Internet Engineering Task Force (IETF)
Request for Comments: 6438
Category: Standards Track
ISSN: 2070-1721

B. Carpenter
Univ. of Auckland
S. Amante
Level 3

November 2011

Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels

Abstract

The IPv6 flow label has certain restrictions on its use. This document describes how those restrictions apply when using the flow label for load balancing by equal cost multipath routing and for link aggregation, particularly for IP-in-IPv6 tunneled traffic.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6438.

Table of Contents

1. Introduction

When several network paths between the same two nodes are known by the routing system to be equally good (in terms of capacity and latency), it may be desirable to share traffic among them. Two such techniques are known as equal cost multipath (ECMP) routing and link aggregation (LAG) [IEEE802.1AX]. There are, of course, numerous possible approaches to this, but certain goals need to be met:

- Maintain roughly equal share of traffic on each path. (In some cases, the multiple paths might not all have the same capacity, and the goal might be appropriately weighted traffic shares rather than equal shares. This would affect the load-sharing algorithm but would not otherwise change the argument.)

- Minimize or avoid out-of-order delivery for individual traffic flows.

- Minimize idle time on any path when queue is non-empty.

There is some conflict between these goals: for example, strictly avoiding idle time could cause a small packet sent on an idle path to overtake a bigger packet from the same flow, causing out-of-order delivery.

One lightweight approach to ECMP or LAG is this: if there are N equally good paths to choose from, then form a modulo(N) hash [RFC2991] from a defined set of fields in each packet header that are certain to have the same values throughout the duration of a flow, and use the resulting output hash value to select a particular path. If the hash function is chosen so that the output values have a uniform statistical distribution, this method will share traffic roughly equally between the N paths. If the header fields included in the hash input are consistent, all packets from a given flow will generate the same hash output value, so out-of-order delivery will

not occur. Assuming a large number of unique flows are involved, it is also probable that the method will avoid idle time, since the queue for each link will remain non-empty.

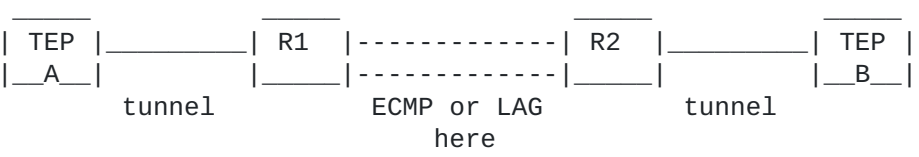
1.1. Choice of IP Header Fields for Hash Input

In the remainder of this document, we will use the term "flow" to represent a sequence of packets that may be identified by either the source and destination IP addresses alone {2-tuple} or the source IP address, destination IP address, protocol number, source port number, and destination port number {5-tuple}. It should be noted that the latter is more specifically referred to as a "microflow" in [RFC2474], but this term is not used in connection with the flow label in [RFC3697].

The question, then, is which header fields are used to identify a flow and serve as input keys to a modulo(N) hash algorithm. A common choice when routing general traffic is simply to use a hash of the source and destination IP addresses, i.e., the 2-tuple. This is necessary and sufficient to avoid out-of-order delivery and, with a wide variety of sources and destinations as one finds in the core of the network, often statistically sufficient to distribute the load evenly. In practice, many implementations use the 5-tuple {dest addr, source addr, protocol, dest port, source port} as input keys to the hash function, to maximize the probability of evenly sharing traffic over the equal cost paths. However, including transport-layer information as input keys to a hash may be a problem for IP fragments [RFC2991] or for encrypted traffic. Including the protocol and port numbers, totaling 40 bits, in the hash input makes the hash slightly more expensive to compute but does improve the hash distribution, due to the variable nature of ephemeral ports. Ephemeral port numbers are quite well distributed [Lee10] and will typically contribute 16 variable bits. However, in the case of IPv6, transport-layer information is inconvenient to extract, due to the variable placement of and variable length of next-headers; all implementations must be capable of skipping over next-headers, even if they are rarely present in actual traffic. In fact, [RFC2460] implies that next-headers, except hop-by-hop options, are not normally inspected by intermediate nodes in the network. This situation may be challenging for some hardware implementations, raising the potential that network equipment vendors might sacrifice the length of the fields extracted from an IPv6 header.

It is worth noting that the possible presence of a Generic Routing Encapsulation (GRE) header [RFC2784] and the possible presence of a GRE key within that header creates a similar challenge to the possible presence of IPv6 extension headers; anything that complicates header analysis is undesirable.

The situation is different in IP-in-IP tunneled scenarios. Identifying a flow inside the tunnel is more complicated, particularly because nearly all hardware can only identify flows based on information contained in the outermost IP header. Assume that traffic from many sources to many destinations is aggregated in a single IP-in-IP tunnel from tunnel endpoint (TEP) A to TEP B (see figure). Then all the packets forming the tunnel have outer source address A and outer destination address B. In all probability, they also have the same port and protocol numbers. If there are multiple paths between routers R1 and R2, and ECMP or LAG is applied to choose a particular path, the 2-tuple or 5-tuple (and its hash) will be constant, and no load sharing will be achieved, i.e., polarization will occur. If there is a high proportion of traffic from one or a small number of tunnels, traffic will not be distributed as intended across the paths between R1 and R2, due to partial polarization. (Related issues arise with MPLS [MPLS-LABEL].)



As noted above, for IPv6, the 5-tuple is quite inconvenient to extract due to the next-header placement. The question therefore arises whether the 20-bit flow label in IPv6 packets would be suitable for use as input to an ECMP or LAG hash algorithm, especially in the case of tunnels where the inner packet header is inaccessible. If the flow label could be used in place of the port numbers and protocol number in the 5-tuple, the implementation would be simplified.

1.2. Flow Label Rules

The flow label was left Experimental by [RFC2460] but was better defined by [RFC3697]. We quote three rules from that RFC:

1. "The Flow Label value set by the source MUST be delivered unchanged to the destination node(s)."

2. "IPv6 nodes MUST NOT assume any mathematical or other properties of the Flow Label values assigned by source nodes."

3. "Router performance SHOULD NOT be dependent on the distribution of the Flow Label values. Especially, the Flow Label bits alone make poor material for a hash key."

These rules, especially the last one, have caused designers to hesitate about using the flow label in support of ECMP or LAG. The fact is that today most nodes set a zero value in the flow label, and the first rule definitely forbids the routing system from changing the flow label once a packet has left the source node. Considering normal IPv6 traffic, the fact that the flow label is typically zero means that it would add no value to an ECMP or LAG hash, but neither would it do any harm to the distribution of the hash values.

However, in the case of an IP-in-IPv6 tunnel, the TEP is itself the source node of the outer packets. Therefore, a TEP may freely set a flow label in the outer IPv6 header of the packets it sends into the tunnel.

The second two rules quoted above need to be seen in the context of [RFC3697], which assumes that routers using the flow label in some way will be involved in some sort of method of establishing flow state: "To enable flow-specific treatment, flow state needs to be established on all or a subset of the IPv6 nodes on the path from the source to the destination(s)." The RFC should perhaps have made clear that a router that has participated in flow state establishment can rely on properties of the resulting flow label values without further signaling. If a router knows these properties, rule 2 is irrelevant, and it can choose to deviate from rule 3.

In the tunneling situation sketched above, routers R1 and R2 can rely on the flow labels set by TEP A and TEP B being assigned by a known method. This allows an ECMP or LAG method to be based on the flow label consistently with [RFC3697], regardless of whether the non-tunnel traffic carries non-zero flow label values.

The IETF has recently revised RFC 3697 [RFC6437]. That revision is fully compatible with the present document and obviates the concerns resulting from the above three rules. Therefore, the present specification applies both to RFC 3697 and to RFC 6437.

2. Normative Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Guidelines

We assume that the routers supporting ECMP or LAG (R1 and R2 in the above figure) are unaware that they are handling tunneled traffic. If it is desired to include the IPv6 flow label in an ECMP or LAG hash in the tunneled scenario shown above, the following guidelines apply:

- Inner packets MUST be encapsulated in an outer IPv6 packet whose source and destination addresses are those of the tunnel endpoints (TEPs).

- The flow label in the outer packet SHOULD be set by the sending TEP to a 20-bit value in accordance with [RFC6437]. The same flow label value MUST be used for all packets in a single user flow, as determined by the IP header fields of the inner packet.

- To achieve this, the sending TEP MUST classify all packets into flows once it has determined that they should enter a given tunnel and then write the relevant flow label into the outer IPv6 header. A user flow could be identified by the sending TEP most simply by its {destination, source} address 2-tuple or by its 5-tuple {dest addr, source addr, protocol, dest port, source port}. At present, there would be little point in using the {dest addr, source addr, flow label} 3-tuple of the inner packet, but doing so would be a future-proof option. The choice of n-tuple is an implementation choice in the sending TEP.

  * As specified in [RFC6437], the flow label values should be chosen from a uniform distribution. Such values will be suitable as input to a load-balancing hash function and will be hard for a malicious third party to predict.

  * The sending TEP MAY perform stateless flow label assignment by using a suitable 20-bit hash of the inner IP header's 2-tuple or 5-tuple as the flow label value.

  * If the inner packet is an IPv6 packet, its flow label value could also be included in this hash.

  * This stateless method creates a small probability of two different user flows hashing to the same flow label. Since [RFC6437] allows a source (the TEP in this case) to define any set of packets that it wishes as a single flow, occasionally labeling two user flows as a single flow through the tunnel is acceptable.

- At intermediate routers that perform load distribution, the hash algorithm used to determine the outgoing component-link in an ECMP and/or LAG toward the next hop MUST minimally include the 3-tuple {dest addr, source addr, flow label} and MAY also include the remaining components of the 5-tuple. This applies whether the traffic is tunneled traffic only or a mixture of normal traffic and tunneled traffic.

  - Intermediate IPv6 router(s) will presumably encounter a mixture of tunneled traffic and normal IPv6 traffic. Because of this, the design should also include {protocol, dest port, source port} as input keys to the ECMP and/or LAG hash algorithms, to provide additional entropy for flows whose flow label is set to zero, including non-tunneled traffic flows.

- Individual nodes in a network are free to implement different algorithms that conform to this specification without impacting the interoperability or function of the network.

- Operations, Administration, and Maintenance (OAM) techniques will need to be adapted to manage ECMP and LAG based on the flow label. The issues will be similar to those that arise for MPLS [RFC4379] and pseudowires [RFC6391].

4. Security Considerations

The flow label is not protected in any way and can be forged by an on-path attacker. However, it is expected that tunnel endpoints and the ECMP or LAG paths will be part of a managed infrastructure that is well protected against on-path attacks (e.g., by using IPsec between the two tunnel endpoints). Off-path attackers are unlikely to guess a valid flow label if an apparently pseudo-random and unpredictable value is used. In either case, the worst an attacker could do against ECMP or LAG is attempt to selectively overload a particular path. For further discussion, see [RFC6437].

5. Acknowledgements

This document was suggested by corridor discussions at IETF 76. Joel Halpern made crucial comments on an early version. We are grateful to Qinwen Hu for general discussion about the flow label. Valuable comments and contributions were made by Miguel Garcia, Brian Haberman, Sheng Jiang, Thomas Narten, Jarno Rajahalme, Brian Weis, and others.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

6.2. Informative References

[IEEE802.1AX] Institute of Electrical and Electronics Engineers, "Link Aggregation", IEEE Standard 802.1AX-2008, 2008.

[Lee10] Lee, D., Carpenter, B., and N. Brownlee, "Observations of UDP to TCP Ratio and Port Numbers", Fifth International Conference on Internet Monitoring and Protection ICIMP 2010, May 2010.

[MPLS-LABEL] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", Work in Progress, May 2011.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC6391] Bryant, S., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, November 2011.

Authors' Addresses

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

EMail: brian.e.carpenter@gmail.com

Shane Amante
Level 3 Communications, LLC
1025 Eldorado Blvd
Broomfield, CO 80021
USA

EMail: shane@level3.net