposed method does not significantly impact the fairness of the local packet distribution. Nevertheless, such a fair distribution might not be preserved if end hosts somehow control the next-hop selection.

4. ACKNOWLEDGMENT

Simon van der linden is supported by a FRIA grant. This work has been partially funded by the FP7 EU project CHANGE.

5. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proc. ACM SIGCOMM IMC*, 2006.

[2] Z. Cao, Z. Wang, and E. Zegura. Performance of Hashing-Based Schemes for Internet Load Balancing. In *Proc. IEEE INFOCOM*, 2000.

[3] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley. Data Center Networking with Multipath TCP. In *Proc. ACM SIGCOMM HotNets Workshop*, 2010.

[4] C. Shannon, E. Aben, K. Claffy, and D. Andersen. The CAIDA Anonymized 2008 Internet Traces – 2008-07-17 12:59:07 - 2008-07-17 14:01:00.