

Analysis of an Equal-Cost Multi-Path Algorithm

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

Equal-cost multi-path (ECMP) is a routing technique for routing packets along multiple paths of equal cost. The forwarding engine identifies paths by next-hop. When forwarding a packet the router must decide which next-hop (path) to use. This document gives an analysis of one method for making that decision. The analysis includes the performance of the algorithm and the disruption caused by changes to the set of next-hops.

1. Hash-Threshold

One method for determining which next-hop to use when routing with ECMP can be called hash-threshold. The router first selects a key by performing a hash (e.g., CRC16) over the packet header fields that identify a flow. The N next-hops have been assigned unique regions in the key space. The router uses the key to determine which region and thus which next-hop to use.

As an example of hash-threshold, upon receiving a packet the router performs a CRC16 on the packet's header fields that define the flow (e.g., the source and destination fields of the packet), this is the key. Say for this destination there are 4 next-hops to choose from. Each next-hop is assigned a region in 16 bit space (the key space). For equal usage the router may have chosen to divide it up evenly so each region is $65536/4$ or 16k large. The next-hop is chosen by determining which region contains the key (i.e., the CRC result).

2. Analysis

There are a few concerns when choosing an algorithm for deciding which next-hop to use. One is performance, the computational requirements to run the algorithm. Another is disruption (i.e., the changing of which path a flow uses). Balancing is a third concern; however, since the algorithm's balancing characteristics are directly related to the chosen hash function this analysis does not treat this concern in depth.

For this analysis we will assume regions of equal size. If the output of the hash function is uniformly distributed the distribution of flows amongst paths will also be uniform, and so the algorithm will properly implement ECMP. One can implement non-equal-cost multi-path routing by using regions of unequal size; however, non-equal-cost multi-path routing is outside the scope of this document.

2.1. Performance

The performance of the hash-threshold algorithm can be broken down into three parts: selection of regions for the next-hops, obtaining the key and comparing the key to the regions to decide which next-hop to use.

The algorithm doesn't specify the hash function used to obtain the key. Its performance in this area will be exactly the performance of the hash function. It is presumed that if this calculation proves to be a concern it can be done in hardware parallel to other operations that need to complete before deciding which next-hop to use.

Since regions are restricted to be of equal size the calculation of region boundaries is trivial. Each boundary is exactly `regionsize` away from the previous boundary starting from 0 for the first region. As we will show, for equal sized regions, we don't need to store the boundary values.

To choose the next-hop we must determine which region contains the key. Because the regions are of equal size determining which region contains the key is a simple division operation.

```
regionsize = key.space.size / #{nexthops}
region = key / regionsize;
```

Thus the time required to find the next-hop is dependent on the way the next-hops are organized in memory. The obvious use of an array indexed by region yields $O(1)$.

2.2. Disruption

Protocols such as TCP perform better if the path they flow along does not change while the stream is connected. Disruption is the measurement of how many flows have their paths changed due to some change in the router. We measure disruption as the fraction of total flows whose path changes in response to some change in the router. This can become important if one or more of the paths is flapping. For a description of disruption and how it affects protocols such as

TCP see [1].

Some algorithms such as round-robin (i.e., upon receiving a packet the least recently used next-hop is chosen) are disruptive regardless of any change in the router. Clearly this is not the case with hash-threshold. As long as the region boundaries remain unchanged the same next-hop will be chosen for a given flow.

Because we have required regions to be equal in size the only reason for a change in region boundaries is the addition or removal of a next-hop. In this case the regions must all grow or shrink to fill the key space. The analysis begins with some examples of this.

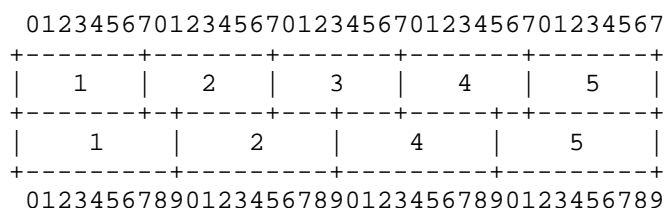


Figure 1. Before and after deletion of region 3

In figure 1. region 3 has been deleted. The remaining regions grow equally and shift to compensate. In this case 1/4 of region 2 is now in region 1, 1/2 (2/4) of region 3 is in region 2, 1/2 of region 3 is in region 4 and 1/4 of region 4 is in region 5. Since each of the original regions represent 1/5 of the flows, the total disruption is $1/5 \cdot (1/4 + 1/2 + 1/2 + 1/4)$ or $3/10$.

Note that the disruption to flows when adding a region is equivalent to that of removing a region. That is, we are considering the fraction of total flows that changes regions when moving from N to N-1 regions, and that same fraction of flows will change when moving from N-1 to N regions.

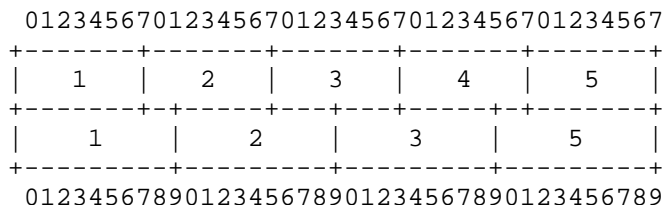


Figure 2. Before and after deletion of region 4

In figure 2. region 4 has been deleted. Again the remaining regions grow equally and shift to compensate. 1/4 of region 2 is now in region 1, 1/2 of region 3 is in region 2, 3/4 of region 4 is in region 3 and 1/4 of region 4 is in region 5. Since each of the original regions represent 1/5 of the flows the, total disruption is 7/20.

To generalize, upon removing a region K the remaining N-1 regions grow to fill the 1/N space. This growth is evenly divided between the N-1 regions and so the change in size for each region is 1/N/(N-1) or 1/(N(N-1)). This change in size causes non-end regions to move. The first region grows and so the second region is shifted towards K by the change in size of the first region. 1/(N(N-1)) of the flows from region 2 are subsumed by the change in region 1's size. 2/(N(N-1)) of the flows in region 3 are subsumed by region 2. This is because region 2 has shifted by 1/(N(N-1)) and grown by 1/(N(N-1)). This continues from both ends until you reach the regions that bordered K. The calculation for the number of flows subsumed from the Kth region into the bordering regions accounts for the removal of the Kth region. Thus we have the following equation.

$$\text{disruption} = \sum_{i=1}^{K-1} \frac{i}{(N)(N-1)} + \sum_{i=K+1}^N \frac{(i-K)}{(N)(N-1)}$$

We can factor 1/((N)(N-1)) out as it is constant.

$$= \frac{1}{(N)(N-1)} \left[\sum_{i=1}^{K-1} i + \sum_{i=K+1}^N (i-K) \right]$$

We now use the the concrete formulas for the sum of integers. The first summation is $(K)(K-1)/2$. For the second summation notice that we are summing the integers from 1 to $N-K$, thus it is $(N-K)(N-K+1)/2$.

$$= \frac{(K-1)(K) + (N-K)(N-K+1)}{2(N)(N-1)}$$

Considering the summations, one can see that the least disruption is when K is as close to half way between 1 and N as possible. This can be proven by finding the minimum of the concrete formula for K holding N constant. First break apart the quantities and collect.

$$= \frac{2K^2K - 2K - 2NK + N^2N + N}{2(N)(N-1)}$$

$$= \frac{K^2K - K - NK}{(N)(N-1)} + \frac{N + 1}{2(N-1)}$$

Since we are minimizing for K the right side $(N+1)/2(N-1)$ is constant as is the denominator $(N)(N-1)$ so we can drop them. To minimize we take the derivative.

$$\begin{aligned} \frac{d}{dk} (K^2K - (N+1)K) \\ = 2K - (N+1) \end{aligned}$$

Which is zero when K is $(N+1)/2$.

The last thing to consider is that K must be an integer. When N is odd $(N+1)/2$ will yield an integer, however when N is even $(N+1)/2$ yields an integer + $1/2$. In the case, because of symmetry, we get the least disruption when K is $N/2$ or $N/2 + 1$.

Now since the formula is quadratic with a global minimum half way between 1 and N the maximum possible disruption must occur when edge regions (1 and N) are removed. If K is 1 or N the formula reduces to $1/2$.

The minimum possible disruption is obtained by letting $K=(N+1)/2$. In this case the formula reduces to $1/4 + 1/(4*N)$. So the range of possible disruption is $(1/4, 1/2]$.

To minimize disruption we recommend adding new regions to the center rather than the ends.

3. Comparison to other algorithms

Other algorithms exist to decide which next-hop to use. These algorithms all have different performance and disruptive characteristics. Of these algorithms we will only consider ones that are not disruptive by design (i.e., if no change to the set of next-hops occurs the path a flow takes remains the same). This will exclude round-robin and random choice. We will look at modulo-N and highest random weight.

Modulo-N is a "simpler" form of hash-threshold. Given N next-hops the packet header fields which describe the flow are run through a hash function. A final modulo-N is applied to the output of the hash. This result then directly maps to one of the next-hops. Modulo-N is the most disruptive of the algorithms; if a next-hop is added or removed the disruption is $(N-1)/N$. The performance of Modulo-N is equivalent to hash-threshold.

Highest random weight (HRW) is a comparative method similar in some ways to hash-threshold with non-fixed sized regions. For each next-hop, the router seeds a pseudo-random number generator with the packet header fields which describe the flow and the next-hop to obtain a weight. The next-hop which receives the highest weight is selected. The advantage with using HRW is that it has minimal disruption (i.e., disruption due to adding or removing a next-hop is always $1/N$.) The disadvantage with HRW is that the next-hop selection is more expensive than hash-threshold. A description of HRW along with comparisons to other methods can be found in [2]. Although not used for next-hop calculation an example usage of HRW can be found in [3].

Since each of modulo-N, hash-threshold and HRW require a hash on the packet header fields which define a flow, we can factor the performance of the hash out of the comparison. If the hash can not be done inexpensively (e.g., in hardware) it too must be considered when using any of the above methods.

The lookup performance for hash-threshold, like modulo-N is an optimal $O(1)$. HRW's lookup performance is $O(N)$.

Disruptive behavior is the opposite of performance. HRW is best with $1/N$. Hash-threshold is between $1/4$ and $1/2$. Finally Modulo-N is $(N-1)/N$.

If the complexity of HRW's next-hop selection process is acceptable we think it should be considered as an alternative to hash-threshold. This could be the case when, for example, per-flow state is kept and thus the next-hop choice is made infrequently.

However, when HRW's next-hop selection is seen as too expensive the obvious choice is hash-threshold as it performs as well as modulo-N and is less disruptive.

4. Security Considerations

This document is an analysis of an algorithm used to implement an ECMP routing decision. This analysis does not directly affect the security of the Internet Infrastructure.

5. References

- [1] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast", RFC 2991, November 2000.
- [2] Thaler, D. and C.V. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates", IEEE/ACM Transactions on Networking, February 1998.
- [3] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P. and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, June 1998.

6. Author's Address

Christian E. Hopps
NextHop Technologies, Inc.
517 W. William Street
Ann Arbor, MI 48103-4943
U.S.A

Phone: +1 734 936 0291
EMail: chopps@nexthop.com

7. Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

