Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 10, 2017

D. Dolson
J. Snellman
Sandvine
M. Boucadair
C. Jacquenet
Orange

March 9, 2017

# Beneficial Functions of Middleboxes
## draft-dolson-plus-middlebox-benefits-03

Abstract

At IETF97, at a meeting regarding the Path Layer UDP Substrate (PLUS) protocol, a request was made for documentation about the benefits that might be provided by permitting middleboxes to have some visibility to transport-layer information.

This document summarizes benefits provided to the Internet by intermediary devices that provide functions apart from normal IP forwarding. Such intermediary devices are often called "middleboxes".

RFC3234 defines a taxonomy of middleboxes and issues in the Internet. Most of those middleboxes utilize or modify application-layer data. This document primarily focuses on devices that observe and act on information carried in the transport layer, and especially information carried in TCP packets.

A primary goal of this document is to provide information to working groups developing new transport protocols, in particular the PLUS and QUIC working groups, to aid understanding of what might be gained or lost by design decisions that may affect (or be affected by) middlebox operation.

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 September 10, 2017.



Table of Contents

## 1. Introduction

From RFC3234 [RFC3234], "A middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host."

Middleboxes are usually (but not exclusively) deployed at locations permitting observation of bidirectional traffic flows. Such locations are typically points where stub networks connect to the Internet; e.g.,:

- Where a residential or business customer connects to its service provider(s), which may include multi-homing.
- On the Gi interface where a GGSN connects to a PDN (see section 3.1 of [RFC6459]).

The QUIC working group and PLUS BoF are debating the appropriate amount of information that end-points should expose to on-path network middleboxes and human trouble-shooters. (Some information used for debugging is discussed in <https://www.snellman.net/blog/archive/2016-12-01-quic-tou/>.) This document itemizes a variety of features provided by middleboxes and by ad hoc analysis performed by operators using packet analyzers.

Many of the techniques described in this document require stateful analysis of transport streams. A generic state machine is described in [I-D.trammell-plus-statefulness].

Although many middleboxes observe and manipulate application-layer content (e.g., session boarder controllers [RFC5853]) they are out of scope for this document, the aim being to describe benefits of middleboxes using transport-layer features. An earlier document [I-D.mm-wg-effect-encrypt] describes the impact of pervasive encryption of application-layer data on network monitoring, protecting and troubleshooting.

This document advocates for transport connections to be measured and managed by the network for the benefit of both parties: for the end-user to receive better quality of experience, and for the network

operator to improve resource usage, the former being a consequence of the latter.

This document does not discuss whether exposing some data to on-path devices for network assistance purposes can be achieved by using in-band or out-of-band mechanisms.

### 1.1. Requirements Language

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 RFC 2119 [RFC2119].

## 2. Measurements

A number of measurements can be made by network devices that are either in-line with the traffic (responsible for forwarding) or receiving off-line copy of traffic from a tap or file capture. These measurements can be used either by automated systems, or for manual network troubleshooting purposes (e.g., using packet analysis tools). The automated systems can further be classified as monitoring systems that compute performance indicators for large numbers of connections and generate aggregated reports from them, and active systems that make decisions on how to handle specific packets based on these performance indicators.

Long-term trends in these measurements can aid an operator in capacity planning. Short-term anomalies revealed by these measurements can identify network breakages, attacks in progress, or misbehaving devices/applications.

### 2.1. Packet Loss

Network problems and under-provisioning can be detected if packet loss is measurable. TCP packet loss can be detected by observing gaps in sequence numbers, retransmitted sequence numbers, and SACK options. Packet loss can be detected per direction.

Gaps indicate loss upstream of the tap point; retransmissions indicate loss downstream of the tap. Selective acknowledgements (SACKs) can be used to detect either upstream or downstream packet loss (although some care needs to be taken to avoid mis-identifying packet reordering as packet loss), and to distinguish between upstream vs. downstream losses.

Packet loss measurements on both sides of the measurement point are an important component in precisely diagnosing insufficiently dimensioned devices or links in networks. Additionally, since packet

losses are one of the two main ways for congestion to manifest (the other being queueing delay), packet loss is an important measurement for any middlebox that needs to make traffic handling decisions based on observed levels of congestion.

## 2.2. Round Trip Times

A TCP packet stream can be used to measure the round-trip time on each side of the measurement point. During the connection handshake, the SYN, SYNACK, and ACK timings can be used to establish a baseline RTT in each direction. Once the connection is established, the RTT between the server and the measurement point can only reliably be determined using TCP timestamps. On the side between the measurement point and the client, the exact timing of data segments and ACKs can be used as an alternative. For this latter method to be accurate when packet loss is present, the connection must use selective acknowledgements.

In many networks, congestion will show up as increasing packet queueing, and congestion-induced packet loss will only happen in extreme cases. RTTs will also show up as a much smoother signal than the discrete packet loss events. This makes RTTs a good way to identify individual subscribers for whom the network is a bottleneck at a given time, or geographical sites (such as cellular towers) that are experiencing large scale congestion.

The main limit of RTT measurement as a congestion signal is the difficulty of reliably distinguishing between the data segments being queued vs. the ACKs being queued.

## 2.3. Measuring Packet Reordering

If a network is reordering packets of transport connections, caused perhaps by ECMP misconfiguration (e.g., described in [RFC2991] and [RFC7690]), the end-points may react as if packet loss is occurring and retransmit packets or reduce forwarding rates. It is therefore beneficial to be able to diagnose packet reordering from within a network.

For TCP, packet reordering can be detected by observing TCP sequence numbers per direction. See for example a number of standard packet reordering metrics in [RFC4737] and informational metrics in [RFC5236].

### 2.4. Throughput and Bottleneck Identification

Although throughput to or from an IP address can be measured without transport-layer measurements, the transport layer provides clues about what the end-points were attempting to do.

One way of quickly excluding the network as the bottleneck during troubleshooting is to check whether the speed is limited by the endpoints. For example, the connection speed might instead be limited by suboptimal TCP options, the sender's congestion window, the sender temporarily running out of data to send, the sender waiting for the receiver to send another request, or the receiver closing the receive window.

This data is also useful for middleboxes used to measure network quality of service. Connections, or portions of connections, that are limited by the endpoints do not provide an accurate measure of network's speed, and can be discounted or completely excluded in such analyses.

### 2.5. DDoS Detection

When an application or network resource is under attack, it is useful to identify this situation from the network perspective, upstream of the attacked resource.

Although detection methods tend to be proprietary, DDoS attack detection is fundamentally one of:

- detecting protocol violations by tracking the transport-layer state machine or application-layer messaging; or
- anomaly detection by noticing atypical traffic patterns taken from measurements.

Two trends in protocol design will make DDoS detection more difficult:

- the desire to encrypt transport-layer communication and sequence numbers;
- the desire to avoid statistical fingerprinting by adding entropy in various forms.

Those desires assist in the worthy goal of improved privacy, but also serve to defeat DDoS detection.

### 2.6. Packet Corruption

One notable source of packet loss is packet corruption. This corruption will generally not be detected until the checksums are validated by the endpoint, and the packet is dropped. This means that detecting the exact location where packets are lost is not sufficient when troubleshooting networks. It should also be possible to find out where packets are being corrupted. IP and TCP checksum verification allows a measurement device to correctly distinguish between upstream packet corruption and normal downstream packet loss.

Transport protocol designers should consider whether a middlebox will be able to detect corrupted or tampered packets.

### 2.7. Application-Layer Measurements

Network health may also be gleaned from application-layer diagnosis. E.g.,

- DNS response times and retransmissions by correlating answers to queries.
- Various protocol-aware voice and video quality analysis.

Could this type of information be provided in a transport layer?

## 3. Functions Beyond Measurement: A Few Examples

This section describes features provided by in-line devices that go beyond measurement by modifying, discarding, delaying, or prioritizing traffic.

### 3.1. NAT

Network Address Translators (NATs) allow multiple devices to share a public address by dividing the transport-layer port space among the devices.

NAT behavior recommendations are found for UDP in BCP 127 [RFC4787] and for TCP in BCP 142 [RFC7857].

To support NAT, there must be transport-layer port numbers that can be modified by the network. The application-layer must not assume the port number was left unchanged (e.g., by including it in a checksum or signing it).

Address sharing is also used in the context of IPv6 transition. For example, DS-Lite AFTR [RFC6333], NAT64 [RFC6146], or MAP-* are

features that are enabled in the network to allow for IPv4 service continuity over an IPv6 network.

Further, because of some multi-homing considerations, IPv6 prefix translation may be enabled by some enterprises by means of NPTv6 [RFC6296].

## 3.2. Firewall

Firewalls are pervasive and essential components that inspect incoming and outgoing traffic. Firewalls are usually the cornerstone of a security policy that is enforced in end-user premises and other locations to provide strict guarantees about traffic that may be authorized to enter/leave the said premises, as well as end-users who may be assigned different clearance levels regarding which networks and portions of the Internet they may acess.

Arguably many users within various types of organizations would not have been granted Internet access if not for safety provided by firewalls.

An important aspect of a firewall policy is differentiating internally-initiated from externally-initiated communications.

For TCP, this is easily done by tracking the TCP state machine. Furthermore, the ending of a TCP connection is indicated by RST or FIN flags.

For UDP, the firewall can be opened if the first packet comes from an internal user, but the closing is generally done by an idle timer of arbitrary duration, which might not match the expectations of the application.

Simple IPv6 firewall capabilities for customer premises equipment (both stateless and stateful) are described in [RFC6092].

A firewall functions better when it can observe the protocol state machine, described generally by Transport-Independent Path Layer State Management [I-D.trammell-plus-statefulness].

## 3.3. DDoS Scrubbing

In the context of a distributed denial-of-service (DDoS) attack, the purpose of a scrubber is to discard attack traffic while permitting useful traffic. E.g., such a mitigator is described in [I-D.ietf-dots-architecture].

When attacks occur against constrained resources, there is obviously a huge benefit in being able to scrub well.

Furthermore, this is solely a task for an on-path network device because neither end-point of a legitimate connection has any control over the source of the attack traffic.

Source-spoofed DDoS attacks can be mitigated at the source using BCP 38 ([RFC2827]), but it is more difficult if source address filtering cannot be applied.

In contrast to devices in the core of the Internet, middleboxes statefully observing bidirectional transport connections can reject source-spoofed TCP traffic based on the inability to provide sensible acknowledgement numbers to complete the three-way handshake. Obviously this requires middlebox visibility into transport-layer state machine.

Middleboxes may also scrub on the basis of statistical classification: testing how likely a given packet is legitimate. As protocol designers add more entropy to headers and lengths, this test becomes less useful and the best scrubbing strategy becomes random drop.

### 3.4. Implicit Identification

In order to enhance the end-user’s quality of experience, some operators deploy implicit identification features that rely upon the correlation of network-related information to access some local services. For example, service portals operated by some operators may be accessed immediately by end-users without any explicit identification for the sake of improved service availability. This is doable thanks to on-path devices that inject appropriate metadata that can be used by the remote server to enforce per-subscriber policies. The information can be injected at the application layer or at the transport layer (when an address sharing mechanism is in use).

An experimental implementation using a TCP option is described in [RFC7974].

For the intended use of implicit identification, it is more secure to have a trusted middlebox mark this traffic than to trust end-user devices.

### 3.5. Performance-Enhancing Proxies

Performance-Enhancing Proxies (PEPs) can improve performance in some types of networks by improving packet spacing or generating local acknowledgements, and are most commonly used in satellite and cellular networks. Transport-Layer PEPs are described in section 2.1.1 of [RFC3135].

PEPs allow central deployment of congestion control algorithms more suited to the specific network, most commonly use of delay-based congestion control. More advanced TCP PEPs deploy congestion control systems that treat all of a single end-user's TCP connections as a single unit, improving fairness and allowing faster reaction to changing network conditions.

Local acknowledgements generated by PEPs speed up TCP slow start by splitting the effective latency, and allow for retransmissions to be done from the PEP rather than from the actual sender, saving downlink bandwidth on retransmissions. Local acknowledgements will also allow a PEP to maintain a local buffer of data appropriate to the actual network conditions, whereas the actual endpoints would often send too much or too little.

A PEP function requires transport-layer fields that allow chunks of data to be identified (e.g., TCP sequence numbers), acknowledgements to be identified (e.g., TCP ACK numbers), and acknowledgements to be created from the PEP.

Note that PEPs are only useful in some types of networks, and poor design could make performance worse.

### 3.6. Network Coding

Network Coding is a technique for compressing traffic or adding redundancy for transmission over low-bandwidth, long-latency links such as satellite links. One method is to deploy network-coding gateways at each end of those links, with a network-coding tunnel between them via the slow/lossy/long-latency links.

The network coding gateways may employ some techniques of PEPs, such as creating acknowledgements of queued data, removing retransmissions and pacing data rates to reduce queue oscillation.

### 3.7. Network-Assisted Bandwidth Aggregation

The Hybrid Access Aggregation Point (HAAP) is a middlebox that allows customers to aggregate the bandwidth of multiple access technologies [I-D.zhang-banana-problem-statement].

One of the approaches uses MPTCP proxies [I-D.nam-mptcp-deployment-considerations] to forward traffic along multiple paths. The MPTCP proxy operates at the transport layer while being located in the operator's network.

The support of multipath transport capabilities by communicating hosts remains a privileged target design so that such hosts can directly use the available resources provided by a variety of access networks they can connect to. Nevertheless, network operators do not control end hosts while the support of MPTCP by content servers remains marginal.

Network-Assisted MPTCP deployment models are designed to facilitate the adoption of MPTCP for the establishment of multi-path communications without making any assumption about the support of MPTCP capabilities by communicating peers. Network-Assisted MPTCP deployment models rely upon MPTCP Conversion Points (MCPs) that act on behalf of hosts so that they can take advantage of establishing communications over multiple paths [I-D.boucadair-mptcp-plain-mode].

Note that an MPTCP proxy can be beneficial even if both the client and the server are MPTCP-compliant. Examples of such cases are listed below:

1. The use of private IPv4 addresses in some access networks. Typically, additional subflows can not be added to the MPTCP connection without the help of an MCP.

2. The assignment of IPv6 prefixes only by some networks. If the server is IPv4-only, IPv6 subflows cannot be added to an MPTCP connection established with that server, by definition.

3. Subscription to some service offerings is subject to volume quota.

### 3.8. Prioritization and Differentiated Services

Bulk traffic may be served with a higher latency than interactive traffic with no reduction in throughput. This fact allows a middlebox function to improve response times in interactive applications by prioritizing, policing, or remarking interactive transport connections differently from bulk traffic transport connections. E.g., gaming traffic may be prioritized over email or software updates.

Middleboxes may identify different classes of traffic by inspecting multiple layers of header and payload.

### 3.9. Measurement-Based Shaping

Basic traffic shaping functionality requires no transport-layer information. All that is needed is a way of mapping each packet to a traffic shaper quota. For example, there may be a rate limit per 5-tuple or per subscriber IP address. However, such fixed traffic shaping rules are wasteful as they end up rate limiting traffic even when the network has free resources available.

More advanced traffic shaping devices use transport layer metrics described in Section 2 to detect congestion on either a per-site or per-user level, and use different traffic shaping rules when congestion is detected. This type of device can overcome limitations of down-stream devices that behave poorly (e.g., by excessive buffering or sub-optimally dropping packets).

### 3.10. Fairness to End-User Quota

Several service offerings rely upon a volume-based charging model. Operators may assist end-users in conserving their data quota by deploying on-path functions that shape traffic that would otherwise be agressively transferred.

For example, a fast download of a video that won’t be viewed completely by the subscriber may lead to quick exhaustion of the data quota. Limiting the video download rate conserves quota for the benefit of the end-user.

## 4. Acknowledgements

The authors thank Brian Trammell and Brian Carpenter for their review and suggestions.

## 5. IANA Considerations

This memo includes no request to IANA.

## 6. Security Considerations

### 6.1. Confidentiality

This document intentionally excludes middleboxes that observe or manipulate application-layer data.

The benefits described in this document can all be implemented without violating confidentiality. However, there is always the question of whether the fields and packet properties used to achieve these benefits may also be used for harm.

In particular, we want to ask what confidentiality is lost by exposing transport-layer fields beyond what can be learned by observing IP-layer fields.

Sequence numbers: an observer can learn how much data is transferred.

Start/Stop indicators: an observer can count transactions for some applications.

Device fingerprinting: an observer may be more easily able to identify a device type when different devices use different default field values or options.

### 6.2. Active Attacks

Being able to observe sequence numbers or session identifiers may make it easier to modify or terminate a transport connection. E.g., observing TCP sequence numbers allows generation of a RST packet that terminates the connection. However, signing transport fields mitigates this attack. The attack and solution are described for the TCP authentication option [RFC5925].

### 6.3. More Information Can Improve Security

Proposition: network maintainability and security can be improved by providing firewalls and DDoS mechanisms with some information about transport connections. In contrast, it would be very difficult to secure a network in which every packet appears unique and filled with random bits.

For denial-of-service (DoS) attacks on bandwidth, the receiving end-point is usually on the wrong side of the constrained network link. This fact makes it seem reasonable to give some clues to allow a middlebox device to help out before the constrained link.

E.g., in a blind attack, an attacker cannot receive data from the target of the attack (section 4.6.3.2 of [RFC3552]). In the case of TCP, the blind attacker cannot complete the three-way handshake.

In the balance, some features providing the ability to mitigate/ filter attacks and fix broken networks will improve security vs. the scenario when all packets are completely opaque.

## 7. References

### 7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <http://www.rfc-editor.org/info/rfc4737>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <http://www.rfc-editor.org/info/rfc4787>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <http://www.rfc-editor.org/info/rfc6333>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <http://www.rfc-editor.org/info/rfc7857>.

## 7.2. Informative References

[I-D.boucadair-mptcp-plain-mode]
Boucadair, M., Jacquenet, C., Bonaventure, O., Behaghel, D., stefano.secci@lip6.fr, s., Henderickx, W., Skog, R., Vinapamula, S., Seo, S., Cloetens, W., Meyer, U., Contreras, L., and B. Peirens, "An MPTCP Option for Network-Assisted MPTCP", draft-boucadair-mptcp-plain-mode-09 (work in progress), October 2016.

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.mm-wg-effect-encrypt]
Moriarty, K. and A. Morton, "Effect of Pervasive Encryption", draft-mm-wg-effect-encrypt-07 (work in progress), February 2017.

[I-D.nam-mptcp-deployment-considerations]
Boucadair, M., Jacquenet, C., Bonaventure, O., Henderickx, W., and R. Skog, "Network-Assisted MPTCP: Use Cases, Deployment Scenarios and Operational Considerations", draft-nam-mptcp-deployment-considerations-01 (work in progress), December 2016.

[I-D.trammell-plus-statefulness]
Kuehlewind, M., Trammell, B., and J. Hildebrand, "Transport-Independent Path Layer State Management", draft-trammell-plus-statefulness-02 (work in progress), December 2016.

[I-D.zhang-banana-problem-statement]
Cullen, M., Leymann, N., Heidemann, C., Boucadair, M., Hui, D., Zhang, M., and B. Sarikaya, "Problem Statement: Bandwidth Aggregation for Internet Access", draft-zhang-banana-problem-statement-03 (work in progress), October 2016.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, DOI 10.17487/RFC2991, November 2000, <http://www.rfc-editor.org/info/rfc2991>.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <http://www.rfc-editor.org/info/rfc3135>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC5236] Jayasumana, A., Piratla, N., Banka, T., Bare, A., and R. Whitner, "Improved Packet Reordering Metrics", RFC 5236, DOI 10.17487/RFC5236, June 2008, <http://www.rfc-editor.org/info/rfc5236>.

[RFC5853] Hautakorpi, J., Ed., Camarillo, G., Penfield, R., Hawrylyshen, A., and M. Bhatia, "Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments", RFC 5853, DOI 10.17487/RFC5853, April 2010, <http://www.rfc-editor.org/info/rfc5853>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <http://www.rfc-editor.org/info/rfc6092>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <http://www.rfc-editor.org/info/rfc6459>.

[RFC7690] Byerly, M., Hite, M., and J. Jaeggli, "Close Encounters of the ICMP Type 2 Kind (Near Misses with ICMPv6 Packet Too Big (PTB))", RFC 7690, DOI 10.17487/RFC7690, January 2016, <http://www.rfc-editor.org/info/rfc7690>.

[RFC7974] Williams, B., Boucadair, M., and D. Wing, "An Experimental TCP Option for Host Identification", RFC 7974, DOI 10.17487/RFC7974, October 2016, <http://www.rfc-editor.org/info/rfc7974>.

Authors' Addresses

David Dolson
Sandvine
408 Albert Street
Waterloo, ON  N2L 3V3
Canada

Phone: +1 519 880 2400
Email: ddolson@sandvine.com

Juho Snellman
Sandvine
Seestrasse 5
Zurich  8002
Switzerland

Email: jsnellman@sandvine.com

Mohamed Boucadair
Orange
4 rue du Clos Courtel
Rennes  35000
France

Email: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
4 rue du Clos Courtel
Rennes  35000
France

Email: christian.jacquenet@orange.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 17, 2018

M. Kuehlewind
B. Trammell
ETH Zurich
J. Hildebrand
November 13, 2017

# Transport-Independent Path Layer State Management
## draft-trammell-plus-statefulness-04

## Abstract

This document describes a simple state machine for stateful network devices on a path between two endpoints to associate state with traffic traversing them on a per-flow basis, as well as abstract signaling mechanisms for driving the state machine. This state machine is intended to replace the de-facto use of the TCP state machine or incomplete forms thereof by stateful network devices in a transport-independent way, while still allowing for fast state timeout of non-established or undesirable flows.

Table of Contents



## 1. Introduction

The boundary between the network and transport layers was originally defined to be that between information used (and potentially modified) hop-by-hop, and that used end-to-end. End-to-end information in the transport layer is associated with state at the endpoints, but processing of network-layer information was assumed to be stateless.

The widespread deployment of stateful middleboxes in the Internet, such as network address and port translators (NAPT), firewalls that model the TCP state machine to distinguish packets belonging from desirable flows from backscatter and random attack traffic, and devices which keep per-flow state for reporting and monitoring

purposes (e.g. IPFIX [RFC7011] Metering Processes), has broken this assumption, and made it more difficult to deploy non-TCP transport protocols in the Internet.

The deployment of new transport protocols encapsulated in UDP with encrypted transport headers (such as QUIC [I-D.ietf-quic-transport]) will present a challenge to the operation of these devices, and their ubquity likewise threatens to impair the deployability of these protocols. There are two main causes for this problem: first, stateful devices often use an internal model of the TCP state machine to determine when TCP flows start and end, allowing them to manage state for these flows; for UDP flows, they must rely on timeouts. These timeouts are generally short relative to those for TCP [IMC-GATEWAYS], requiring UDP- encapsulated transports either to generate unproductive keepalive traffic for long-lived sessions, or to tolerate connectivity problems and the necessity of reconnection due to loss of on-path state.

This document presents an abstract solution to this problem by defining a transport-independent state machine to be implemented at per-flow state- keeping middleboxes as a replacement for incomplete TCP state modeling. A key concept behind this approach is that encryption of transport protocol headers allows a transport protocol to separate its wire image - what it looks like to devices on path - from its internal semantics. We advocate the creation of a minimal wire image for these protocols that exposes enough information to drive the state machine presented. Present and future evolution of encrypted transport protocols can then happen behind this wire image, and Middleboxes implementing this state machine can use signals from a UDP encapsulation common to a set of encrypted transport protocols can have equivalent state information to that provided by TCP, reducing the friction between deployed middleboxes and these new transport protocols.

## 2. Terminology

In this document, the term "flow" is defined to be compatible with the definition given in [RFC7011]: A flow is defined as a set of packets passing a device on the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. one or more network layer header fields (e.g., destination IP address) or transport layer header fields (e.g., destination port number) that the device has access to;

2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc.);

3. one or more of the fields derived from packet treatment at the device (e.g., next-hop IP address, the output interface, etc.).

A packet is defined as belonging to a flow if it completely satisfies all the defined properties of the flow.

A bidirectional flow or biflow is defined as compatible with [RFC5103], by joining the "forward direction" flow with the "reverse direction" flow, derived by reversing the direction of directional fields (ports and IP addresses). Biflows are only relevant at devices positioned so as to see all the packets in both directions of the biflow, generally on the endpoint side of the service demarcation point for either endpoint as defined in the reference path given in [RFC7398].

## 3. State Machine

A transport-independent state machine for on-path devices is shown in Figure 1. It was designed to have the following properties:

- A device on path that can see traffic in both directions between two endpoints knows that each side of an association wishes that association to continue. This allows firewalls to delegate policy decisions about accepting or continuing an association to the servers they protect.

- A device on path that can see traffic in both directions between two endpoints knows that each device can receive traffic at the source address it provides. This allows firewalls to provide protection against trivially spoofed packets.

Both of these properties hold with current firewalls and network address translation devices observing the flags and sequence/ acknowledgment numbers exposed by TCP.

It relies on six states, three configurable timeouts, and a set of signals defined in Section 4. The states are defined as follows:

- zero: there is no state for a given flow at the device

- uniflow: at least one packet has been seen in one direction

- associating: at least one packet has been seen in one direction, and an indication that the receiving endpoint wishes to continue the association has been seen in the other direction.

- associated: a flow in associating state has further demonstrated that the initial sender can receive packets at its given source address.

- stop-wait: one side of a connection has sent an explicit stop signal, waiting for confirmation

- stopping: stop signal confirmed, association is stopping.

We refer to the zero and uniflow states as "uniflow states", as they are relevant both for truly unidirectional flows, as well as in situations where an on-path device can see only one side of a communication. We refer to the remaining four states as "biflow states", as they are only applicable to true bidirectional flows, where the on-path device can see both sides of the communication.

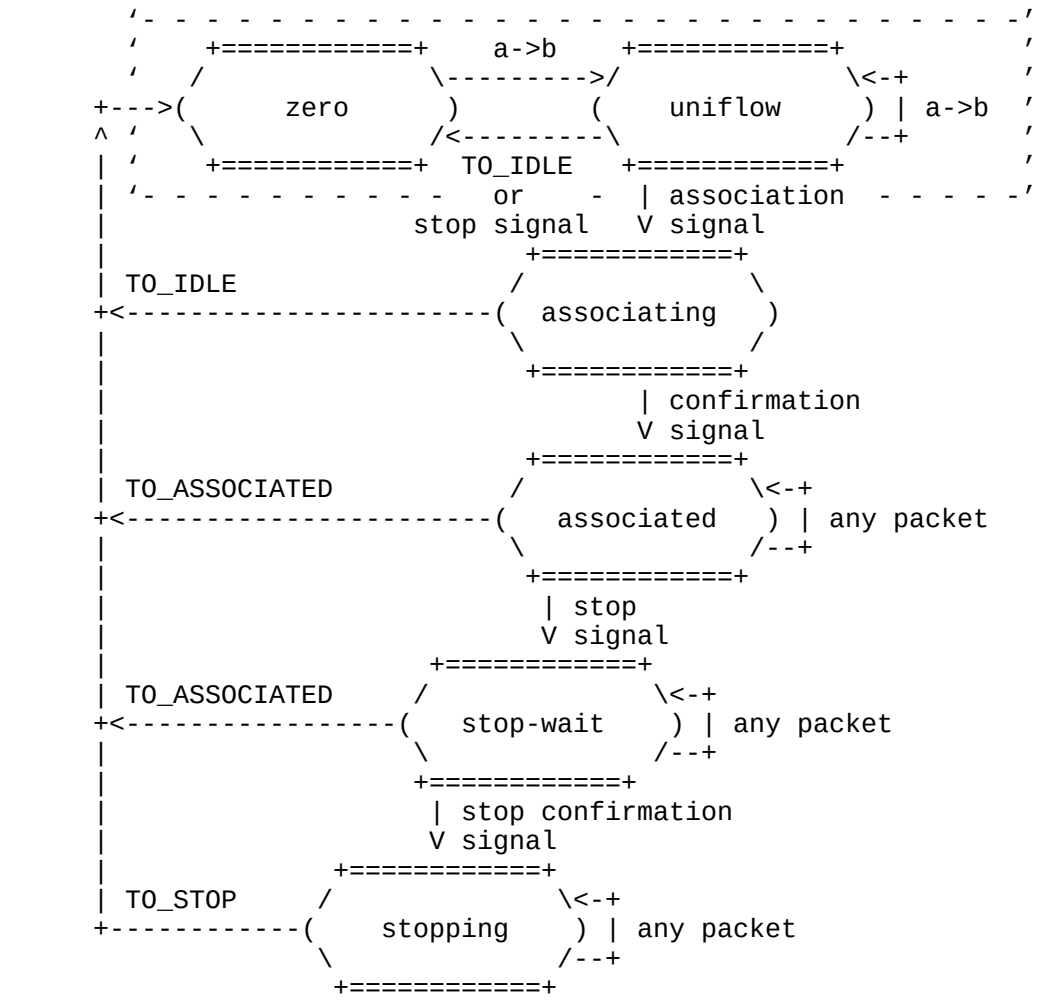


Figure 1: Transport-Independent State Machine for Stateful On-Path Devices

The three timeouts are defined as follows:

- TO_IDLE, the unidirectional idle timeout, can be considered equivalent to the idle timeout for transport protocols where the device has no information about session start and end (e.g. most UDP protocols).

- TO_ASSOCIATED, the bidirectional idle timeout, can be considered equivalent to the timeout for transport protocols where the device has information about session start and end (e.g. TCP).

- o TO_STOP is the teardown timeout: how long the device will account additional packets to a flow after confirming a close signal, ensuring retransmitted and/or reordered close signal don't lead to the spurious creation of new flow state.

Selection of timeouts is a configuration and implementation detail, but generally TO_STOP <= TO_IDLE << TO_ASSOCIATED; see [IMC-GATEWAYS] for an analysis of the magnitudes of these timeouts in presently deployed gateway devices.

### 3.1. Uniflow States

Every packet received by a device keeping per-flow state must associate that packet with a flow (see Section 4.1). When a device receives a packet associated with a flow it has no state for, and it is configured to forward the packet instead of dropping it, it moves that flow from the zero state into the uniflow state and starts a timer TO_IDLE. It resets this timer for any additional packet it forwards in the same direction as long as the flow remains in the uniflow state. When timer TO_IDLE expires on a flow in the uniflow state, the device drops state for the flow and performs any processing associated with doing so: tearing down NAT bindings, stopping associated firewall pinholes, exporting flow information, and so on. The device may also drop state on a stop signal, if observed.

Some devices will only see one side of a communication, e.g. if they are placed in a portion of a network with asymmetric routing. These devices use only the zero and uniflow states (as marked in Figure 1.) In addition, true uniflows - protocols which are solely unidirectional (e.g. some applications over UDP) - will also use only the uniflow-only states. In either case, current devices generally don't associate much state with observed uniflows, and an idle timeout is generally sufficient to expire this state.

### 3.2. Biflow States

A uniflow transitions to the associating state when the device observes an association signal, and further to the associated state when the device observes a subsequent confirmation signal; see Section 4.2 for details. If the flow has not transitioned to from the associating to the associated state after TO_IDLE, the device drops state for the flow.

After transitioning to the associated state, the device starts a timer TO_ASSOCIATED. It resets this timer for any packet it forwards in either direction. The associated state represents a fully established bidirectional communication. When timer TO_ASSOCIATED

expires, the device assumes that the flow has shut down without signaling as such, and drops state for the flow, performing any associated processing. When a bidirectional stop signal (see Section 4.3) is confirmed, the flow transitions to the stopping state.

When a flow enters the stopping state, it starts a timer TO_STOP. While the stop signal should be the last packet on a flow, the TO_STOP timer ensures that reordered packets after the stop signal will be accounted to the flow. When this timer expires, the device drops state for the flow, performing any associated processing.

### 3.3. Additional States and Actions

This document is concerned only with states and transitions common to transport- and function- independent state maintenance. Devices may augment the transitions in this state diagram depending on their function. For example, a firewall that decides based on some information beyond the signals used by this state machine to shut down a flow may transition it directly to a blacklist state on shutdown. Or, a firewall may fail to forward additional packets in the uniflow state until an association signal is observed.

## 4. Abstract Signaling Mechanisms

The state machine in Section 3 requires four signals: a new flow signal, the first packet observed in a flow in the zero state; an association signal, allowing a device to verify that an endpoint wishes a bidirectional communication to be established or to continue; a confirmation signal, allowing a device to confirm that the initiator of a flow is reachable at its purported source address; and a stop signal, noting that an endpoint wishes to stop a bidirectional communication. Additional related signals may also be useful, depending on the function a device provides. There are a few different ways to implement these signals; here, we explore the properties of some potential implementations.

We assume the following general requirements for these signals; parallel to those given in [draft-trammell-plus-abstract-mech]:

- At least the endpoints can verify the integrity of the signals exposed, and shut down a transport association when that verification fails, in order to reduce the incentive for on-path devices to attempt to spoof these signals.

- Endpoints and devices on path can probabilistically verify that a originator of a signal is on-path.

### 4.1. Flow Identification

In order to keep per-flow state, each device using this state machine must have a function it can apply to each packet to be able to extract common properties to identify the flow it is associated with. In general, the set of properties used for flow identification on presently deployed devices includes the source and destination IP address, the source and destination transport layer port number, the transport protocol number. The differentiated services field [RFC2474] may also be included in the set of properties defining a flow, since it may indicate different forwarding treatment.

However, other protocols may use additional bits in their own headers for flow identification. In any case, a protocol implementing signaling for this state machine must specify the function used for flow identification.

### 4.2. Association and Confirmation Signaling

An association signal indicates that the endpoint that received the first packet seen by the device has indeed seen that packet, and is interested in continuing conversation with the sending endpoint. This signal is roughly an in-band analogue to consent signaling in ICE [RFC7675] that is carried to every device along the path.

A confirmation signal indicates that the endpoint that sent the first packet seen by the device is reachable at its purported source address, and is necessary to prevent spoofed or reflected packets from driving the state machine into the associated state. It is roughly equivalent to the final ACK in the TCP three-way handshake.

These two signals are related to each other, in that association requires the receiving endpoint of the first packet to prove it has seen that packet (or a subsequent packet), and to acknowledge it wants to continue the association; while confirmation requires the sending endpoint to prove it has seen the association token.

Transport-independent, path-verifiable association and confirmation signaling can be implemented using three values carried in the packet headers: an association token, a confirmation nonce, and an echo token.

The association token is a cryptographically random value generated by the endpoint initiating a connection, and is carried on packets in the uniflow state. When a receiving endpoint wishes to send an association signal, it generates an echo token from the association token using a well-known, defined function (e.g. a truncated SHA-256 hash), and generates a cryptographically random confirmation nonce.

The initiating endpoint sends a confirmation signal on the next packet it sends after receiving the confirmation nonce, by applying a function to the echo token and the confirmation nonce, and sending the result as a new association token.

Devices on path verify that the echo token corresponds to a previously seen association token to recognize an association signal, and recognize that an association token corresponds to a previously seen echo token and confirmation nonce to recognize an association signal.

If the association token and confirmation nonce are predictable, off-path devices can spoof association and confirmation signals. In choosing the number of bits for an association token, there is a tradeoff between per-packet overhead and state overhead at on-path devices, and assurance that an association token is hard to guess. This tradeoff must be evaluated at protocol design time.

There are a few considerations in choosing a function (or functions) to generate the echo token from the association token, to verify an echo token given an association token, and to derive a next association token from the echo token and confirmation nonce. The functions could be extremely simple (e.g., identity for the echo token and addition for the nonce) for ease of implementation even in extremely constrained environments. Using one-way functions (e.g., truncated SHA-256 hash to derive echo token from association token; XOR followed by truncated SHA-256 hash to derive association token from echo token and confirmation nonce) requires slightly more work from on-path devices, but the primitives will be available at any endpoint using an encrypted transport protocol. In any case, a concrete implementation of association and confirmation signaling must choose a set of functions, or mechanism for unambiguously choosing one, at both endpoints as well as along the path.

#### 4.2.1. Start-of-flow versus continual signaling

There are two possible points in the design space here: these signals could be continually exposed throughout the flow, or could be exposed only on the first few packets of a connection (those corresponding to the cryptographic and/or transport state handshakes in the overlying protocols).

In the former case, an on-path device could re-establish state in the middle of a flow; e.g. due to a reboot of the device, due to a NAT association change without the endpoints' knowledge, or due to idle periods longer than the TO_ESTABLISHED timeout value. The on-path device would receive no special information about which packets were associated with the start of association. In this case, the series

of exposed association tokens, echo tokens, and confirmation nonces can also be observed to derive a running round-trip time estimate for the flow.

In the latter case, an on-path device would need to observe the start of the flow to establish state, and would be able to distinguish connection-start packets from other packets.

### 4.3. Bidirectional Stop Signaling

The transport-independent state machine uses bidirectional stop signaling to tear down state. This requires a stop signal to be observed in one direction, and a stop confirmation signal to be observed in the other, to complete tearing down an association.

A stop signal is directly carried or otherwise encoded in the protocol header to indicate that a flow is ending, whether normally or abnormally, and that state associated with the flow should be torn down. Upon decoding a stop signal, a device on path should move the flow from uniflow state to zero, or from associated state to stop-wait state, to wait for a confirmation signal in the other direction. While in stop-wait state, state will be maintained until a timer set to TO_ASSOCIATED expires, with any packet forwarded in either direction reseting the timer.

A stop confirmation signal is directly carried or otherwise encoded in the protocol header to indicate that the endpoint receiving the stop signal confirms that the stop signal is valid. The stop confirmation signal contains some assurance that the far endpoint has seen the stop signal. When a stop confirmation signal is observed in the opposite direction from the stop signal, a device on path should move the flow from stop-wait state to stopping state. The flow will then remain in stopping state until a timer set to TO_STOP has expired, after which state for the flow will be dropped. The stopping timeout TO_STOP is intended to ensure that any packets reordered in delivery are accounted to the flow before state for it is dropped.

We assume the encoding of stop and stop confirmation signals into a packet header, as with all other signals, is integrity protected end-to-end. Stop signals, as association signals, could be forged by a single on-path device. However, unless a stop confirmation signal that can be associated with the stop signal is observed in the other direction, the flow remains in stop-wait state, during which state is maintained and packets continue to be forwarded in both directions. So this attack is of limited utility; an attacker wishing to inject state teardown would need to control at least one on-path device on

each side of a target device to spoof both stop and corresponding stop confirmation signals.

#### 4.3.1. Authenticated Stop Signaling

Additionally, the stop and stop confirmation signals could be designed to authenticate themselves. Each endpoint could reveal a stop hash during the initial association, which is the result of a chosen cryptographic hash function applied to a stop token which that endpoint keeps secret. An endpoint wishing to end the association then reveals the stop token, which can be verified both by the far endpoint and devices on path which have cached the stop hash to be authentic. A stop confirmation signal additionally contains information derived from the initiating stop signal's stop token, as further assurance that the stop token was observed by the far endpoint.

### 4.4. Separate Utility

Although all of these signals are required to drive the state machine described by this document, note that association/confirmation and bidirectional stop signaling have separate utility. A transport protocol may expose the end of a flow without any proof of association or confirmation of return routability of the initiator. Alternately, the transport protocol could rely on short timeouts to clean up stale state on path, while exposing continuous association and confirmation signals to quickly reestablish state.

## 5. Deployment Considerations

The state machine defined in this document is most useful when implemented in a single instantiation (wire format for signals, and selection of functions for deriving values to be exposed and verified) by multiple transport protocols. It is intended for use with protocols that encrypt their transport- layer headers, and that are encapsulated within UDP, as is the case with QUIC [I-D.ietf-quic-transport]. Definition of that instantiation is out of scope for the present revision of this document.

The following subsections discuss incentives for deployment of this state machine both at middleboxes and at endpoints.

### 5.1. Middlebox Deployment

The state machine defined herein is designed to replace TCP state-tracking for firewalls and NAT devices. When encrypted transport protocols encapsulated in UDP adopt a set of signals and a wire format for those signals to drive this state machine, these

middleboxes could continue using TCP-like logic to handle those UDP flows. Recognizing the wire format used by those signals would allow these middleboxes to distinguish "UDP with an encrypted transport" from undifferentiated UDP, and to treat the former case more like TCP, providing longer timeouts for established flows, as well as stateful defense against spoofed or reflected garbage traffic.

### 5.2. Endpoint Deployment

An encrypted, UDP-encapsulated transport protocol has two primary incentives to expose these signals. First, allowing firewalls on networks that generally block UDP (about 3-5% of Internet-connected networks, depending on the study) to distinguish "UDP with an encrypted transport" traffic from other UDP traffic may result in less blocking of that traffic. Second, the difference between the timeouts TO_IDLE and TO_ASSOCIATED, as well as the continuous state establishment possible with some instantiations of the association and confirmation signals, would allow these transport protocols to send less unproductive keepalive traffic for long-lived, sparse flows.

While both of these advantages require middleboxes on path to recognize and use the signals driving this state machine, we note that content providers driving the deployment of this protocols are also operators of their own content provision networks, and that many of the benefits of encrypted- encapsulated transport firewalls will accrue to them, giving these content providers incentives to deploy both endpoints and middleboxes.

## 6. Signal mappings for transport protocols

We now show how this state machine can be driven by signals available in TCP and QUIC.

### 6.1. Signal mapping for TCP

A mapping of TCP flags to transitions in to the state machine in Section 3 shows how devices currently using a model of the TCP state machine can be converted to use this state machine.

TCP [RFC0793] provides start-of-flow association only. A packet with the SYN and ACK flags set in the absence of the FIN or RST flags, and an in-window acknowledgment number, is synonymous with the association signal. A packet with the ACK flag set in the absence of the FIN or RST flags after an initial SYN, and an in-window acknowledgment number, is synonymous with the confirmation signal. For a typical TCP flow:

1. The initial SYN places the flow into uniflow state,

2. The SYN-ACK sent in reply acts as a association signal and places the flow into associating state,

3. The ACK sent in reply acts as a confirmation signal and places the flow into associated state,

4. The final FIN is a stop signal, and

5. the ACK of the final FIN is a stop confirmation signal, moving the flow into stopping state.

Note that abormally closed flows (with RST) do not provide stop confirmation, and are therefore not provided for by this state machine. Due to TCP's support for half-closed flows, additional state modeling is necessary to extract a stop signal from the final FIN.

Note also that the association and stop signals derived from the TCP header are not integrity protected, and association and confirmation signals based on in-window ACK are not particularly resistant to off-path attacks [IMC-TCP]. The state machine is therefore more susceptible to manipulation when used with vanilla TCP as when with a transport protocol providing full integrity protection for its headers end-to-end.

### 6.2. Signal mapping for QUIC

QUIC [I-D.ietf-quic-transport] is a moving target; however, signals for driving this state machine are fundamentally compatible with the protocol's design and could easily be added to the protocol specification.

Specifically, QUIC's handshake is visible to on-path devices, as it begins with an unencrypted version negotiation which exposes a 64-bit connection ID, which can serve as an association and echo token as in Section 4.2. The function of the confirmation nonce is not fully exposed to the path at this point, but could be implemented by exposing information from the proof of source address ownership (section 7.4 of [I-D.ietf-quic-transport]) or via echoing the random initial packet number (as suggested by https://github.com/quicwg/base-drafts/pull/391).

The addition of a public reset signal that would act as a stop signal as in Section 4.3 is presently under discussion within the QUIC working group; the proposal for self-authenticating public reset at

https://github.com/quicwg/base-drafts/pull/20 inspired the addition of Section 4.3.1 to this document.

## 7. IANA Considerations

This document has no actions for IANA.

## 8. Security Considerations

This document defines a state machine for transport-independent state management on middleboxes, using in-band signaling, to replace the commonly- implemented current practice of incomplete TCP state modeling on these devices. It defines new signals for state management. While these signals can be spoofed by any device on path that observes traffic in both directions, we presume the presence of end-to-end integrity protection of these signals provided by the upper-layer transport driving them. This allows such spoofing to be detected and countered by endpoints, reducing the threat from on-path devices to connection disruption, which such devices are trivially placed to perform in any case.

## 9. Acknowledgments

Thanks to Christian Huitema for discussions leading to this document, and to Andrew Yourtchenko for the feedback. The mechanism for using a revealed value to prove ownership of a stop token was inspired by Eric Rescorla's suggestion to use a fundamentally identical mechanism for the QUIC public reset.

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

## 10. References

### 10.1. Normative References

[RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, DOI 10.17487/RFC5103, January 2008, <https://www.rfc-editor.org/info/rfc5103>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, DOI 10.17487/RFC7398, February 2015, <https://www.rfc-editor.org/info/rfc7398>.

## 10.2. Informative References

[draft-trammell-plus-abstract-mech]
Trammell, B., "Abstract Mechanisms for a Cooperative Path Layer under Endpoint Control", September 2016.

[I-D.hardie-path-signals]
Hardie, T., "Path signals", draft-hardie-path-signals-01 (work in progress), May 2017.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-07 (work in progress), October 2017.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-07 (work in progress), October 2017.

[IMC-GATEWAYS]
Hatonen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolahti, P., and M. Kojo, "An experimental study of home gateway characteristics (Proc. ACM IMC 2010)", October 2010.

[IMC-TCP] Luckie, M., Beverly, R., Wu, T., Allman, M., and k. claffy, "Resilience of Deployed TCP to Blind Attacks. (Proc. ACM IMC 2015)", October 2015.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[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, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <https://www.rfc-editor.org/info/rfc7675>.

## Authors' Addresses

Mirja Kuehlewind
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch

Joe Hildebrand

Email: hildjj@cursive.net