Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 January 2024

H. Asai
Preferred Networks / WIDE Project

10 July 2023

Separation of Data Path and Data Flow Sublayers in the Transport Layer
draft-asai-tsvwg-transport-review-04

Abstract

This document reviews the architectural design of the transport layer. In particular, this document proposes to separate the transport layer into two sublayers; the data path and the data flow layers. The data path layer provides functionality on the data path, such as connection handling, path quality and trajectory monitoring, waypoint management, and congestion control for the data path resource management. The data flow layer provides additional functionality upon the data path layer, such as flow control for the receive buffer management, retransmission for reliable data delivery, and transport layer security. The data path layer multiplexes multiple data flow layer protocols and provides data path information to the data flow layer to control data transmissions, such as prioritization and inverse multiplexing for multipath protocols.

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 11 January 2024.

Table of Contents

1. Introduction 3
2. Transport Layer Functionality Review 4
2.1. Conventional Communication Layer Model 4
2.2. Data Path-aware Networking 5
2.3. Resource Management: Flow Control and Congestion Control 6
2.4. Communication Models and Transport Layer Protocols . . . 7
2.5. Multipath Protocols 9
2.6. Reliable Data Communication 9
2.7. Security 10
2.8. Summary 10
3. Specifications of Data Path Layer and Data Flow Layer 11
3.1. Data Path Layer 11
3.1.1. In-band trajectory monitoring 11
3.1.2. Waypoint management 12
3.1.3. Bidirectional connection establishment 13
3.1.4. One-to-many connection 13
3.1.5. Data path quality (congestion) monitoring and congestion control 13
3.1.6. Data flow multiplexing 14
3.1.7. Packet duplication 14
3.1.8. Data path security 14
3.2. Data Flow Layer 14
3.2.1. Retransmission for reliable data communication . . . 15
3.2.2. Flow control for receive-buffer management 15
3.2.3. Flow prioritization 15
3.2.4. Data flow and end-to-end security 15
3.2.5. Inverse multiplexing for multipath protocols 15
4. Use Cases 16
4.1. Multipath Transport Layer Protocols 16
4.2. Congestion Control Acceleration 17
4.3. Multi-access Edge Computing 18
4.4. In-Network Computing 18
4.5. Flow Arbitration 19
5. Implementation Considerations 20
5.1. Data Path Layer Protocol over IP 20
5.2. Data Path Layer Protocol over UDP with Port Number . . . 20

6. IANA Considerations 21
7. Security Considerations 21
8. Acknowledgements 22
9. Normative References 22
10. Informative References 24
Author's Address 26

1. Introduction

Various transport layer protocols have been developed to satisfy diversified requirements of applications over heterogeneous networks. For example, Transmission Control Protocol (TCP) [RFC0793], one of the most commonly used transport layer protocols on the Internet, implements flow control and retransmission mechanisms for reliable communication. TCP also implements a congestion control mechanism to efficiently and effectively utilize shared network bandwidth resources. Multipath protocols, such as multipath TCP [RFC8684], have been developed to utilize multiple data paths between end-hosts for better throughput. Below the transport layer, Explicit Congestion Notification (ECN) [RFC3168] has been added to the Internet protocol to assist congestion control of transport layer protocols. Basically, rich functionality between end-hosts are implemented on the transport layer according to the end-to-end principle. As a result, transport layer protocols have become complex as they need to implement many functionality, and thus, it sacrifices the middlebox-friendliness and the extensibility of transport layer protocols.

This document aims to clarify the transport layer's functionality and specifies two sublayers of the transport layer, the data path and the data flow layers, for better extensibility of transport layer protocols. Hence, this document does not intend to obsolete the transport layer or violate the current Internet architecture. In this document, the data path functionality of the transport layer, such as bidirectional connection handling, waypoint management, and congestion control, is separated from the data flow functionality, such as flow control for the receive buffer management, retransmission for reliable data delivery, and transport layer security.

This document first reviews the transport layer's functionality from the viewpoint of data paths and data flows in Section 2. It then specifies the data path layer and the data flow layer in Section 3. The data path and data flow layers provide the data path and the data flow functionality, respectively. The separation of the transport layer into these two sublayers enalbes us to invent data path and data flow layer protocols for advanced Internet technologies, such as multi-access edge computing (MEC) and in-network computing, as discussed in Section 4.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Transport Layer Functionality Review

This section reviews the conventional layering and transport layer functionality. It then summarizes the transport layer functionality by distinguishing data paths and data flows.

2.1. Conventional Communication Layer Model

RFC 1122 [RFC1122] defines three communication layers, link layer, Internet layer, transport layer, and the interfaces between these layers. Link-layer protocols provide hop-by-hop data communications. Internet layer protocols such as the Internet Protocol (IP), the Internet Control Message Protocol (ICMP), and the Internet Group Management Protocol (IGMP) provide fragmentation, hop-by-hop datagram forwarding, and end-to-end datagram delivery. RFC 1123 [RFC1123] defines the interface between the application layer and the transport layer.

The Internet design follows the end-to-end principle to keep the simplicity of the Internet layer. Thus, the main functionality of the Internet layer is to provide end-to-end reachability. It does not guarantee datagram integrity, which means that packet loss, duplication, corruption, or reordering may occur. Over the Internet layer, the transport layer implements such functionality to provide datagram integrity on the end-hosts to achieve end-to-end communications.

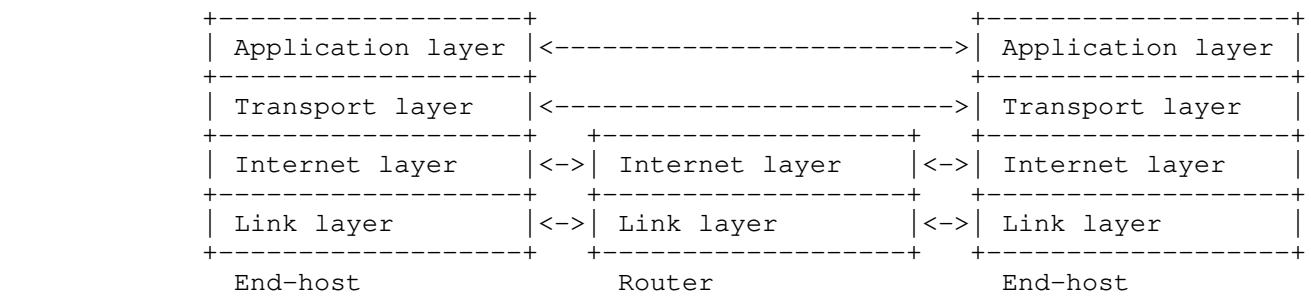


Figure 1: Conventional layering of Internet architecture

The transport layer provides various functions over the IP. User Datagram Protocol (UDP) [RFC0768] and TCP are the most commonly used transport layer protocols. UDP is a connectionless transport layer protocol with a minimum header. TCP implements flow control according to the receiver's buffer capacity, retransmission for reliable communication, congestion control by a packet delivery status such as packet loss and delay. The transport layer may implement an end-to-end security function, Transport Layer Security (TLS) [RFC8446].

Figure 1 illustrates the conventional layering of Internet architecture. A router forwards an IP datagram to a next-hop router corresponding to the destination address. In this way, the Internet layer protocol, IP, provides end-to-end reachability through hop-by-hop routing. The transport layer provides additional functions for end-to-end communications.

2.2. Data Path-aware Networking

As described above, the Internet layer's main functionality has been end-to-end reachability with hop-by-hop packet forwarding based on destination IP address. Therefore, it is unaware of data paths, i.e., trajectories of packets. Best-effort communications over ''dumb'' networks without the quality of service (QoS) have been the Internet principle [RFC2768]. This end-to-end principle of the Internet has achieved a scalable architecture. However, computer networks' advancement introduced QoS and middleboxes (e.g., transparent proxies, Network Address Translation (NAT) [RFC3022], firewalls) to achieve high-quality data communication and optimize data communication over distributed and heterogeneous computer networks. These technologies require to be aware of data paths associated with data flows.

The Internet layer has been extended to support QoS. Differentiated Services, DiffServ [RFC2474], is one of the technologies to implement QoS. It enables autonomous and scalable service discrimination using an IP header field, differentiated services codepoint (DSCP), to implement QoS for a data path. Network operators may configure policy-based routing based on DSCP values to be aware of data paths. As another technology to treat QoS, Segment Routing [RFC8402] enables data path configuration for individual flows by leveraging the source routing paradigm. Thus, it is essential to be aware of data paths to properly apply QoS to data communication traffic.

Middleboxes are more complex than QoS. They add various functions such as firewalls, TCP offloading, transcoding, and content caches to a data path. These functions require to be aware of waypoints to be activated. Moreover, they are possibly stateful, and thus, require to consistently use the same waypoint for a single data flow. Policy-based routing technologies with packet classification using the IP and the transport layer protocol headers have collectively been leveraged to route traffic through the middleboxes as the IP layer is not aware of data paths. If a middlebox is transparent to end-hosts, it should be installed at the network gateway or redirected by policy-based routing to activate the function. Otherwise, an end-host should specify the middlebox as a target end-host. In addition to these middleboxes, distributed computing technologies, such as MEC and in-network computing, also require to be aware of data paths. Note that overlay networks or application layer service discovery mechanisms may be used to achieve data path-ware networking for these distributed computing technologies.

In summary, advanced computer networks require to treat data paths for QoS, middleboxes, and distributed computing. In policy-based routing, packet classification for data path treatment may use the header information of transport layer protocols such as port numbers as well as IP header fields.

2.3. Resource Management: Flow Control and Congestion Control

As the Internet layer does not provide resource management functionality, transport layer protocols may implement it, such as flow control and congestion control. Both flow control and congestion control mechanisms control packet transmission for resource management. However, the target resources are different. They control packet transmission according to the receiver’s buffer capacity and the network bandwidth capacity, respectively.

For example, in TCP’s flow control, a receiver announces the remaining receive buffer size as the window size. Hence, flow control is not aware of network bandwidth capacity. On the other

hand, congestion control is performed based on data communication quality information, such as packet loss and delay. The underlying Internet layer may provide congestion information by ECN. In this manner, transport layer protocols control congestion depending on the data path's network resources. Therefore, congestion control is associated with a data path, while flow control is associated with end-hosts.

As discussed above, congestion control should be performed on the associated data path. However, in current transport layer protocols such as TCP, Datagram Congestion Control Protocol (DCCP) [RFC4340], and Stream Control Transmission Protocol (SCTP) [RFC4960], an individual flow independently performs congestion control even if the same data path multiplexes multiple flows. Therefore, multiple flows cannot collectively perform congestion control for a data path. For example, when a data path multiplexes a TCP and a UDP flows, the TCP flow's congestion control may affect the other UDP flow's quality. Moreover, transport layer protocols utilize network resources on a fair basis. Thus, flow prioritization cannot be implemented at the transport layer, although a QoS technology such as DiffServ may be leveraged.

In summary, congestion control should be performed on a data path instead of data flows as it controls network resources, while flow control should be performed per flow as it is associated with the resources of end-hosts, from the viewpoint of resource management in the transport layer.

2.4. Communication Models and Transport Layer Protocols

This subsection summarizes four communication models between end-hosts at the Internet layer; unicast, anycast [RFC7094], broadcast [RFC0919], and multicast [RFC1112], and then discusses transport layer protocols for these communication models.

Unicast is the most fundamental communication model that performs a data communication between two end-hosts. Most transport layer protocols such as TCP are designed for unicast. Unicast may be unidirectional, but some transport layer protocols only support a bidirectional data communication.

Anycast is a special communication model extended from unicast. Therefore, the same transport layer protocols as unicast are often used. Anycast performs a one-to-one data communication, but a destination end-host is not uniquely identified by its IP address because multiple end-hosts share an identical IP address. Instead, a destination end-host is determined by IP routing in anycast. Thus, anycast needs to take into account data path changes for stateful connections as the destination end-host may change by IP routing during a long-lived transaction [RFC4786].

Broadcast and multicast are a communication model that one end-host sends a packet to multiple end-hosts. A difference between broadcast and multicast is that multicast is to deliver a packet to multiple end-hosts that join a specific multicast group while broadcast delivers packets to all end-hosts within a specified broadcast domain (limited by the hop count). In other words, multicast is referred to as a group data communication. In multicast, IGMP and Protocol Independent Multicast (PIM) (e.g., PIM-DM [RFC3973] and PIM-SM [RFC7761]) are used to build data paths (i.e., multicast tree) from a source end-host to destination end-hosts for a group. Therefore, data paths are controlled by the Internet layer. Reliable transport layer protocols may be used in the control plane [RFC6559]. Over the data paths, stateless and non-reliable transport layer protocols such as UDP are usually used. For TCP-friendly multicast, [RFC4654] defines a congestion control protocol for multicast. To achieve reliable multicast, Multicast Transport Protocol (MTP) [RFC1301] defines a transport layer protocol with flow control, QoS and error recovery for multicast. However, it has been pointed out that MTP has a potential scalability issue for flow handling [RFC1458].

In these four communication models, IP routing provides end-to-end reachability and determines data paths. Transport layer protocols are not basically aware of data paths, although operational considerations exist in anycast for uncertainty of data paths [RFC4786]. As pointed out in [RFC1458], data flow management requires more resources to manage the states compared to data path functionality such as congestion control. It implies that the resource management for data flows is more significant in one-to-many communication of multicast cases than unicast cases discussed in Section 2.3.

From the viewpoint of the communication models, anycast and multicast rely on data paths controlled by IP routing. Particularly in multicast, routers manage multicast tree information (i.e., data path states), and may assist congestion control of transport protocols [RFC3048]. Furthermore, routers may implement a retransmission mechanism, such as PGM Reliable Transport Protocol [RFC3208], upon a data path. Such a mechanism requires routers to manage flow states

of the transport layer. It is similar to the mechanism of TCP's congestion control acceleration middleboxes in unicast or anycast. Although these mechanisms are implemented in routers or middleboxes, they are obviously the functionality of the transport layer.

2.5. Multipath Protocols

Multipath TCP [RFC8684] and SCTP utilize multiple data paths over multiple endpoint addresses. As these multipath protocols are unaware of data paths, they distinguish the data paths by endpoint IP addresses. Accordingly, multiple flows of these protocols may use an identical data path without recognizing it. Thus, multipath protocols of the transport layer implement a mechanism to handle multiple data paths in terms of endpoint IP addresses, although it is not sufficient for the data path management.

Multipath protocols are also responsible for inverse multiplexing to split a data stream into multiple data paths. This inverse multiplexing is collectively performed with flow control and retransmission. It is independent of data paths except for congestion control as the inverse multiplexing as well as flow control and retransmission do not need to be aware of data paths but congestion control is performed on each data path.

2.6. Reliable Data Communication

Transport layer protocols may implement retransmission and reordering functions to recover lost or reordered datagrams for reliable data communication. This functionality is independent of data paths, and consequently, should be implemented over data paths. Instead, ''smart'' end-hosts or middleboxes may implement it.

An end-host may transmit a duplicate (replicate) packet for improving reliability [RFC8655]. This functionality is also independent of data paths. However, a router in a data path may be capable of duplicating a packet because the duplication process does not require significant computing resources, unlike retransmission. This duplication (replication) functionality may be implemented in other layers than the transport layer, such as the link layer and the Internet layer, as the DetNet data plane [RFC8938] does.

2.7. Security

The transport layer may implement an end-to-end security function, such as Transport Layer Security (TLS) [RFC8446] and Datagram Transport Layer Security (DTLS) [RFC6347]. As TLS does not encrypt the underlying transport layer protocol header, middleboxes such as TCP offloading can still work. However, some protocols implementing a transport layer protocol over DTLS, such as SCTP over DTLS used in WebRTC Data Channel [RFC8831] and QUIC [RFC9000], encrypt the transport layer header, and consequently, have difficulty cooperating with these middleboxes, as addressed in RFC 9065 [RFC9065]. RFC 9065 [RFC9065] suggests to add OAM information to network layer headers, while this document proposes such information to data path layer headers.

In this way, transport layer's security functions have focused on end-to-end security. However, middleboxes that terminate a security function, such as mcTLS [mcTLS], have been proposed. Moreover, distributed computing paradigms such as MEC may also introduce middleboxes allowed to terminate security functions to deploy a computing function. Therefore, security should be considered at three levels. data path security, data flow security, and end-to-end security. Note that data link security such as Wi-Fi Protected Access (WPA) is out of the scope of this document, although data path layer security may adopt data link security for hop-by-hop security.

2.8. Summary

As described above, the transport layer functionality is summarized by distinguishing data paths and data flows as follows:

The following functionality is categorized as data path functions.

* In-band trajectory monitoring
* Waypoint management
* Bidirectional connection establishment
* One-to-many connection for multicast
* Data path quality (congestion) monitoring and congestion control
* Data flow multiplexing
* Packet duplication
* Data path security

The following functionality is categorized as data flow functions.

* Retransmission for reliable data communication

* Flow control for receive-buffer management

* Flow prioritization

* Data flow and end-to-end security

* Inverse multiplexing for multipath protocols

3. Specifications of Data Path Layer and Data Flow Layer

As summarized in Section 2.8, this document separates data paths and data flows from the transport layer. Hence, this document divides the transport layer into two sublayers; the data path and the data flow layers. This section specifies the functionality of these two sublayers.

3.1. Data Path Layer

The data path layer provides the functionality to make the transport layer aware of data paths upon the Internet layer. The data path layer forwards packets without manipulating the payload. It means that the data path layer does not provide fragmentation, segmentation, or reassembly. Instead, the Internet layer or the data flow layer provide these functions.

The data path layer MAY implement in-band trajectory monitoring, bidirectional connection establishment, one-to-many connection for multicast, data path quality monitoring and congestion control, data flow multiplexing, and packet duplication. A data path layer protocol MAY implement other functions related to data paths.

3.1.1. In-band trajectory monitoring

The data path layer MAY provide an in-band trajectory monitoring function. The in-band trajectory monitoring function enables an upper-layer protocol to detect path change events and a single point of failure for multipath protocols.

DPH: Data path layer protocol header

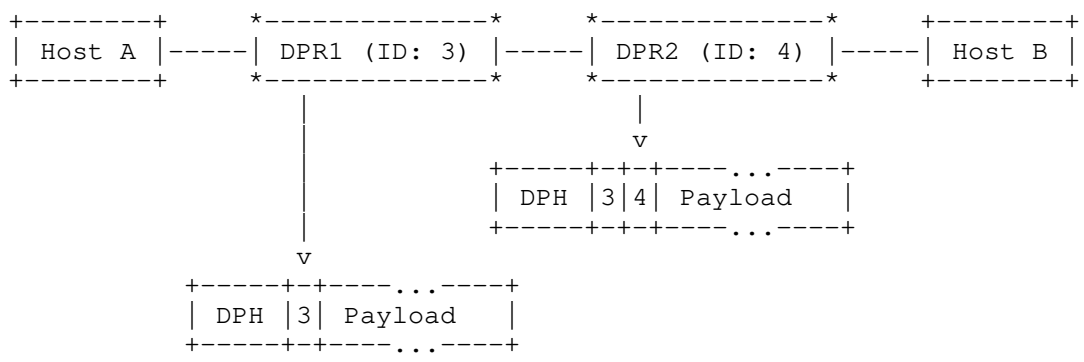


Figure 2: Appending data path router identifiers to distinguish the data path

One approach to implement this functionality is prepending, appending, or XOR-ing device identifiers like in-band network telemetry or in-situ OAM [RFC9197] by data path layer protocol's routers. Figure 2 shows an example of in-band trajectory monitoring. A router that handles a data path layer protocol appends the router identifier to the header. In case the upper-layer protocol does not require the path information but path change events, the router can apply an XOR operation with a hashed identifier to a fixed-length field. Other alternative approaches MAY be used.

The latter approach separates the control plane and the data plane. The control plane manages the data path, and the data plane monitors if packets pass through the dedicated data path. The data path control MAY leverage underlay protocols such as Segment Routing.

3.1.2. Waypoint management

The data path layer MAY support waypoint management functionality. The in-band trajectory monitoring functionality described above does not provide the functionality to designate the data path's waypoints. However, some middleboxes such as firewalls and in-network computing require to designate waypoints to activate the in-network functions.

A data path layer protocol MAY define a specification to control the waypoints of a data path. There are two approaches to designate waypoints over the Internet layer; 1) setting a waypoint as the destination IP address and 2) using routing technologies such as source routing and policy-based routing. The former approach includes Network Address Translation (NAT) [RFC3022] and proxy services. The latter approach MAY leverage underlay protocols to

designate waypoints, such as Segment Routing and IPv6 Type 2 Routing Header [RFC6275]. Other alternative approaches MAY be used in a data path layer protocol.

Note that this waypoint management functionality could be considered as the Internet layer's functionality like policy-based routing, source routing, and segment routing. This document considers the waypoint management is the functionality of the data path layer. The data path layer interacts with the Internet layer to function the waypoint management.

3.1.3. Bidirectional connection establishment

The data path layer SHOULD support both unidirectional and bidirectional paths as unicast is the most fundamental communication model. A bidirectional path MAY be symmetric or asymmetric. As the characteristics of unidirectional and bidirectional paths are different, some data path layer protocols MAY only support either unidirectional or bidirectional paths.

A data path protocol supporting bidirectional data paths SHOULD implement a handshake mechanism to establish a bidirectional connection, such as a 3-way handshake of TCP and a 4-way handshake of SCTP. It MAY provide a 0-RTT connection establishment feature such as TLS 1.3 [RFC8446] and TCP Fast Open [RFC7413].

3.1.4. One-to-many connection

The data path layer MAY support one-to-many connection for multicast. In addition to the one-to-many connection, it MAY support many-to-one (reverse) connection. These types of connections are beneficial to use cases such as multi-access edge computing (Section 4.3) and in-network computing (Section 4.4).

3.1.5. Data path quality (congestion) monitoring and congestion control

The data path layer SHOULD implement congestion control. However, Data path layer protocols supporting only unidirectional paths MAY not implement congestion control because it cannot implement congestion or data path quality reporting.

Congestion control is performed based on data path quality or congestion reporting from the receiver end-host or intermediate nodes that process a data path layer protocol. Data path quality or congestion signals MAY be packet losses, delay, or ECN, as used in many transport layer protocols. Other signals or metrics, such as in-band network telemetry information, MAY be used.

3.1.6. Data flow multiplexing

The data path layer enables us to collectively handle different characteristics (e.g., service level requirements) of transport layer protocols such as stream and datagram protocols.

3.1.7. Packet duplication

On a lossy data path, a data path layer protocol MAY implement packet duplication for improving reliability. However, the data path layer SHOULD NOT provide retransmission because it requires significant resources.

3.1.8. Data path security

A data path layer protocol MAY implement data path security. Data path security provides a security function between data path routers. As pointed out in Section 2.7, the data path layer assumes middleboxes that terminate a security function. Therefore, data (payload) encryption provided by data path security MAY be terminated at a data path router. In addition to encryption, other security functions, such as authentication, authorization, and accounting for a data path router, MAY be implemented as data path security. Data path layer MAY implement mutual authentication mechanisms, such as EAP-TLS [RFC5216].

3.2. Data Flow Layer

The data flow layer MAY implement various functions for end-to-end data communication over the data path layer. Data flow layer protocols MAY be stream, datagram, or message-oriented protocols. Middleboxes, MEC nodes, or in-network computing nodes MAY process data flow layer protocols. Therefore, a node processing data flow layer protocols SHOULD be as ``smart'' as end-hosts.

A data flow layer protocol MAY implement a retransmission function for reliable data communication, a flow control mechanism for receive-buffer management, flow prioritization for effective network resource utilization, a security extension such as TLS and DTLS, and a multipath transport layer protocol using multiple bidirectional paths over the data path layer.

3.2.1. Retransmission for reliable data communication

A data flow layer protocol MAY provide retransmission for reliable data communication. To this end, a node implementing the data flow layer protocol MUST equip a transmit buffer for retransmission to retransmit lost corrupted packets. It MUST also equip a receive buffer to reassemble a stream and a message from a sequence of packets in a stream and message-oriented data flow layer protocols. The receive buffer also handles packet reordering and duplication.

3.2.2. Flow control for receive-buffer management

Flow control is necessary for receive-buffer management. Therefore, a data flow layer protocol MAY implement the flow control mechanism. A receiver node advertises the available receive-buffer size in the data flow layer, like TCP's window size, when implementing the flow control mechanism. A sender node controls the transmission so that transmitted data do not exceed the receive-buffer size.

3.2.3. Flow prioritization

The prioritization of data flows multiplexed in a data path layer protocol MAY be implemented on the data flow layer using the data path layer information, such as monitored data path quality or congestion. A data flow layer protocol MAY provide a service code point or priority so that nodes processing the data flow protocol discriminate a flow.

3.2.4. Data flow and end-to-end security

A data flow layer protocol MAY implement a data flow or end-to-end security function. TLS and DTLS can be used for stream and datagram data flows, respectively. In such cases, data flow security is identical to end-to-end security. If we consider a mechanism that terminates a security function for a data flow, such as [mcTLS], data flow security is handled by a middlebox.

3.2.5. Inverse multiplexing for multipath protocols

A multipath data flow protocol MAY leverage multiple bidirectional data paths established by data path layer protocols. The data flow layer is responsible for the inverse multiplexing functionality. Therefore, the multipath data flow protocol divides a stream, a message, or a datagram into these multiple data paths. Note that the data path layer performs congestion control for each data path, while the data flow layer performs flow control.

4. Use Cases

This document does not specify any data path protocols or data flow protocols, but the data path and data flow layers' architectural design. For better understanding, this section describes the use cases of the data path and the data flow layers. In the use cases, a data path router (DPR) and a data flow layer node (DFN) denote a router that processes a data path layer protocol and a node that processes a data flow layer protocol, respectively.

4.1. Multipath Transport Layer Protocols

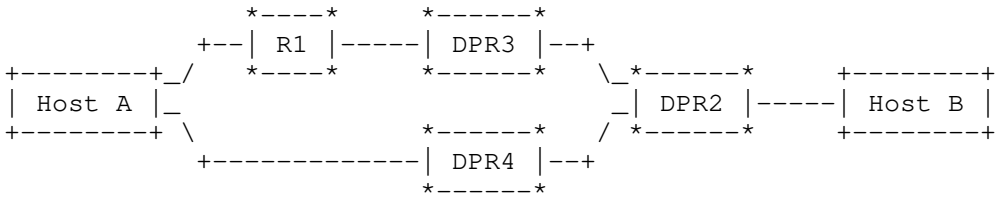


Figure 3: An example of multipath data communication

Figure 3 shows an example of multipath data communication over two data paths. The end-hosts A and B establish two bidirectional data paths; A-R1-DPR3-R2-B and A-DPR4-DPR2-B, and they use a data flow layer protocol over these data paths for end-to-end communication. The data path layer protocols monitor the data path quality and perform congestion control for each data path. The data flow layer protocol performs flow control and inverse multiplexing into these data paths.

Multipath transport layer protocols MAY leverage in-band trajectory monitoring to detect a shared waypoint. We assume that the data path routers manipulate the header of a data path layer protocol. They add the router identifier to the data path layer protocol's header to report the trajectory to the end-hosts. When sending packets from end-host A to end-host B over these paths, each packet reports trajectory A-DPR3-DPR2-B or A-DPR4-DPR2-B. In this way, end-host B recognizes two different data paths and detects a shared data path router, DPR2, on the multiple paths, potentially a single point of failure for end-to-end communication.

4.2. Congestion Control Acceleration

Some TCP congestion control acceleration functions proxy TCP sessions to shorten the apparent latency. The acceleration functions acknowledge receipt of packets. Using a loss-based congestion control algorithm, both congestion control and flow control use the acknowledgments (ACKs). For congestion control, missing ACKs report packet losses, and consequently, they present the data path quality to control the transmission rate to avoid congestion. For flow control, ACKs report successful data delivery. Then, the sender can release part of the transmit buffer. Otherwise, the sender retransmits the lost data.

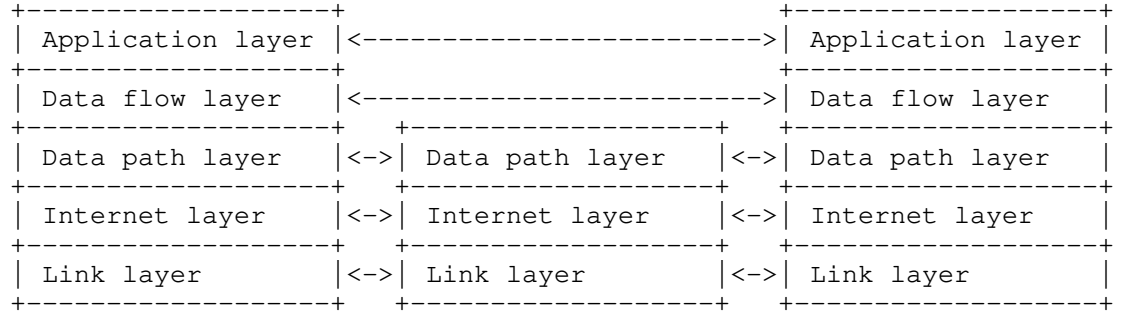


Figure 4: An example of a communication model for congestion control acceleration

Figure 4 illustrates an example of a communication model for congestion control acceleration using the data path layer. The accelerator or the receiver reports the data path quality to the sender on the data path layer to perform congestion control. There are various ways to measure data path quality. For example, data path routers MAY use the buffer capacity in the same way as ECN. If a data path router can detect packet losses for a connection of data path layer protocols, the packet losses or statistics MAY be used to report the data path quality. The data path quality report by the accelerator enables the fast response of congestion control algorithms.

The data flow layer is responsible for retransmission for reliable communication and flow control for receive-buffer management. The accelerator MAY implement a data flow layer protocol as well as the data path layer protocol to proxy data flows for these functions.

4.3. Multi-access Edge Computing

Multi-access Edge Computing (MEC) enables application service providers to deploy computing resources into an access network edge, where is closer to end-hosts (subscribers, users or clients) than data centers. As a MEC node is also an end-host, it can use any transport layer protocols.

A challenge of a transport layer protocol for MEC is that MEC needs to take into account the proximity for low latency and efficient resource utilization. Therefore, it requires to be aware of data paths and their QoS. In-band trajectory monitoring and waypoint management of the data path layer enable this proximity-aware data transport. Moreover, an end-host that uses MEC may be mobile and move to other networks. The data path layer MAY implement mobility support, by enhancing the existing mobility support mechansims such as Mobile IP [RFC5944].

Furthermore, a MEC node could be used as a middlebox, such as data buffer for streaming communication, transcoding, data filtering, and a broker for pub/sub communication, that collectively works with high-performance computing resources in data centers. In such cases, data flows are terminated by a MEC node. In the conventional techniques, building an overlay network is one of the solutions for these cases to terminate and proxy data flows. The data flow layer MAY abstract and implement part of functionality of overlay networks, as discussed in Section 4.4.

4.4. In-Network Computing

In-network computing [I-D.irtf-coinrg-use-cases] is a novel distributed computing paradigm. It requires to be aware of the data path's waypoints because computing components are placed in between end-hosts. The message-oriented protocol MAY adopt the pub/sub communication model, widely used in machine-to-machine communication.

The data path layer establishes a data path while designating the waypoints to perform in-network computing. The data flow layer over the data path implements a message-oriented protocol or a stream protocol to feed data into in-network computing nodes. The message-oriented protocol MAY adopt the pub/sub communication model, widely used in machine-to-machine communication.

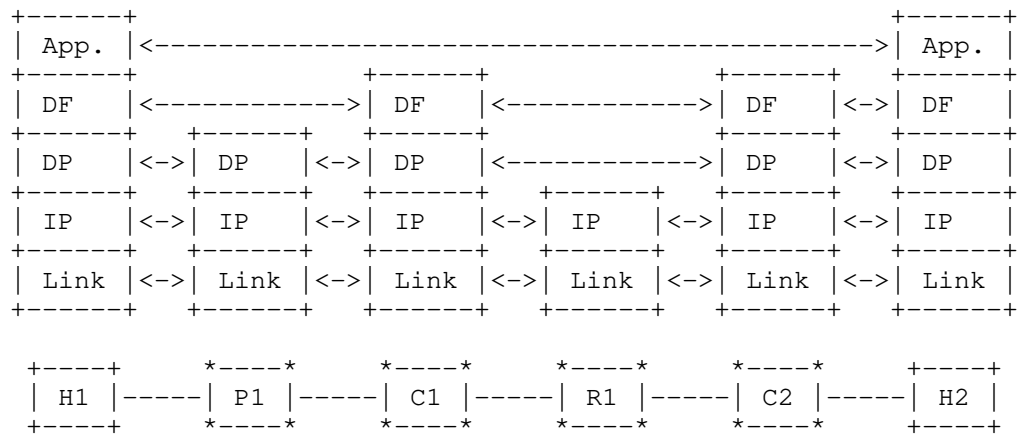


Figure 5: An example of communication model of in-network computing

Figure 5 illustrates an example of the communication model of in-network computing. A data path is established between H1 and H2. A data path layer protocol designates C1 and C2 as the waypoints. Over the established data path, H1 transmits messages to H2 using a data flow layer protocol. C1 and C2 process the messages according to the programs running on C1 and C2.

4.5. Flow Arbitration

The separation of the data path and the data flow layer enables flow-level prioritization. As the data path layer is responsible for congestion control, the multiplexed data flows over a data path can collectively perform resource arbitration for the data path bandwidth capacity.

For example, a data path multiplexes two data flows; one of them requires a constant data rate, and the other is tolerant of delayed data delivery. Flow control can prioritize the former data flow and decrease the transmission rate when the data path layer detects congestion. This flow arbitration is crucial in coexisting deadline-based and best-effort flows.

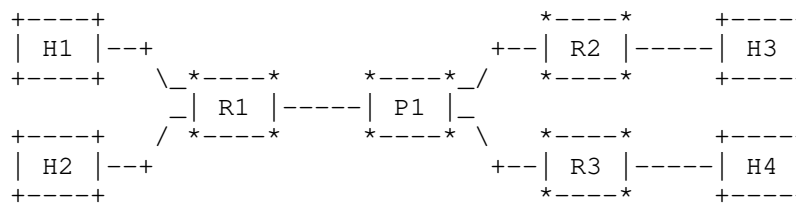


Figure 6: An example of flow arbitration over multiple data paths

Moreover, the data path layer MAY support data path quality monitoring for multiple data paths. If data path routers monitor multiple data paths' bandwidth capacity, they can report the estimated capacity to arbitrate multiple flows over the data paths. Figure 6 shows an example flow arbitration over multiple data paths. Suppose data flows X and Y are over the data paths H1-R1-P1-R2-H3 and H2-R1-P1-R3-H4, respectively; the data path router P1 can monitor the data path quality such as packet losses for these data paths. If the bandwidth utilization reaches the capacity, the data path router requests resource arbitration. A flow MAY reduce the data rate using the flow control mechanism in the data flow layer if its priority is not high. Thus, these two flows can autonomously perform the flow arbitration. This flow arbitration MAY fail and perform best-effort communication, such in case both flows are the high priority and do not reduce the data rate.

5. Implementation Considerations

Although this document does not specify the data path layer and the data flow layer protocols, this section discusses the implementation of these protocols' specifications. This document presents the following three implementation models for data path layer protocols.

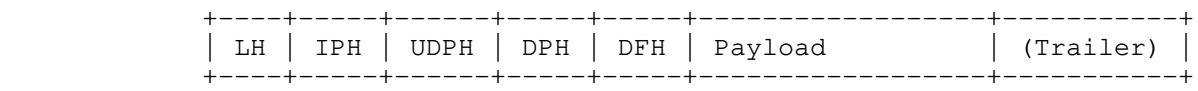
5.1. Data Path Layer Protocol over IP

The most straightforward way to implement a data path layer protocol is over the Internet Protocol (IP) by using the Protocol field specified in RFC 791 [RFC0791] in IP version 4 (IPv4) or the Next Header field specified in RFC 8200 [RFC8200] in IP version 6 (IPv6).

This model, however, has an issue that routers and middleboxes may drop the data path layer protocol packets as other protocols than ICMP, TCP, and UDP may be filtered on the Internet.

5.2. Data Path Layer Protocol over UDP with Port Number

An alternative model to implement a data path layer protocol is to use UDP to encapsulate the data path layer protocol packets in UDP. To identify a data path layer protocol packet in UDP, a specific port number can be used. However, this model adds semantics to a port number interpretation as a specific data path layer protocol is intended to be processed by some middleboxes between the end-hosts.



* LH: Link-layer protocol header
* IPH: IP header
* UDPH: UDP header
* DPH: Data path layer protocol header
* DFH: Data flow layer protocol header

Figure 7: An implementation of data path layer and data flow layer protocols over UDP

Figure 7 illustrates an example packet format of data path layer and data flow layer protocols over UDP. In this figure, the UDP header is not intended to provide the transport layer's functionality but is introduced for interoperability with existing middleboxes on the Internet.

A data path layer protocol MAY use hop-by-hop UDP connections to designate waypoints like an overlay network. Otherwise, data path routers require to leverage routing technologies such as Segment Routing and policy-based routing to support waypoint management.

6. IANA Considerations

This document does not have IANA Considerations.

7. Security Considerations

End-to-end encryption becomes critical to Internet protocols for privacy and security. The data path layer MUST be visible to intermediate nodes, such as routers and middleboxes, by design to process and manipulate a data path layer protocol. Therefore, this document proposes to implement transport layer security at the data flow layer. It exposes data path information. Therefore, data path layer protocols MUST NOT include privacy-sensitive information in the protocol header.

Data path layer protocols may be invisible to intermediate nodes if an underlay protocol encrypts the payload. For example, IPsec [RFC6071] encrypts the payload of IP datagrams. In such cases, intermediate nodes cannot process or manipulate data path layer protocols.

8. Acknowledgements

We thank Kenichi Murata, Ryokichi Onishi, Yusuke Doi, and Masahiro Ishiyama for their comments and review of this document.

9. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

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

[RFC0919] Mogul, J., "Broadcasting Internet Datagrams", STD 5, RFC 919, DOI 10.17487/RFC0919, October 1984, <https://www.rfc-editor.org/info/rfc919>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <https://www.rfc-editor.org/info/rfc5216>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <https://www.rfc-editor.org/info/rfc5944>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <https://www.rfc-editor.org/info/rfc6275>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[RFC8831] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channels", RFC 8831, DOI 10.17487/RFC8831, January 2021, <https://www.rfc-editor.org/info/rfc8831>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9197] Brockners, F., Ed., Bhandari, S., Ed., and T. Mizrahi, Ed., "Data Fields for In Situ Operations, Administration, and Maintenance (IOAM)", RFC 9197, DOI 10.17487/RFC9197, May 2022, <https://www.rfc-editor.org/info/rfc9197>.

10. Informative References

[RFC1301] Armstrong, S., Freier, A., and K. Marzullo, "Multicast Transport Protocol", RFC 1301, DOI 10.17487/RFC1301, February 1992, <https://www.rfc-editor.org/info/rfc1301>.

[RFC1458] Braudes, R. and S. Zabele, "Requirements for Multicast Protocols", RFC 1458, DOI 10.17487/RFC1458, May 1993, <https://www.rfc-editor.org/info/rfc1458>.

[RFC2768] Aiken, B., Strassner, J., Carpenter, B., Foster, I., Lynch, C., Mambretti, J., Moore, R., and B. Teitelbaum, "Network Policy and Services: A Report of a Workshop on Middleware", RFC 2768, DOI 10.17487/RFC2768, February 2000, <https://www.rfc-editor.org/info/rfc2768>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, DOI 10.17487/RFC3048, January 2001, <https://www.rfc-editor.org/info/rfc3048>.

[RFC3208] Speakman, T., Crowcroft, J., Gemmell, J., Farinacci, D., Lin, S., Leshchiner, D., Luby, M., Montgomery, T., Rizzo, L., Tweedly, A., Bhaskar, N., Edmonstone, R., Sumanasekera, R., and L. Vicisano, "PGM Reliable Transport Protocol Specification", RFC 3208, DOI 10.17487/RFC3208, December 2001, <https://www.rfc-editor.org/info/rfc3208>.

[RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <https://www.rfc-editor.org/info/rfc3973>.

[RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, DOI 10.17487/RFC4654, August 2006, <https://www.rfc-editor.org/info/rfc4654>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[RFC6559] Farinacci, D., Wijnands, IJ., Venaas, S., and M. Napierala, "A Reliable Transport Mechanism for PIM", RFC 6559, DOI 10.17487/RFC6559, March 2012, <https://www.rfc-editor.org/info/rfc6559>.

[RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <https://www.rfc-editor.org/info/rfc7094>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://www.rfc-editor.org/info/rfc7413>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[RFC9065] Fairhurst, G. and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", RFC 9065, DOI 10.17487/RFC9065, July 2021, <https://www.rfc-editor.org/info/rfc9065>.

[I-D.irtf-coinrg-use-cases] Kunze, I., Wehrle, K., Trossen, D., Montpetit, M., de Foy, X., Griffin, D., and M. Rio, "Use Cases for In-Network Computing", Work in Progress, Internet-Draft, draft-irtf-coinrg-use-cases-04, 30 June 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-coinrg-use-cases-04>.

[mcTLS] Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., López, D., Papagiannaki, K., Rodriguez, P., and P. Steenkiste, "Multi-Context TLS (mcTLS) Enabling Secure In-Network Functionality in TLS", SIGCOMM Comput. Commun. Rev., vol. 45, no. 4, pp. 199-212, 2015.

Author's Address

Hirochika Asai
Preferred Networks, Inc. / WIDE Project
1-6-1 Otemachi, Chiyoda, Tokyo
100-0004
Japan
Email: panda@wide.ad.jp