A Vocabulary of Path Properties
draft-enghardt-panrg-path-properties-02

Abstract

This document defines and categorizes information about Internet paths that an entity, such as a host, might have or want to have. This information is expressed as properties of paths between two hosts.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

Because the current Internet provides an IP-based best-effort bit pipe, hosts have little information about paths to other hosts. A Path Aware Network exposes information about one or multiple paths through the network to hosts or the network infrastructure.

It is impossible to provide an exhaustive list of path properties, as with every new technology and protocol, novel properties might become relevant. In this document, we specify a set of path properties which might be useful in the following use cases: Traffic policies, network monitoring, and path selection.

- Traffic policies: Entities such as network operators or end users may want to define traffic policies leveraging path awareness. Such policies can allow or disallow sending traffic over specific networks or nodes, select an appropriate protocol depending on the capabilities of the on-path devices, or adjust protocol parameters to an existing path. An example of a traffic policy is a video streaming application choosing an (initial) video quality based on the achievable data rate, or the monetary cost of the link using a volume-based or flat-rate cost model. Another example is an enterprise network where all traffic has to go through a firewall, in which case the host needs to be aware of on-path firewalls.

- Network monitoring: Network operators can use path properties (e.g., measured by on-path devices), to observe Quality of Service (QoS) characteristics of recent end-user traffic, and identify potential problems with their network early on, before the end-user complains.

- Path selection: In some cases, entities can choose to use a certain path (or subset of paths) from a set of paths to achieve a specific goal. As the possible benefits of a well chosen path varies based on the goal, as a baseline, a path selection
algorithm should aim to not perform worse than the default case most of the time. Depending on the goal, an entity may prefer paths with different properties, e.g., retrieving a small webpage as quickly as possible requires low latency paths, or retrieving a large file in a peer-to-peer network requires paths with high achievable data rate. Additionally, there may be trade-offs between path properties (e.g., latency and data rate), and entities may influence these trade-offs with their choices. A network (e.g., an AS) can adjust its path selection for internal or external routing based on the path properties. In BGP, the Multi Exit Discriminator (MED) attribute decides which path to choose if other attributes are equal; in a path aware network, instead of using this single MED value, other properties such as maximum or available/expected data rate could additionally be used to improve load balancing. A host might be able to select between a set of paths, either if there are several paths to the same destination (e.g., if the host is a mobile device with two wireless interfaces, both providing a path), or if there are several destinations, and thus several paths, providing the same service (e.g., Application-Layer Traffic Optimization (ALTO) [RFC5693], an application layer peer-to-peer protocol allowing hosts a better-than-random peer selection). Care needs to be taken when selecting paths based on path properties, as path properties that were previously measured may have become outdated and, thus, useless to predict the path properties of packets sent now.

Such path properties may be relatively dynamic, e.g. current Round Trip Time, close to the origin, e.g. nature of the access technology on the first hop, or far from the origin, e.g. list of ASes traversed.

Usefulness over time is fundamentally different for dynamic and non-dynamic properties. The merit of a momentary measurement of a dynamic path property diminishes greatly as time goes on, e.g. the merit of an RTT measurement from a few seconds ago is quite small, while a non-dynamic path property might stay relevant, e.g. a NAT can be assumed to stay on a path during the lifetime of a connection, as the removal of the NAT would break the connection.

Non-dynamic properties are further separated into (local) domain properties related to the first few hops of the connection, and backbone properties related to the remaining hops. Domain properties expose a high amount of information to hosts and strongly influence the connection behavior while there is little influence and information about backbone properties.
Dynamic properties are not separated into domain and backbone properties, since most of these properties are defined for a complete path and it is difficult and seldom useful to define them on part of the path. There are exceptions such as dynamic wireless access properties, but these do not justify separation into different categories.

This document addresses the first of the questions in Path-Aware Networking [I-D.irtf-panrg-questions], which is a product of the PANRG in the IRTF.

2. Terminology

Node: An entity which processes packets, e.g., sends, receives, forwards, or modifies them.

Host: A node that processes packets that are explicitly addressed to itself.

Router: A node that processes packets that are not explicitly addressed to itself.

Link: A medium or communication facility that connects two or more nodes with each other and enables them to exchange packets. A link can be physical, e.g., a WiFi network which connects an Access Point to stations, or virtual, e.g., a virtual switch which connects two virtual machines hosted on the same physical machine.

Path element: Either a node or a link.

Path: A sequence of adjacent path elements, alternating between nodes and links, starting and ending with a host. A path can be viewed as an abstraction on a specific layer, omitting lower layer path elements. For example, a router implementing IPv6 may be a path element on a path when considering the network layer. If this router does not implement transport layer functionality, it is hidden when a higher layer, such as the transport or application layer, is considered. In the case of multicast or broadcast, a single packet may be sent over multiple paths at once – one path for each combination of sending and receiving host.

Subpath: Given a path, a subpath is a sequence of adjacent path elements of this path, starting and ending with a node.

Flow: One or multiple packets which are traversing the same subpath or path. For example, a flow can consist of all packets sent within a TCP session with the same five-tuple between two hosts, or it can consist of all packets sent on the same physical link.
Property: A trait of one or a sequence of path elements, or a trait of a flow with respect to one or a sequence of path elements. An example of a link property is the maximum data rate that can be sent over the link. An example of a node property is the administrative domain that the node belongs to. An example of a property of a flow with respect to a subpath is the aggregated one-way delay of the flow being sent from one node to another node over a subpath. A property is thus described by a tuple containing the sequence of path elements, the flow or an empty set if no packets are relevant for the property, the name of the property (e.g., maximum data rate), and the value of the property (e.g., 1Gbps).

Aggregated property: A collection of multiple values of a property into a single value, according to a function. A property can be aggregated over multiple path elements (i.e., a path), e.g., the MTU of a path as the minimum MTU of all links on the path, over multiple packets (i.e., a flow), e.g., the median one-way latency of all packets between two nodes, or over both, e.g., the mean of the queuing delays of a flow on all nodes along a path. The aggregation function can be numerical, e.g., median, sum, minimum, it can be logical, e.g., "true if all are true", "true if at least 50\% of values are true", or an arbitrary function which maps multiple input values to an output value.

Measured property: A property that is observed for a specific path element or path, e.g., using measurements. For example, the one-way delay of a specific packet can be measured.

Estimated property: An approximate calculation or judgment of the value of a property. For example, an estimated property may describe the expected median one-way latency of packets sent on a path within the next second. An estimated property includes the reliability of the estimate. The notion of reliability depends on the property. For example, it may be the confidence level and interval for numerical properties or the likelihood that a property holds for non-numerical properties.

3. Domain Properties

Domain path properties relate to path elements within the first hop or the first few hops, which are usually in the same administrative domain as a host considering them.

Due to the potential physical proximity and pre-existing trust or contractual relationships between hosts and path elements within the same domain, domain properties may be more accessible to the host than other properties.
Furthermore, hosts may be able to influence both which domain they are in and which path elements in this domain to connect to, and they may be able to influence the properties of path elements within this domain. For example, a user might select between multiple potential adjacent links by selecting between multiple available WiFi Access Points. Or when connected to an Access Point, the user may move closer to enable their device to use a different access technology, potentially increasing the data rate available to the device. Another example is a user changing their data plan to reduce the Monetary Cost to transmit a given amount of data across a network.

Access Technology: The physical or link layer technology used for transmitting or receiving a flow on one or multiple path elements in the same domain. The Access Technology may be given in an abstract way, e.g., as a WiFi, Wired Ethernet, or Cellular link. It may also be given as a specific technology, e.g., as a 2G, 3G, 4G, or 5G cellular link, or an 802.11a, b, g, n, or ac WiFi link. Other path elements relevant to the access technology may include on-path devices, such as elements of a cellular backbone network. Note that there is no common registry of possible values for this property.

Monetary Cost: The price to be paid to transmit a specific flow across a subpath.

4. Backbone Properties

Backbone path properties relate to path elements not within the same domain as a host considering them, thus, in the backbone from the host’s point of view.

Typically, backbone properties are less accessible to a host than domain properties, due to the potential increased distance and the lack of pre-existing trust or contractual relationship.

Additionally, hosts are less likely to be able to influence which path elements form their path in the backbone, as well as their properties.

Some path properties relate to the entire path or to subpaths, part of which often lies outside of a host’s domain. Thus, such properties are listed as Backbone Properties.

Presence of a certain network function on the path: Indicates that a node performs a certain network function on a flow, e.g., whether the node acts as a proxy, as a firewall, or performs Network Address Translation (NAT). This node may be either in the same domain as the host or in a different domain, i.e., the backbone.
Administrative Entity: The administrative entity, e.g., the AS, to which a path element or subpath belongs.

Disjointness: For a set of two paths, the number of shared path elements can be a measure of intersection (e.g., Jaccard coefficient, which is the number of shared elements divided by the total number of elements). Conversely, the number of non-shared path elements can be a measure of disjointness (e.g., 1 - Jaccard coefficient). A multipath protocol might use disjointness of paths as a metric to reduce the number of single points of failure.

Path MTU: The maximum size, in octets, of an IP packet that can be transmitted without fragmentation on a subpath.

Transport Protocols available: Whether a specific transport protocol can be used to establish a connection over a path or subpath. A host may cache its knowledge about recent successfully established connections using specific protocols, e.g., a QUIC connection, or an MPTCP subflow.

Protocol Features available: Whether a specific protocol feature is available over a path or subpath, e.g., Explicit Congestion Notification (ECN), or TCP Fast Open.

5. Dynamic Properties

Dynamic path properties relate to the transmission of an individual packet or of a flow over a subpath. Properties related to a path element which constitutes a single layer 2 domain are abstracted from the used physical and link layer technology, similar to [RFC8175].

 Typically, Dynamic Properties can be measured or approximated, and might be made available in an aggregated form, such as averages or minimums. Dynamic Path Properties can be measured by the host itself or by a different entity. See [ANRW18-Metrics] for a discussion of how to measure some dynamic path properties at the host.

Some dynamic properties are defined in different directions for the same path element, e.g., for transmitting and receiving packets.

Maximum Data Rate (Transmit/Receive): The theoretical maximum data rate, in bits per second, that can be achieved on a link, subpath, or path, for receiving or transmitting traffic.

Current Data Rate (Transmit/Receive): The data rate, in bits per second, at which a link is currently receiving or transmitting traffic.
Latency: The time delay between a node sending a packet and a different node on the same path receiving the same packet.

Latency variation: The variation of the latency within a flow.

Packet Loss: The percentage of packets within a flow which are sent by one node, but not received by a different node.

Congestion: Whether a protocol feature such as ECN has provided information that there currently is congestion on a path.

6. Security Considerations

If nodes are basing policy or path selection decisions on path properties, they need to rely on the accuracy of path properties that other devices communicate to them. In order to be able to trust such path properties, nodes may need to establish a trust relationship or be able to verify the authenticity, integrity, and correctness of path properties received from another node.

7. IANA Considerations

This document has no IANA actions.

8. Informative References

[ANRW18-Metrics]

[I-D.irtf-panrg-questions]

[RFC5693]

[RFC8175]
Acknowledgments

Thanks to the Path-Aware Networking Research Group for the discussion and feedback. Thanks to Adrian Perrig and Matthias Rost for the feedback. Thanks to Paul Hoffman for the editorial changes.

Authors’ Addresses

Theresa Enghardt
TU Berlin

Email: theresa@inet.tu-berlin.de

Cyrill Kraehenbuehl
ETH Zuerich

Email: cyrill.kraehenbuehl@inf.ethz.ch
Open Questions in Path Aware Networking
draft-irtf-panrg-questions-02

Abstract

This document poses open questions in path-aware networking, as a background for framing discussions in the Path Aware Networking proposed Research Group (PANRG). Path-aware networking has two aspects: the exposure of properties of available Internet paths to endpoints and applications running on them, and allowing endpoints and applications to use these properties to select paths through the Internet for their traffic.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 23, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction to Path-Aware Networking

In the current Internet architecture, the interdomain network layer provides an unverifiable, best-effort service: an application can assume that a packet with a given destination address will eventually be forwarded toward that destination, but little else. A transport layer protocol such as TCP can provide reliability over this best-effort service, and a protocol above the network layer such as IPsec AH [RFC4302] or TLS [RFC5246] can authenticate the remote endpoint. However, no explicit information about the path is available, and assumptions about that path sometimes do not hold, sometimes with serious impacts on the application, as in the case with BGP hijacking attacks.

By contrast, in a path-aware internetworking architecture, endpoints have the ability to select or influence the path through the network used by any given packet, and the network and transport layers explicitly expose information about the path or paths available between two endpoints to those endpoints and the applications running on them, so that they can make this selection.

Path selection provides transparency and control to applications and users of the network. Selection may be made at either the application layer or the transport layer. Path control at the packet level enables the design of new transport protocols that can leverage multipath connectivity across maximally-disjoint paths through the
Internet, even over a single physical interface. When exposed to applications, or to end-users through a system configuration interface, path control allows the specification of constraints on the paths that traffic should traverse, for instance to confound passive surveillance in the network core.

We note that this property of "path awareness" already exists in many Internet-connected networks in an intradomain context. Indeed, much of the practice of network engineering using encapsulation at layer 3 can be said to be "path aware", in that it explicitly assigns traffic at tunnel endpoints to a given path within the network. Path-aware internetworking seeks to extend this awareness across domain boundaries without resorting to overlays, except as a transition technology.

2. Questions

Realizing path-aware networking requires answers to a set of open research questions. This document poses these questions, as a starting point for discussions about how to realize path awareness in the Internet, and to direct future research efforts within the Path Aware Networking Research Group.

2.1. A Vocabulary of Path Properties

In order for information about paths to be exposed to an endpoint, and for the endpoint to make use of that information, it is necessary to define a common vocabulary for path properties. The elements of this vocabulary could include relatively static properties, such as the presence of a given node or service function on the path; as well as relatively dynamic properties, such as the current values of metrics such as loss and latency.

This vocabulary must be defined carefully, as its design will have impacts on the expressiveness of a given path-aware internetworking architecture. This expressiveness also exhibits tradeoffs. For example, a system that exposes node-level information for the topology through each network would maximize information about the individual components of the path at the endpoints at the expense of making internal network topology universally public, which may be in conflict with the business goals of each network's operator.

The first question: how are path properties defined and represented?
2.2. Discovery, Distribution, and Trustworthiness of Path Properties

Once endpoints and networks have a shared vocabulary for expressing path properties, the network must have some method for distributing those path properties to the endpoint. Regardless of how path property information is distributed to the endpoints, the endpoints require a method to authenticate the properties - to determine that they originated from and pertain to the path that they purport to.

Choices in distribution and authentication methods will have impacts on the scalability of a path-aware architecture. Possible dimensions in the space of distribution methods include in-band versus out-of-band, push versus pull versus publish-subscribe, and so on. There are temporal issues with path property dissemination as well, especially with dynamic properties, since the measurement or elicitation of dynamic properties may be outdated by the time that information is available at the endpoints, and interactions between the measurement and dissemination delay may exhibit pathological behavior for unlucky points in the parameter space.

The second question: how do endpoints get access to trustworthy path properties?

2.3. Supporting Path Selection

Access to trustworthy path properties is only half of the challenge in establishing a path-aware architecture. Endpoints must be able to use this information in order to select paths for traffic they send. As with the dissemination of path properties, choices made in path selection methods will also have an impact on the tradeoff between scalability and expressiveness of a path-aware architecture. One key choice here is between in-band and out-of-band control of path selection. Another is granularity of path selection (whether per packet, per flow, or per larger aggregate), which also has a large impact on the scalability/expressiveness tradeoff. Path selection must, like path property information, be trustworthy, such that the result of a path selection at an endpoint is predictable.

The third question: how can endpoints select paths to use for traffic in a way that can be trusted by both the network and the endpoints?

2.4. Interfaces for Path Awareness

In order for applications to make effective use of a path-aware networking architecture, the control interfaces presented by the network and transport layers must also expose path properties to the application in a useful way, and provide a useful set of paths among which the application can select. Path selection must be possible
based not only on the preferences and policies of the application developer, but of end-users as well. Also, the path selection interfaces presented to applications and end users will need to support multiple levels of granularity. Most applications’ requirements can be satisfied with the expression path selection policies in terms of properties of the paths, while some applications may need finer-grained, per-path control.

The fourth question: how can interfaces to the transport and application layers support the use of path awareness?

2.5. Implications of Path Awareness for the Data Plane

In the current Internet, the basic assumption that at a given time all traffic for a given flow will traverse a single path, for some definition of path, generally holds. In a path aware network, this assumption no longer holds. The absence of this assumption has implications for the design of protocols above any path-aware network layer.

For example, one advantage of multipath communication is that a given end-to-end flow can be "sprayed" along multiple paths in order to confound attempts to collect data or metadata from those flows for pervasive surveillance purposes [RFC7624]. However, the benefits of this approach are reduced if the upper-layer protocols use linkable identifiers on packets belonging to the same flow across different paths. Clients may mitigate linkability by opting to not re-use cleartext connection identifiers, such as TLS session IDs or tickets, on separate paths. The privacy-conscious strategies required for effective privacy in a path-aware Internet are only possible if higher-layer protocols such as TLS permit clients to obtain unlinkable identifiers.

The fifth question: how should transport-layer and higher layer protocols be redesigned to work most effectively over a path-aware networking layer?

2.6. What is an Endpoint?

The vision of path-aware networking articulated so far makes an assumption that path properties will be disseminated to endpoints on which applications are running (terminals with user agents, servers, and so on). However, incremental deployment may require that a path-aware network "core" be used to interconnect islands of legacy protocol networks. In these cases, it is the gateways, not the application endpoints, that receive path properties and make path selections for that traffic. The interfaces provided by this gateway are necessarily different than those a path-aware networking layer
provides to its transport and application layers, and the path property information the gateway needs and makes available over those interfaces may also be different.

The sixth question: how is path awareness (in terms of vocabulary and interfaces) different when applied to tunnel and overlay endpoints?

2.7. Operating a Path Aware Network

The network operations model in the current Internet architecture assumes that traffic flows are controlled by the decisions and policies made by network operators, as expressed in interdomain routing protocols. In a network providing path selection to the endpoints, however, this assumption no longer holds, as endpoints may react to path properties by selecting alternate paths. Competing control inputs from path-aware endpoints and the interdomain routing control plane may lead to more difficult traffic engineering or nonconvergent forwarding, especially if the endpoints’ and operators’ notion of the "best" path for given traffic diverges significantly.

A concept for path aware network operations will need to have clear methods for the resolution of apparent (if not actual) conflicts of intent between the network’s operator and the path selection at an endpoint. It will also need set of safety principles to ensure that increasing path control does not lead to decreasing connectivity; one such safety principle could be "the existence of at least one path between two endpoints guarantees the selection of at least one path between those endpoints."

The seventh question: how can a path aware network in a path aware internetwork be effectively operated, given control inputs from the network administrator as well as from the endpoints?

2.8. Deploying a Path Aware Network

The vision presented in the introduction discusses path aware networking from the point of view of the benefits accruing at the endpoints, to designers of transport protocols and applications as well as to the end users of those applications. However, this vision requires action not only at the endpoints but within the interconnected networks offering path aware connectivity. While the specific actions required are a matter of the design and implementation of a specific realization of a path aware protocol stack, it is clear that any path aware architecture will require network operators to give up some control of their networks over to endpoint-driven control inputs.
Here the question of apparent versus actual conflicts of intent arises again: certain network operations requirements may appear essential, but are merely accidents of the interfaces provided by current routing and management protocols. Incentives for deployment must show how existing network operations requirements are met through new path selection and property dissemination mechanisms.

The incentives for network operators and equipment vendors to do provide be made clear, in terms of a plan to transition [RFC8170] an internetwork to path-aware operation, one network and facility at a time.

The eighth question: how can the incentives of network operators and end-users be aligned to realize the vision of path aware networking?

3. Acknowledgments

Many thanks to Adrian Perrig, Jean-Pierre Smith, Mirja Kuehlewind, Olivier Bonaventure, Martin Thomson, Shwetha Bhandari, Chris Wood, Lee Howard, and Mohamed Boucadair for discussions leading to questions in this document, and for feedback on the document itself.

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.

4. References

4.1. Normative References


4.2. Informative References
[RFC7624]  Barnes, R., Schneier, B., Jennings, C., Hardie, T.,
          Trammell, B., Huitema, C., and D. Borkmann,
          "Confidentiality in the Face of Pervasive Surveillance: A
          Threat Model and Problem Statement", RFC 7624,
          DOI 10.17487/RFC7624, August 2015,

[RFC8170]  Thaler, D., Ed., "Planning for Protocol Adoption and
          Subsequent Transitions", RFC 8170, DOI 10.17487/RFC8170,

Author’s Address

Brian Trammell
Google
Gustav-Gull-Platz 1
8004 Zurich
Switzerland

Email: ietf@trammell.ch
Path Aware Networking: Obstacles to Deployment (A Bestiary of Roads Not Taken)
draft-irtf-panrg-what-not-to-do-03

Abstract

At the first meeting of the Path Aware Networking Research Group, the research group agreed to catalog and analyze past efforts to develop and deploy Path Aware technologies, most of which were unsuccessful, in order to extract insights and lessons for path-aware networking researchers.

This document contains that catalog and analysis.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 24, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
1. Introduction

At the first meeting of the Path Aware Networking Research Group [PANRG], at IETF 99 [PANRG-99], Oliver Bonaventure led a discussion of "A Decade of Path Awareness" [PATH-Decade], on attempts, which
were mostly unsuccessful for a variety of reasons, to exploit Path Aware technologies and achieve a variety of goals over the past decade. At the end of this discussion, two things were abundantly clear.

- The Internet community has accumulated considerable experience with many Path Aware technologies over a long period of time, and

- Although some Path Aware technologies have been successfully deployed (for example, Differentiated Services, or DiffServ [RFC2475]), most of these technologies haven’t seen widespread adoption. The reasons for non-adoption are many, and are worthy of study.

The meta-lessons from that experience were

- Path Aware Networking has been more Research than Engineering, so establishing an IRTF Research Group for Path Aware Networking is the right thing to do [RFC7418].

- Analyzing a catalog of past experience to learn the reasons for non-adoption would be a great first step for the Research Group.

Allison Mankin, as IRTF Chair, officially chartered the Path Aware Networking Research Group in July, 2018.

This document contains the analysis performed by that research group (see Section 2), based on that catalog (see Section 4).

1.1. A Note About Path-Aware Technologies Included In This Document

This document does not catalog every technology about Path Aware Networking that was not implemented and deployed. Instead, we include enough technologies to provide background for the lessons included in Section 2 to guide researchers and protocol engineers in their work.

No shame is intended for the technologies included in this document. As shown in Section 2, the quality of specific technologies had little to do with whether they were deployed or not. Based on the technologies cataloged in this document, it is likely that when these technologies were put forward, the proponents were trying to engineer something that could not be engineered without first carrying out research. Actual shame would be failing to learn from experience, and failing to share that experience with other networking researchers and engineers.
1.2. Venue for Discussion of this Document

(RFC Editor: please remove this section before publication)

Discussion of specific contributed experiences and this document in general should take place on the PANRG mailing list.

1.3. A Note for the Research Group

(RFC Editor: please remove this section before publication)

The editor and research group chairs are aware that the current version of this document is tilted toward transport-level Path Aware technologies, and would like to interact with other IETF protocol communities who have experience with Path Aware technologies.

It is worth looking at the Lessons Learned in Section 2 to see whether the Internet has changed in ways that would make some lessons less applicable for future protocol design.

1.4. A Note for the Editor

(Remove after taking these actions)

The to-do list for upcoming revisions includes

- Confirm that the Summary of Lessons Learned makes sense and is complete, in consultation with the Research Group.

- If the Research Group identifies technologies that provided lessons that aren’t included in Section 2, solicit contributions for those technologies.

- Provide better context for Section 2, to make sure that individual lessons aren’t considered in isolation, and to distinguish between impediments to deployment and blockers for deployment.

1.5. Architectural Guidance

As background for understanding the Lessons Learned contained in this document, the reader is encouraged to become familiar with the Internet Architecture Board’s documents on "What Makes for a Successful Protocol?" [RFC5218] and "Planning for Protocol Adoption and Subsequent Transitions" [RFC8170].

Although these two documents do not specifically target path-aware networking protocols, they are helpful resources for readers seeking to improve their understanding of considerations for successful
adoption and deployment of any protocol. For example, the Basic Success Factors described in Section 2.1 of [RFC5218] are helpful for readers of this document.

Because there is an economic aspect to decisions about deployment, the IAB Workshop on Internet Technology Adoption and Transition [ITAT] report [RFC7305] also provides food for thought.

Most of the Lessons Learned in Section 2 reflect considerations described in [RFC5218], [RFC7305], and [RFC8170].

2. Summary of Lessons Learned

This section summarizes the Lessons Learned from the contributed sections in Section 4.

Each Lesson Learned is tagged with one or more contributions that encountered this obstacle as a significant impediment to deployment. Other contributed technologies may have also encountered this obstacle, but this obstacle may not have been the biggest impediment to deployment.

It is useful to notice that sometimes an obstacle might impede deployment, while at other times, the same obstacle might prevent deployment entirely. The research group discussed distinguishing between obstacles that impede and obstacles that prevent, but it appears that the boundary between "impede" and "prevent" can shift over time - some of the Lessons Learned are based on both Path Aware technologies that were not deployed, and Path Aware technologies that were deployed, but were not deployed widely or quickly. See Section 4.6 and Section 4.6.3 as one example of this shifting boundary.

- The benefit of Path Awareness must be great enough to overcome entropy for already-deployed devices. The colloquial American English expression, "If it ain't broke, don't fix it" is a "best current practice" on today's Internet. (See Section 4.3, Section 4.5, and Section 4.4).

- Providing benefits for early adopters can be key - if everyone must deploy a technology in order for the technology to provide benefits, or even to work at all, the technology is unlikely to be adopted. (See Section 4.2 and Section 4.3).

- Adaptive end-to-end protocol mechanisms may respond to feedback quickly enough that the additional realizable benefit from a new Path Aware mechanism may be much smaller than anticipated (see Section 4.3 and Section 4.5).
Follow the money. If operators can’t charge for a Path Aware technology to recover the costs of deploying it, the benefits to the operator must be really significant. Corollary: If operators charge for a Path Aware technology, the benefits to the user must be significant enough to justify the cost. (See Section 4.5, Section 4.1, and Section 4.2).

Impact of a Path Aware technology requiring changes to operational practices can prevent deployment of promising technology. (See Section 4.6, including Section 4.6.3).

Per-connection state in intermediate devices can be an impediment to adoption and deployment. This is especially true as we move from the edge of the network into the routing core (See Section 4.1 and Section 4.2).

Many modern routers, especially high-end routers, have not been designed to make heavy use of in-band mechanisms such as IPv4 and IPv6 Router Alert Options (RAO), so operators can be reluctant to deploy technologies that rely on these mechanisms. (See Section 4.7).

If the endpoints do not have any trust relationship with the intermediate devices along a path, operators can be reluctant to deploy technologies that rely on endpoints sending unauthenticated control signals to routers. (See Section 4.2 and Section 4.7. We also note this still remains a factor hindering deployment of DiffServ).

If intermediate devices along the path can’t be trusted, it’s unlikely that endpoints will rely on signals from intermediate devices to drive changes to endpoint behaviors. (See Section 4.5, Section 4.4). The lowest level of trust is sufficient for a device issuing a message to confirm that it has visibility of the packets on the path it is seeking to control [RFC8085] (e.g., an ICMP message included a quoted packet from the source). A higher level of trust can arise when a network device could have a long or short term trust relationship with the sender it controls.

Because the Internet is a distributed system, if the distance that information from distant hosts and routers travels to a Path Aware host or router is sufficiently large, the information may no longer represent the state and situation at the distant host or router when it is received. In this case, the benefit that a Path Aware technology provides likely decreases. (See Section 4.3).

Providing a new feature/signal does not mean that it will be used. Endpoint stacks may not know how to effectively utilize Path-Aware
transport protocol technologies, because the technology may require information from applications to permit them to work effectively, but applications may not a-priori know that information. Even if the application does know that information, the de-facto API has no way of signaling the expectations of applications for the network path. Providing this awareness requires an API that signals more than the packets to be sent. TAPS is exploring such an API [TAPS-WG], yet even with such an API, policy is needed to bind the application expectations to the network characteristics. (See Section 4.1 and Section 4.2).

3. Template for Contributions

There are many things that could be said about the Path Aware networking technologies that have been developed. For the purposes of this document, contributors are requested to provide

- the name of a technology, including an abbreviation if one was used
- if available, a long-term pointer to the best reference describing the technology
- a short description of the problem the technology was intended to solve
- a short description of the reasons why the technology wasn’t adopted
- a short statement of the lessons that researchers can learn from our experience with this technology.

This document is being built collaboratively. To contribute your experience, please send a Github pull request to https://github.com/panrg/draft-dawkins-panrg-what-not-to-do.

4. Contributions

Additional contributions that provide Lessons Learned beyond those already captured in Section 2 are welcomed.

4.1. Stream Transport (ST, ST2, ST2+)

The suggested references for IntServ are:

- ST - A Proposed Internet Stream Protocol [IEN-119]
- Experimental Internet Stream Protocol, Version 2 (ST-II) [RFC1190]
The first version of Stream Transport, ST [IEN-119], was published in the late 1970’s and was implemented and deployed on the ARPANET at small scale. It was used throughout the 1980’s for experimental transmission of voice, video, and distributed simulation.

The second version of the ST specification (ST2) [RFC1190] [RFC1819] was an experimental connection-oriented internetworking protocol that operated at the same layer as connectionless IP. ST2 packets could be distinguished by their IP header protocol numbers (IP, at that time, used protocol number 4, while ST2 used protocol number 5).

ST2 used a control plane layered over IP to select routes and reserve capacity for real-time streams across a network path, based on a flow specification communicated by a separate protocol. The flow specification could be associated with QoS state in routers, producing an experimental resource reservation protocol. This allowed ST2 routers along a path to offer end-to-end guarantees, primarily to satisfy the QoS requirements for realtime services over the Internet.

4.1.1. Reasons for Non-deployment

Although implemented in a range of equipment, ST2 was not widely used after completion of the experiments. It did not offer the scalability and fate-sharing properties that have come to be desired by the Internet community.

The ST2 protocol is no longer in use.

4.1.2. Lessons Learned.

As time passed, the trade-off between router processing and link capacity changed. Links became faster and the cost of router processing became comparatively more expensive.

The ST2 control protocol used "hard state" - once a route was established, and resources were reserved, routes and resources existing until they were explicitly released via signaling. A soft-state approach was thought superior to this hard-state approach, and led to development of the IntServ model described in Section 4.2.
4.2. Integrated Services (IntServ)

    The suggested references for IntServ are:
    
    o RFC 1633 Integrated Services in the Internet Architecture: an
      Overview [RFC1633]
    
    o RFC 2211 Specification of the Controlled-Load Network Element
      Service [RFC2211]
    
    o RFC 2212 Specification of Guaranteed Quality of Service [RFC2212]
    
    o RFC 2215 General Characterization Parameters for Integrated
      Service Network Elements [RFC2215]
    
    o RFC 2205 Resource ReSerVation Protocol (RSVP) [RFC2205]

    In 1994, when the IntServ architecture document [RFC1633] was
    published, real-time traffic was first appearing on the Internet. At
    that time, bandwidth was still a scarce commodity. Internet Service
    Providers built networks over DS3 (45 Mbps) infrastructure, and sub-
    rate (< 1 Mpbs) access was common. Therefore, the IETF anticipated a
    need for a fine-grained QoS mechanism.

    In the IntServ architecture, some applications can require service
    guarantees. Therefore, those applications use the Resource
    Reservation Protocol (RSVP) [RFC2205] to signal QoS reservations
    across network paths. Every router in the network maintains per-flow
    soft-state to a) perform call admission control and b) deliver
    guaranteed service.

    Applications use Flow Specification (Flow Specs) [RFC2210] to
    describe the traffic that they emit. RSVP reserves capacity for
    traffic on a per Flow Spec basis.

4.2.1. Reasons for Non-deployment

    Although IntServ has been used in enterprise and government networks,
    IntServ was never widely deployed on the Internet because of its
    cost. The following factors contributed to operational cost:
    
    o IntServ must be deployed on every router that is on a path where
      IntServ is to be used
    
    o IntServ maintained per flow state

    As IntServ was being discussed, the following occurred:
For many expected uses, it became more cost effective to solve the QoS problem by adding bandwidth. Between 1994 and 2000, Internet Service Providers upgraded their infrastructures from DS3 (45 Mbps) to OC-48 (2.4 Gbps). This meant that even if an endpoint was using IntServ in an IntServ-enabled network, its requests would never be denied, so endpoints and Internet Service Providers had little reason to enable IntServ.

DiffServ [RFC2475] offered a more cost-effective, albeit less fine-grained, solution to the QoS problem.

4.2.2. Lessons Learned.

The following lessons were learned:

- Any mechanism that requires a router to maintain per-flow state is not likely to succeed, unless the additional cost for offering the feature can be recovered from the user.

- Any mechanism that requires an operator to upgrade all of its routers is not likely to succeed, unless the additional cost for offering the feature can be recovered from the user.

In environments where IntServ has been deployed, trust relationships with endpoints are very different from trust relationships on the Internet itself, and there are often clearly-defined hierarchies in Service Level Agreements (SLAs), and well-defined transport flows operating with pre-determined capacity and latency requirements over paths where capacity or other attributes are constrained.

IntServ was never widely deployed to manage capacity across the Internet. However, the technology that it produced was deployed for reasons other than bandwidth management. RSVP is widely deployed as an MPLS signaling mechanism. BGP reuses the RSVP concept of Filter Specs to distribute firewall filters, although they are called Flow Spec Component Types in BGP [RFC5575].

4.3. Quick-Start TCP

The suggested references for Quick-Start TCP are:

- RFC 4782 Quick-Start for TCP and IP [RFC4782]

- Determining an appropriate initial sending rate over an underutilized network path [SAF07]

- Fast Startup Internet Congestion Control for Broadband Interactive Applications [Sch11]
Quick-Start [RFC4782] is an Experimental TCP extension that leverages support from the routers on the path to determine an allowed initial sending rate for a path through the Internet, either at the start of data transfers or after idle periods. A corresponding mechanism was also specified for other congestion controllers (e.g., "Quick-Start for the Datagram Congestion Control Protocol (DCCP)" [RFC5634]). In these cases, a sender cannot easily determine an appropriate initial sending rate, given the lack of information about the path. The default TCP congestion control therefore uses the time-consuming slow-start algorithm. With Quick-Start, connections are allowed to use higher initial sending rates if there is significant unused bandwidth along the path, and if the sender and all of the routers along the path approve the request.

By examining the Time To Live (TTL) field in Quick-Start packets, a sender can determine if routers on the path have approved the Quick-Start request. However, this method is unable to take into account the routers hidden by tunnels or other network devices invisible at the IP layer.

The protocol also includes a nonce that provides protection against cheating routers and receivers. If the Quick-Start request is explicitly approved by all routers along the path, the TCP host can send at up to the approved rate; otherwise TCP would use the default congestion control. Quick-Start requires modifications in the involved end-systems as well in routers. Due to the resulting deployment challenges, Quick-Start was only proposed in [RFC4782] for controlled environments.

The Quick-Start mechanism is a lightweight, coarse-grained, in-band, network-assisted fast startup mechanism. The benefits are studied by simulation in a research paper [SAF07] that complements the protocol specification. The study confirms that Quick-Start can significantly speed up mid-sized data transfers. That paper also presents router algorithms that do not require keeping per-flow state. Later studies [Sch11] comprehensively analyzes Quick-Start with a full Linux implementation and with a router fast path prototype using a network processor. In both cases, Quick-Start could be implemented with limited additional complexity.
4.3.1. Reasons for Non-deployment

However, experiments with Quick-Start in [Sch11] revealed several challenges:

- Having information from the routers along the path can reduce the risk of congestion, but cannot avoid it entirely. Determining whether there is unused capacity is not trivial in actual router and host implementations. Data about available capacity visible at the IP layer may be imprecise, and due to the propagation delay, information can already be outdated when it reaches a sender. There is a trade-off between the speedup of data transfers and the risk of congestion even with Quick-Start. This could be mitigated by only allowing Quick-Start to access a proportion of the unused capacity along a path.

- For scalable router fast path implementation, it is important to enable parallel processing of packets, as this is a widely used method e.g. in network processors. One challenge is synchronization of information between different packets, which should be avoided as much as possible.

- Only some types of application traffic can benefit from Quick-Start. Capacity needs to be requested and discovered. The discovered capacity needs to be utilized by the flow, or it implicitly becomes available for other flows. Failing to use the requested capacity may have already reduced the pool of Quick-Start capacity that was made available to other competing Quick-Start requests. The benefit is greatest when senders use this only for bulk flows and avoid sending unnecessary Quick-Start requests, e.g. for flows that only send a small amount of data. Choosing an appropriate request size requires application-internal knowledge that is not commonly expressed by the transport API. How a sender can determine the rate for an initial Quick-Start request is still a largely unsolved problem.

There is no known deployment of Quick-Start for TCP or other IETF transports.

4.3.2. Lessons Learned

Some lessons can be learned from Quick-Start. Despite being a very light-weight protocol, Quick-Start suffers from poor incremental deployment properties, both regarding the required modifications in network infrastructure as well as its interactions with applications. Except for corner cases, congestion control can be quite efficiently performed end-to-end in the Internet, and in modern stacks there is
not much room for significant improvement by additional network support.

After publication of the Quick-Start specification, there have been large-scale experiments with an initial window of up to 10 MSS [RFC6928]. This alternative "IW10" approach can also ramp-up data transfers faster than the standard congestion control, but it only requires sender-side modifications. As a result, this approach can be easier and incrementally deployed in the Internet. While theoretically Quick-Start can outperform "IW10", the improvement in completion time for data transfer times can, in many cases, be small. After publication of [RFC6928], most modern TCP stacks have increased their default initial window.

4.4. ICMP Source Quench

The suggested references for ICMP Source Quench are:

- INTERNET CONTROL MESSAGE PROTOCOL [RFC0792]

The ICMP Source Quench message [RFC0792] allowed an on-path router to request the source of a flow to reduce its sending rate. This method allowed a router to provide an early indication of impending congestion on a path to the sources that contribute to that congestion.

4.4.1. Reasons for Non-deployment

This method was deployed in Internet routers over a period of time, the reaction of endpoints to receiving this signal has varied. For low speed links, with low multiplexing of flows the method could be used to regulate (momentarily reduce) the transmission rate. However, the simple signal does not scale with link speed, or the number of flows sharing a link.

The approach was overtaken by the evolution of congestion control methods in TCP [RFC2001], and later also by other IETF transports. Because these methods were based upon measurement of the end-to-end path and an algorithm in the endpoint, they were able to evolve and mature more rapidly than methods relying on interactions between operational routers and endpoint stacks.

After ICMP Source Quench was specified, the IETF began to recommend that transports provide end-to-end congestion control [RFC2001]. The Source Quench method has been obsoleted by the IETF [RFC6633], and both hosts and routers must now silently discard this message.
4.4.2. Lessons Learned

This method had several problems:

First, [RFC0792] did not sufficiently specify how the sender would react to the ICMP Source Quench signal from the path (e.g., [RFC1016]). There was ambiguity in how the sender should utilize this additional information. This could lead to unfairness in the way that receivers (or routers) responded to this message.

Second, while the message did provide additional information, the Explicit Congestion Notification (ECN) mechanism [RFC3168] provided a more robust and informative signal for network devices to provide early indication that a path has become congested.

The mechanism originated at a time when the Internet trust model was very different. Most endpoint implementations did not attempt to verify that the message originated from an on-path device before they utilized the message. This made it vulnerable to denial of service attacks. In theory, routers might have chosen to use the quoted packet contained in the ICMP payload to validate that the message originated from an on-path device, but this would have increased per-packet processing overhead for each router along the path, would have required transport functionality in the router to verify whether the quoted packet header corresponded to a packet the router had sent. In addition, section 5.2 of [RFC4443] noted ICMPv6-based attacks on hosts that would also have threatened routers processing ICMPv6 Source Quench payloads. As time passed, it became increasingly obvious that the lack of validation of the messages exposed receivers to a security vulnerability where the messages could be forged to create a tangible denial of service opportunity.

4.5. Triggers for Transport (TRIGTRAN)

The suggested references for TRIGTRAN are:

- TRIGTRAN BOF at IETF 55 [TRIGTRAN-55]
- TRIGTRAN BOF at IETF 56 [TRIGTRAN-56]

TCP [RFC0793] has a well-known weakness - the end-to-end flow control mechanism has only a single signal, the loss of a segment, and TCP implementations since the late 1980s have interpreted the loss of a segment as evidence that the path between two endpoints may have become congested enough to exhaust buffers on intermediate hops, so that the TCP sender should "back off" - reduce its sending rate until it knows that its segments are now being delivered without loss [RFC2581]. More modern TCP stacks have added a growing array of
strategies about how to establish the sending rate [RFC5681], but when a path is no longer operational, TCP would continue to retry transmissions, which would fail, again, and double their Retransmission Time Out (RTO) timers with each failed transmission, with the result that TCP would wait many seconds before retrying a segment, even if the path becomes operational while the sender is waiting for its next retry.

The thinking behind TRIGTRAN was that if a path completely stopped working because a link along the path was "down", somehow TCP could be signaled when that link returned to service, and the sending TCP could retry immediately, without waiting for a full retransmission timeout (RTO) period.

4.5.1. Reasons for Non-deployment

The early dreams for TRIGTRAN were dashed because of an assumption that TRIGTRAN triggers would be unauthenticated. This meant that any "safe" TRIGTRAN mechanism would have relied on a mechanism such as setting the IPv4 TTL or IPv6 Hop Count to 255 at a sender and testing that it was 254 upon receipt, so that a receiver could verify that a signal was generated by an adjacent sender known to be on the path being used, and not some unknown sender which might not even be on the path (e.g., "The Generalized TTL Security Mechanism (GTSM)" [RFC5082]). This situation is very similar to the case for ICMP Source Quench messages as described in Section 4.4, which were also unauthenticated, and could be sent by an off-path attacker, resulting in deprecation of ICMP Source Quench message processing [RFC6633].

TRIGTRAN’s scope shrunk from "the path is down" to "the first-hop link is down".

But things got worse.

Because TRIGTRAN triggers would only be provided when the first-hop link was "down", TRIGTRAN triggers couldn’t replace normal TCP retransmission behavior if the path failed because some link further along the network path was "down". So TRIGTRAN triggers added complexity to an already complex TCP state machine, and did not allow any existing complexity to be removed.

There was also an issue that the TRIGTRAN signal was not sent in response to a specific host that had been sending packets, and was instead a signal that stimulated a response by any sender on the link. This needs to scale when there are multiple flows trying to use the same resource, yet the sender of a trigger has no understanding how many of the potential traffic sources will respond by sending packets - if recipients of the signal back-off their
responses to a trigger to improve scaling, then that immediately mitigates the benefit of the signal.

Finally, intermediate forwarding devices required modification to provide TRIGTRAN triggers, but operators couldn’t charge for TRIGTRAN triggers, so there was no way to recover the cost of modifying, testing, and deploying updated intermediate devices.

Two TRIGTRAN BOFs were held, at IETF 55 [TRIGTRAN-55] and IETF 56 [TRIGTRAN-56], but this work was not chartered, and there was no interest in deploying TRIGTRAN unless it was chartered and standardized in the IETF.

4.5.2. Lessons Learned.

The reasons why this work was not chartered, much less deployed, provide several useful lessons for researchers.

- TRIGTRAN started with a plausible value proposition, but networking realities in the early 2000s forced reductions in scope that led directly to reductions in potential benefits, but no corresponding reductions in costs and complexity.

- These reductions in scope were the direct result of an inability for hosts to trust or authenticate TRIGTRAN signals they received from the network.

- Operators did not believe they could charge for TRIGTRAN signaling, because first-hop links didn’t fail frequently, and TRIGTRAN provided no reduction in operating expenses, so there was little incentive to purchase and deploy TRIGTRAN-capable network equipment.

It is also worth noting that the targeted environment for TRIGTRAN in the late 1990s contained links with a relatively small number of directly-connected hosts - for instance, cellular or satellite links. The transport community was well aware of the dangers of sender synchronization based on multiple senders receiving the same stimulus at the same time, but the working assumption for TRIGTRAN was that there wouldn’t be enough senders for this to be a meaningful problem. In the 2010s, it is common for a single "link" to support many senders and receivers on a single link, likely requiring TRIGTRAN senders to wait some random amount of time before sending after receiving a TRIGTRAN signal, which would have reduced the benefits of TRIGTRAN even more.
4.6. Shim6

The suggested references for Shim6 are:

- RFC5533 Shim6: Level 3 Multihoming Shim Protocol for IPv6
  [RFC5533]

The IPv6 routing architecture [RFC1887] assumed that most sites on the Internet would be identified by Provider Assigned IPv6 prefixes, so that Default-Free Zone routers only contained routes to other providers, resulting in a very small routing table.

For a single-homed site, this could work well. A multihomed site with only one upstream provider could also work well, although BGP multihoming from a single upstream provider was often a premium service (costing more than twice as much as two single-homed sites), and if the single upstream provider went out of service, all of the multihomed paths could fail simultaneously.

IPv4 sites often multihomed by obtaining Provider Independent prefixes, and advertising these prefixes through multiple upstream providers. With the assumption that any multihomed IPv4 site would also multihome in IPv6, it seemed likely that IPv6 routing would be subject to the same pressures to announce Provider Independent prefixes, resulting in a global IPv6 routing table that exhibited the same problems as the global IPv4 routing table. During the early 2000s, work began on a protocol that would provide the same benefits for multihomed IPv6 sites without requiring sites to advertise Provider Independent prefixes into the global routing table.

This protocol, called Shim6, allowed two endpoints to exchange multiple addresses ("Locators") that all mapped to the same endpoint ("Identity"). After an endpoint learned multiple Locators for the other endpoint, it could send to any of those Locators with the expectation that those packets would all be delivered to the endpoint with the same Identity. Shim6 was an example of an "Identity/Locator Split" protocol.

Shim6, as defined in [RFC5533] and related RFCs, provided a workable solution for IPv6 multihoming using Provider Assigned prefixes, including capability discovery and negotiation, and allowing end-to-end application communication to continue even in the face of path failure, because applications don’t see Locator failures, and continue to communicate with the same Identity using a different Locator.
4.6.1. Reasons for Non-deployment

Note that the problem being addressed was "site multihoming", but Shim6 was providing "host multihoming". That meant that the decision about what path would be used was under host control, not under router control.

Although more work could have been done to provide a better technical solution, the biggest impediments to Shim6 deployment were operational and business considerations. These impediments were discussed at multiple network operator group meetings, including [Shim6-35] at [NANOG-35].

The technology issues centered around concerns that Shim6 relied on the host to track all the connections, while also tracking Identity/ Locator mappings in the kernel, and tracking failures to recognize that a backup path has failed.

The operator issues centered around concerns that operators were performing traffic engineering, but would have no visibility or control over hosts when they chose to begin using another path, and relying on hosts to engineer traffic exposed their networks to oscillation based on feedback loops, as hosts move from path to path. At a minimum, traffic engineering policies must be pushed down to individual hosts. In addition, the usual concerns about firewalls that expected to find a transport-level protocol header in the IP payload, and won’t be able to perform firewalling functions because its processing logic would have to look past the Identity header.

The business issues centered removing or reducing the ability to sell BGP multihoming service, which is often more expensive than single-homed connectivity.

4.6.2. Lessons Learned

It is extremely important to take operational concerns into account when a path-aware protocol is making decisions about path selection that may conflict with existing operational practices and business considerations.

4.6.3. Addendum on MultiPath TCP

During discussions in the PANRG session at IETF 103 [PANRG-103-Min], Lars Eggert, past Transport Area Director, pointed out that during charter discussions for the Multipath TCP working group [MP-TCP], operators expressed concerns that customers could use Multipath TCP to loadshare TCP connections across operators simultaneously and compare passive performance measurements across network paths in real
time, changing the balance of power in those business relationships. Although the Multipath TCP working group was chartered, this concern could have acted as an obstacle to deployment.

Operator objections to Shim6 were focused on technical concerns, but this concern could have also been an obstacle to Shim6 deployment if the technical concerns had been overcome.

4.7. Next Steps in Signaling (NSIS)

The suggested references for NSIS are:

- the concluded working group charter [NSIS-CHARTER-2001]
- RFC 5971 GIST: General Internet Signalling Transport [RFC5971]
- RFC 5973 NAT/Firewall NSIS Signaling Layer Protocol (NSLP) [RFC5973]
- RFC 5974 NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling [RFC5974]
- RFC 5981 "Authorization for NSIS Signaling Layer Protocols [RFC5981]

The Next Steps in Signaling (NSIS) Working Group worked on signaling technologies for network layer resources (e.g., QoS resource reservations, Firewall and NAT traversal).

When RSVP [RFC2205] was used in deployments, a number of questions came up about its perceived limitations and potential missing features. The issues noted in the NSIS Working Group charter [NSIS-CHARTER-2001] include interworking between domains with different QoS architectures, mobility and roaming for IP interfaces, and complexity. Later, the lack of security in RSVP was also recognized ([RFC4094]).

The NSIS Working Group was chartered to tackle those issues and initially focused on QoS signaling as its primary use case. However, over time a new approach evolved that introduced a modular architecture using application-specific signaling protocols (the NSIS Signaling Layer Protocol (NSLP)) on top of a generic signaling transport protocol (the NSIS Transport Layer Protocol (NTLP)).

The NTLP is defined in [RFC5971]. Two NSLPs are defined: the NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling [RFC5974] as well as the NAT/Firewall NSIS Signaling Layer Protocol (NSLP) [RFC5973].
4.7.1. Reasons for Non-deployment

The obstacles for deployment can be grouped into implementation-related aspects and operational aspects.

- Implementation-related aspects:

  Although NSIS provides benefits with respect to flexibility, mobility, and security compared to other network signaling technologies, hardware vendors were reluctant to deploy this solution, because it would require additional implementation effort and would result in additional complexity for router implementations.

  The NTLP mainly operates as path-coupled signaling protocol, i.e., its messages are processed at the intermediate node’s control plane that are also forwarding the data flows. This requires a mechanism to intercept signaling packets while they are forwarded in the same manner (especially along the same path) as data packets. One reason for the non-deployment of NSIS is the usage of the IPv4 and IPv6 Router Alert Option (RAO) to allow for an efficient interception of those path-coupled signaling messages: This option requires router implementations to correctly understand and implement the handling of RAOs, e.g., to only process packet with RAOs of interest and to leave packets with irrelevant RAOs in the fast forwarding processing path (a comprehensive discussion of these issues can be found in [RFC6398]). The latter was an issue with some router implementations at the time of standardization.

  Another reason is that path-coupled signaling protocols that interact with routers and request manipulation of state at these routers (or any other network element in general) are under scrutiny: a packet (or sequence of packets) out of the mainly untrusted data path is requesting creation and manipulation of network state. This is seen as potentially dangerous (e.g., opens up a Denial of Service (DoS) threat to a router’s control plane) and difficult for an operator to control. End-to-end signaling approaches were considered problematic (see also section 3 of [RFC6398]). There are recommendations on how to secure NSIS nodes and deployments (e.g., [RFC5981]).

- Operational Aspects:

  End-to-end signaling technologies not only require trust between customers and their provider, but also among different providers. Especially, QoS signaling technologies would require some kind of dynamic service level agreement support that would imply (potentially quite complex) bilateral negotiations between different Internet service providers. This complexity was not considered to be justified and increasing the bandwidth (and thus avoiding
bottlenecks) was cheaper than actively managing network resource bottlenecks by using path-coupled QoS signaling technologies. Furthermore, an end-to-end path typically involves several provider domains and these providers need to closely cooperate in cases of failures.

4.7.2. Lessons Learned

One goal of NSIS was to decrease the complexity of the signaling protocol, but a path-coupled signaling protocol comes with the intrinsic complexity of IP-based networks, beyond the complexity of the signaling protocol itself. Sources of intrinsic complexity include:

- the presence of asymmetric routes between endpoints and routers
- the lack of security and trust at large in the Internet infrastructure
- the presence of different trust boundaries
- the effects of best-effort networks (e.g., robustness to packet loss)
- divergence from the fate sharing principle (e.g., state within the network).

Any path-coupled signaling protocol has to deal with these realities.

Operators view the use of IPv4 and IPv6 Router Alert Option (RAO) to signal routers along the path from end systems with suspicion, because these end systems are usually not authenticated and heavy use of RAOs can easily increase the CPU load on routers that are designed to process most packets using a hardware "fast path".

4.8. IPv6 Flow Label

The suggested references for IPv6 Flow Label are:

- IPv6 Flow Label Specification [RFC6437]

IPv6 specifies a 20-bit field Flow Label field [RFC6437], included in the fixed part of the IPv6 header and hence present in every IPv6 packet. An endpoint sets the value in this field to one of a set of pseudo-randomly assigned values. If a packet is not part of any flow, the flow label value is set to zero [RFC3697]. A number of Standards Track and Best Current Practice RFCs (e.g., [RFC8085], [RFC6437], [RFC6438]) encourage IPv6 endpoints to set a non-zero
value in this field. A multiplexing transport could choose to use multiple flow labels to allow the network to independently forward its subflows, or to use one common value for the traffic aggregate. The flow label is present in all fragments. IPsec was originally put forward as one important use-case for this mechanism and does encrypt the field [RFC6438].

Once set, the flow label can provide information that can help inform network devices about subflows present at the transport layer, without needing to interpret the setting of upper layer protocol fields [RFC6294]. This information can also be used to coordinate how aggregates of transport subflows are grouped when queued in the network and to select appropriate per-flow forwarding when choosing between alternate paths [RFC6438] (e.g. for Equal Cost Multipath Routing (ECMP) and Link Aggregation (LAG)).

4.8.1. Reasons for Non-deployment

Despite the field being present in every IPv6 packet, the mechanism did not receive as much use as originally envisioned. One reason is that to be useful it requires engagement by two different stakeholders:

○ Endpoint Implementation:

For network devices along a path to utilize the flow label there needs to be a non-zero value value inserted in the field [RFC6437] at the sending endpoint. There needs to be an incentive for an endpoint to set an appropriate non-zero value. The value should appropriately reflect the level of aggregation the traffic expects to be provided by the network. However, this requires the stack to know granularity at which flows should be identified (or conversely which flows should receive aggregated treatment), i.e., which packets carry the same flow label. Therefore, setting a non-zero value may result in additional choices that need to be made by an application developer.

Although the standard [RFC3697] forbids any encoding of meaning into the flow label value, the opportunity to use the flow label as a covert channel or to signal other meta-information may have raised concerns about setting a non-zero value [RFC6437].

Before methods are widely deployed to use this method, there could be no incentive for an endpoint to set the field.

○ Operational support in network devices:

A benefit can only be realized when a network device along the path also uses this information to inform its decisions. Network
equipment (routers and/or middleboxes) need to include appropriate support so they can utilize the field when making decisions about how to classify flows, or to inform forwarding choices. Use of any optional feature in a network device also requires corresponding updates to operational procedures, and therefore is normally only introduced when the cost can be justified.

A benefit from utilizing the flow label is expected to be increased quality of experience for applications - but this comes at some operational cost to an operator, and requires endpoints to set the field.

4.8.2. Lessons Learned

The flow label is a general purpose header field for use by the path. Multiple uses have been proposed. One candidate use was to reduce the complexity of forwarding decisions. However, modern routers can use a "fast path", often taking advantage of hardware to accelerate processing. The method can assist in more complex forwarding, such as ECMP and load balancing.

Although [RFC6437] recommended that endpoints should by default choose uniformly-distributed labels for their traffic, the specification permitted an endpoint to choose to set a zero value. This ability of endpoints to choose to set a flow label of zero has had consequences on deployability:

- Before wide-scale support by endpoints, it would be impossible to rely on a non-zero flow label being set. Network devices therefore would need to also employ other techniques to realize equivalent functions. An example of a method is one assuming semantics of the source port field to provide entropy input to a network-layer hash. This use of a 5-tuple to classify a packet represents a layering violation [RFC6294]. When other methods have been deployed, they increase the cost of deploying standards-based methods, even though they may offer less control to endpoints and result in potential interaction with other uses/interpretation of the field.

- Even though the flow label is specified as an end-to-end field, some network paths have been observed to not transparently forward the flow label. This could result from non-conformant equipment, or could indicate that some operational networks have chosen to re-use the protocol field for other (e.g. internal purposes). This results in lack of transparency, and a deployment hurdle to endpoints expecting that they can set a flow label that is utilized by the network. The more recent practice of "greasing" [GREASE] would suggest that a different outcome could have been
achieved if endpoints were always required to set a non-zero value.

- [RFC1809] noted that setting the choice of the flow label value can depend on the expectations of the traffic generated by an application, which suggests an API should be presented to control the setting or policy that is used. However, many currently available APIs do not have this support.

A growth in the use of encrypted transports, (e.g. QUIC [QUIC-WG]) seems likely to raise similar issues to those discussed above and could motivate renewed interest in utilizing the flow label.

5. Security Considerations

This document describes Path Aware technologies that were not adopted and widely deployed on the Internet, so it doesn’t affect the security of the Internet.

If this document meets its goals, we may develop new technologies for Path Aware Networking that would affect the security of the Internet, but security considerations for those technologies will be described in the corresponding RFCs that specify them.

6. IANA Considerations

This document makes no requests of IANA.

7. Acknowledgments

Initial material for Section 4.1 on ST2 was provided by Gorry Fairhurst.

Initial material for Section 4.2 on IntServ was provided by Ron Bonica.

Initial material for Section 4.3 on Quick-Start TCP was provided by Michael Scharf.

Initial material for Section 4.4 on ICMP Source Quench was provided by Gorry Fairhurst.

Initial material for Section 4.5 on Triggers for Transport (TRIGTRAN) was provided by Spencer Dawkins.

Section 4.6 on Shim6 builds on initial material describing obstacles provided by Erik Nordmark, with background added by Spencer Dawkins.
Initial material for Section 4.7 on Next Steps In Signaling (NSIS) was provided by Roland Bless and Martin Stiemerling.

Initial material for Section 4.8 on IPv6 Flow Labels was provided by Gorry Fairhurst.

Our thanks to C.M. Heard, Gorry Fairhurst, Joe Touch, Joeri de Ruiter, Roland Bless, Ruediger Geib, and Wes Eddy, who provided review comments on previous versions.

Special thanks to Adrian Farrel for helping Spencer navigate the twisty little passages of Flow Specs and Filter Specs in IntServ, RSVP, MPLS, and BGP. They are all alike, except for the differences [Colossal-Cave].

8. Informative References


[TRIGTRAN-56]
"Triggers for Transport BOF at IETF 56", November 2003,

Author’s Address

Spencer Dawkins (editor)
Wonder Hamster Internetworking

Email: spencerdawkins.ietf@gmail.com
Multipath Use Case and Requirement for Security

draft-rass-panrg-mpath-usecase-01

Abstract

This document describes a use case of multipath to achieve full CIA+ by using symmetric cryptography and point-to-point shared secrets.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 13, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

Public-key cryptography is a convenient tool for end-to-end security, but in practice can be cumbersome or complicated for non-expert users to apply. Certificate- and key management rely on complex infrastructures and to a significant extent impose monetary cost and human effort.

This document describes a method of using symmetric cryptography and point-to-point shared secrets to establish full CIA+ (confidentiality, integrity, availability and authenticity) end-to-end security. The respective schemes rely on multipath transmission and threshold cryptography, and are intended to work transparently for the users, i.e., entirely below the application layer. The only involvement of human action is for the key establishment, which is in
our setting equivalent to a pairing of devices, as is familiar from other contexts, such as Bluetooth.

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 [RFC2119].

2. Assumptions

We assume a network of bidirectional links, represented as an undirected graph G=(V,E). An edge e=(v_1, v_2) in the set E represents a point-to-point connection between the nodes v_1 and v_2 in the network. We assume that every such pair (v_i, v_j) in E shares an individual secret k_ij, which is individually distinct for all edges (i.e., no two pairs have, other than by coincidence, the same secret). The secret exchange or establishment is left to arbitrary means, e.g., any device pairing scheme [I-D.ietf-dnssd-pairing] or cryptographic methods like Diffie-Hellman key exchange [RFC2631] would be admissible, up to quantum key distribution [BB84]. Indeed, end-to-end security in quantum networks is the most natural application area of multipath transmission as we discuss here.

We further assume that keys between adjacent nodes in the network have been exchanged in an authentic manner; say, by sufficient proximity during the device pairing (e.g., near field communication).

3. Multipath Routing

Multipath routing offers the remarkable ability of establishing public-key like security without computational intractability. This means that periodic updates of keys or server certificates are no longer required in such systems; updates to keys for symmetric crypto are much easier by device re-pairing or refreshing keys from existing key material, such as is done in quantum key distribution (QKD). Multipath transmission, requiring no quantum technology per se, offers nonetheless the same level of security QKD [BB84] and can resist attacks by quantum computers (like post-quantum cryptography [BD08]).

The key element to this end is using multiple paths to send a message, which in the simplest instance is just like humble symmetric encryption: consider two nodes A and B that have no direct connection between them (i.e., A and B are several hops apart). Let us assume that two paths connect A to B, where those paths intersect only at A and B (we call such paths node-disjoint). If so, then A can choose a
session key $k$ that it sends to $B$ over the first path, and deliver the
encrypted payload over the second path. If the encryption is chosen
properly (e.g., Vernam cipher [Ver55]) and the adversary does not
intercept both paths, the connection remains secure.

The scheme straightforwardly generalizes to more than two paths,
where the payload is (always) split into shares and transmitted over
separate paths in parallel or sequentially. Security comes from the
proper encoding/creation of the pieces so that an attacker needs to
intercept a certain number of paths in order to breach
confidentiality, insert a forged message, or cause a denial of
service. The fundamental circumstance implying security here is the
existence of $k \geq 2$ node and link disjoint paths, so that an
adversary needs to conquer at least $k$ nodes in the network to breach
security (by mounting a person-in-the-middle attack).

Security (up to QKD without trusted relays) thus hinges on the
following network-related assumptions:

3.1. Multi-path Service and User-Network Interface

There are at least two disjoint paths between node $A$ and node $B$, so $A$
can send packets to node $B$ via different paths efficiently and
reliably.

New User-Network Interface (UNI) should be defined to exchange
information between end device/application and network. The
information may include but not limited to:

- User expectation: such as number of paths, bandwidth required etc.
- Path aware info: the network should dynamically provide end-device
  information such as number of paths available, each path’s
  attributes: path reliability, routing quality, bandwidth, path
  elements etc.

3.2. Path and Routing Reliability

The sender $A$ can deliberately choose any among the existing paths to
its receiver $B$ to transmit a message. The routing is reliable in the
sense that there is at least a probabilistic guarantee for the packet
to travel over exactly the chosen route with a likelihood $p$ that $A$
can quantify (not necessarily control). In other words, the chances
for the path to be blocked, or for the packet to take a detour for
any reason (e.g., load balancing, temporary congestions, or similar)
is at most $1-p$, with the value of $p$ being known to $A$. The ideal case
$p = 1$ expresses that the chosen route has a perfect reliability
(i.e., no deviations and guaranteed delivery).
It is admissible to express the path reliability in terms of several such probabilities, referring to different dimensions for the quality of service. That is, we may define a probability \( p_1 \) for the packet to stay on the chosen route, another probability \( p_2 \) for the packet to be delivered at all (i.e., not being blocked), or similar.

There are per se no stringent constraints regarding latencies or for several packets to arrive in the order of transmission, since the outer (cryptographic) transmission protocols can handle this. However, the aforementioned probabilities quantifying the quality of routing need to be accurately known to the sender \( A \). Suitable protocols to handle path deviations (temporary detours) and to optimize quality-of-service tradeoffs based on such knowledge are found in the literature [Ras13], [RK12].

3.3. Cross Domain Path Reliability

If two distinct network domains are joined together, the topologies of both networks are reliably made known to the nodes in the respective other network. Chosen routes from one network into the other must remain quantifiably reliable in the sense of section 3.2 above, i.e., a node \( A \) in one network must still be able to determine a probability \( p \) for a packet to stay on its route and to arrive at the designated destination across all network domains that it traverses.

3.4. Cross Domain Network Connections

Whenever a node \( A \) has an outside connection to a node in another network domain \( N_2 \), \( A \) should not have a second connection to another node in the same network domain \( N_2 \). That is, if two network domains \( N_1 \) and \( N_2 \) are joined together via \( k \) links, those links should pairwise connect \( k \) distinct nodes in \( N_1 \) to another \( k \) distinct nodes in \( N_2 \). This assures that the so-constructed larger network retains the necessary number of (at least) \( k \) node disjoint paths across the domains (by avoiding bottle-neck connections between networks \( N_1 \) and \( N_2 \)).

3.5. Updates upon Changing Network Topologies

The information described under the preceding sections needs to remain up-to-date whenever \( A \) wishes to send a packet somewhere. Changing topologies such as in ad hoc networks call for a proper and reliable updating scheme to \( A \)'s local information about the network topology. This includes also changes in topologies of remote network domains (that the sender does not itself belong to).
3.6. Enforced Device Pairing and De-Pairing

Whenever a node $X$ joins a network, it must establish shared secrets (for cryptography) with any neighbor with whom it has a direct point-to-point connection. Whenever a node $X$ leaves the network, nodes losing the connection to $X$ need to abandon their cryptographic key formerly assigned to the connection with $X$. The key exchange protocols can be arbitrary (cf. section 2), but the device pairing must in any case be authenticated.

4. Summary

The ability to route messages along chosen paths in a network, together with sufficient vertex connectivity and unique neighborhoods for each node opens up the possibility to achieve end-to-end security:

- without public-key cryptography.
- using only light-weight symmetric cryptographic primitives (encryption and hashing).
- and with the most trivial key-management consisting of only the exchange of keys between directly connected devices (along device pairing).

5. Security Considerations

TBD.

6. Acknowledgements

TBD.

7. References

7.1. Normative References

[I-D.ietf-dnssd-pairing]

7.2. Informative References


[Ras18] Stefan Rass, CoRR abs/1810.05602 (2018)., "Perfectly secure communication, based on graph-topological addressing in unique-neighborhood networks".


[Sha79] Adi Shamir, ACM 22 (1979), no. 11, 612--613., "How to share a secret".
Appendix A. Cryptographic and Graph-Theoretic Basics

A.1. Secret Sharing

We assume a message $m$ to come as a binary string of length $L$. A simple $k$-out-of-$k$ secret sharing is by picking a set of $k-1$ random strings $s_1, s_2, \ldots, s_{(k-1)}$ of the same length as $m$ and computes $s_k := m \text{ XOR } s_1 \text{ XOR } s_2 \text{ XOR } \ldots \text{ XOR } s_{(k-1)}$. Information-theoretically, one can prove [Sha49] that the recovery of $m$ is impossible from any set of less than $k$ of the strings $s_1, \ldots, s_k$ (since the missing string effectively acts as a one-time pad concealing $m$).

The sharing as just described is replaceable by more sophisticated schemes, such as Shamir’s polynomial sharing [Sha79], which adds error correction capabilities [MS81] via using an isomorphy to Reed-Solomon encoding. We shall, however, hereafter not further relate to standardized versions of Reed-Solomon forward error correcting codes [LRPP09], but rather work with the above simple scheme instead.

Abstractly, we shall introduce a sharing function $SPLIT(m, k)$ that decomposes an input message $m$ into a set of $k$ shares according to any scheme of choice (for the description in this document, the above XOR-based scheme will suffice). The inverse of $SPLIT$ will be the function $COMBINE(s_1, \ldots, s_k)$, taking $k$ (out of a potentially larger set) of shares to reconstruct the message $m$ from it. Note that $COMBINE$ internally may invoke error correction algorithms [LRPP09], which we do not further expand here.

A.2. Network Connectivity

If a node $A$ wants to transmit a message $m$ to a node $B$, we assume that $A$ can choose a path, or a set of paths through the network along a physical connection (over multiple hops) to the end-node $B$. Further, we assume that the network’s node connectivity is such that more than one route from $A$ to $B$ exists, and that at least two routes exist that do not intersect other than at $A$ and $B$ (node-disjoint paths). It is known that the existence of $k$ node-disjoint paths is equivalent to the graph admitting a $k$-vertex cut; equivalently, we call such a graph $k$-vertex-connected. The smallest graph with that property is the complete graph with $k+1$ nodes. Furthermore, if two $k$-vertex-connected graphs are given, we can combine them into one (big) $k$-vertex connected graph $G$ as follows: we pick $k$ distinct nodes $u_1, \ldots, u_k$ in $G_1$ and another $k$ distinct nodes $v_1, \ldots, v_k$ in $G_2$, and connect the two graphs by adding edges $(u_i, v_i)$ for all $i=1,2,\ldots,k$. The resulting graph contains all nodes and edges from $G_1$ and $G_2$, plus the connecting edges between the two graphs. It is provably a $k$-vertex-connected graph, admitting at least $k$ node-
disjoint paths between any two nodes in either graph and from any
node in G_1 to any node in G_2 and vice versa.

While it may not be too optimistic to hope for a large k in the
existing internet topology, matters of resilience against failure of
single nodes in the network call for a least k=2, so that the network
remains connected if one node (and hence the adjacent edge) fails.

Let A’s available routes to B be enumerated as R_1, ..., R_k, which A
picks with likelihoods p_1, ..., p_k, say, p_i := 1/i for an
equiprobable choice of a single route. Moreover, we let each
transmission use their point-to-point shared secrets to encrypt a
message along the network edge (v_i, v_j) under the key k_ij (e.g.,
by means of the Advanced Encryption Standard or others).

Appendix B. Multipath Transmission and Game-Theoretic Security

B.1. End-to-end Confidentiality - Parallel Version

To confidentially transmit the message m, A proceeds as follows:

1. Decompose m into shares \{s_1, ..., s_k\} := SPLIT(m)
2. Send each share s_i over the route R_i (for i = 1, ..., k) in
parallel to B.
3. B, upon receiving all shares recovers the message as m :=
COMBINE(r_1, ..., r_k).

By construction, the attacker needs to gather all k shares to recover
m, so that if the attacker can intercept only less than k paths, the
message m remains perfectly concealed (by the aforementioned
arguments). A picture of the scheme is found at
https://www.syssec.at/user/themes/syssec-theme/images/publikationen/
MPTrans.png

B.2. End-to-end Confidentiality - Sequential-Parallel Version

The above scheme can be further strengthened by a two-stage sharing
as follows: as before, let m the message that A wishes to send to B
in perfect privacy. It proceeds as follows:

1. Decompose m into n shares \{s_1, ..., s_n\} := SPLIT(m)
2. For i = 1, 2, ..., n: send each share s_i by the parallel scheme
described above, resulting in the transmission of shares r_i1, ...
..., r_i k for the share s_i
3. The receiver B then needs to (i) reconstruct every share \( s_i := \text{COMBINE}(r_{i1}, \ldots, r_{ik}) \) (as in the parallel version above), and (ii) reconstruct the overall message as \( m := \text{COMBINE}(s_1, \ldots, s_n) \).

An attacker needs to intercept the entirety of shares for each individual transmission, as well as for all the sequential transmissions. Unless the attacker can mount a full person-in-the-middle attack, the message \( m \) remains perfectly concealed. Even if the attacker has a positive probability \( q < 0 \) to catch all shares for a single transmission in step 2, the probability to catch the entirety of \( n \) sequential shares (created in step 1) equals \((1-q)^n \) (the path choices are made stochastically independent). In choosing \( n \) large enough, A can make the adversary’s success chances exponentially small.

B.3. Randomized Routing to Maximize Security against Node (Failures)

Suppose that the attacker can intercept a fixed maximum number \( t < k \) of nodes, where \( k \) is the network’s vertex connectivity. If the network is such that certain routes are more or less reliable than others (e.g., some routes may be easier to intercept for the adversary or temporarily be unavailable), there is no obligation in the above scheme to use the full set of paths per parallel transmission. Instead, to transmit a share (whether in the parallel or sequential-parallel version of the transmission), the sender may randomly pick the route \( R_i \) with likelihood \( p_i \), and transmit the share over the chosen route.

Knowing the choice rules \( p_1, \ldots, p_k \) for the \( k \) routes that A can choose from, the attacker may seek to compute an optimal strategy for intercepting, resulting in probabilities \( q_1, \ldots, q_{|V|} \) for nodes to attack (excluding the nodes for A and B here, since our security goal is confidentiality, disregarding impersonation attacks for the moment).

The optimal computation of probabilities to choose routes, and individual likelihoods to intercept nodes amounts to a simple two-person matrix game [Ras13], whose saddle-point value (computable by means of linear optimization) systematically quantifies (bounds) the likelihood for the attacker to succeed. For a simplified example, assuming that all nodes are equally "easy" for the attacker to conquer, yet with a bound to no more than 1 node to be under the adversary’s control at a time, the optimal choice for the sender A would be an equiprobable pick among the routes, i.e., \( p_i := 1/k \) for all \( i \), and an equiprobable choice of victim nodes for the attacker (here, we assumed that the sender uses only a single path at a time).
B.4. Availability

The XOR sharing used in Section 2.1 is vulnerable against packet loss (whether this happens by coincidence or due to the attacker’s actions; DoS attacks). Making the scheme resilient against packet loss or damage calls for error correction capabilities within the COMBINE function, e.g., using the methods described in [LRPP09]. A full-fledged scheme using Reed-Solomon error correction towards optimized availability and confidentiality is described by [FFGV07].

B.5. End-to-End Authenticity

Using a similar idea [RS10], authenticity of messages is accomplishable by message authentication codes. Since the sender A shares secrets only with her/his direct neighbors, it can only use their secrets to attach a message authentication code. The receiver B, being several hops away from A, does not know the secrets to verify the MAC, but, thanks to its ability of chosen path routing, can ask A’s neighbors to verify the MACs on B’s behalf.

Putting this to practice, A authenticates a message m for B as follows, using the keys \( \{k_1, \ldots, k_n\} \) that A shares with its direct neighbors in the network. We write MAC(m,k) to denote a message authentication code (MAC) for a message m computed under the (secret) key k. Moreover, let H be a cryptographically strong hash function (e.g., SHA-3 or likewise).

1. A computes hash-MACs, e.g., using the HMAC scheme in [RFC2104], and attaches the MACs \( \{a_i := MAC(H(m), k_i) \mid i=1,2,\ldots,n\} \) to the message.

2. B receives the message m' (say, over a multipath transmission scheme with chosen routes as described above). To verify that m' is authentic, B computes the hash h' = H(m') and asks A's neighbors to verify the respective MACs. To this end, B contacts the i-th neighbor of A on a chosen route, and sends the data \( \{h', a_i'\} \) to A's neighbor with whom A shares the secret k_i. Here, the value a_i' is the MAC that B received (which could equally well have been corrupted).

3. A’s neighbor no. i uses its secret k_i to verify if MAC(h',k_i) = a_i'. It replies the result ("yes" or "no") back over the same route as how the query came in. This process happens concurrently at all of A’s neighbors (for i = 1, \ldots, n).

4. B collects all replies and takes either a majority decision or (in the most stringent setting) rejects if any of the replies comes back negative.
The condition upon which B accepts A’s message as authentic may depend on how resilient one needs to be about an adversary potentially manipulating the verification query to B. If B rejects upon a single negative verification, then even an attacker that can conquer only a single node on any of the chosen paths can mount a denial-of-service. On the contrary, if B accepts the majority vote, then the attacker needs to intercept (and manipulate) more than half of the routes chosen.

The security of this scheme follows by similar arguments as in the case for confidentiality: the scheme is secure as long as the adversary cannot mount a full person-in-the-middle attack, conditional on the attacker’s inability to find hash-collisions (in that sense, the scheme is, unlike the multipath transmission for confidentiality), only computationally secure.

Note that confidentiality of the message against the verifying neighbors is not directly addressed here beyond the point of sending a hash of m for verification instead of the full message. Heuristically, the message thus remains concealed to the extent of the neighbor’s inability to find a meaningful pre-image for the received value h’. We assumed the neighbors to be honest, unless being under the attacker’s control, so that a denial-of-service or intentionally incorrect response is in any case possible, and cannot be ruled out by this protocol.

B.5.1. Non-Repudiation

Under proper graph topological properties, the above authentication scheme, though based on symmetric cryptography only, shares the non-repudiation feature of public-key digital signatures. In fact, if the set of secrets shared between a node and its direct neighbors (or a subset thereof) is unique, i.e., distinct, for each node, then no other node than A can create the MAC-set attached to the message m. Networks with that property are easy to recognize based on their adjacency matrix [Ras18]; moreover, the "unique-neighborhood property" is preserved upon the same network merging operations as described above for k-vertex-connectivity.

B.6. Integrity

From the construction of Section 3.4, integrity is directly implied by the use of hashes that additionally act as checksums. That is, any distortion on the transmission line will with overwhelming
probability invalidate the MAC or inner hash, thus causing the protocol to indicate this error. Conversely, if B accepts A’s message as authentic, integrity verification is accomplished in the same blow, unless the attacker managed to forge the message as a whole (in which case, integrity is also unhedgeable).

Authors’ Addresses

Stafan Rass
Universitaet Klagenfurt
EMail: stefan.rass@aau.at

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA
EMail: yingzhen.qu@futurewei.com

Lin Han
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA
EMail: lin.han@futurewei.com