Network Working Group
Internet-Draft
Intended Status: Experimental
Expires: July 22, 2014

W. Lei
W. Zhang
S. Liu
Northeastern University
January 23, 2014

A Framework of Multipath Transport System Based on Application-Level Relay (MPTS-AR) draft-leiwm-tsvwg-mpts-ar-01.txt

Abstract

Multipath transport is an important way to improve the efficiency of data delivery. This document defines a multipath transport system framework in which application-level relays are deployed to provide the conditions to enable multiple paths between source and destination. In the proposed framework, endpoints are allowed to use multiple paths, including the default IP path and relay paths, to transport data in a single session. A relay path may via one or more application-level relays which provide application-level relay services for endpoints. This framework defines three kinds of logical entities including user agent, relay server and relay controller. Relay server provides relay service for user agents based on a local path-table. Relay controller manages relay servers and relay paths. User agent maintains multiple end-to-end paths which include a default path and multiple relay paths. The framework also defines a relay service control protocol named OpenPath protocol in control plane to manage relay servers and relay paths, and a profile of multipath transport protocol suite in data plane to facilitate multipath data transport. The multipath transport system framework can support various applications including applications requiring timely delivery of real-time data such as streaming media, and applications requiring ordered reliable delivery of stream of data such as file transfer.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright Notice

Table of Contents

1. Introduction

For end-to-end multimedia session, multipath transport has more benefits than single path transmission. Multiple disjoint (or partially disjoint) paths could provide greater transmission bandwidth, and redundancy among multiple disjoint paths increases transmission reliability by protecting multiple paths from failure of one, so multipath transport can promote quality of transmission service. Moreover, from the perspective of whole network transmission efficacy, multipath transport can achieve load balance and increase the efficiency of the network resource usage.

In order to achieve multipath transport, the following two problems need to be considered: 1) how to build multiple paths between communication endpoints, and 2) given multipath paths, how to implement multipath transport between communication endpoints.

The current underlying IP routing protocol can only build a single transport path between endpoints. This implies that although the Internet routing infrastructure is highly redundant, current underlying routing protocols fail to fully utilize the network redundancy. And it is nearly impossible to update protocol stack of existing network devices to support multipath routing due to tremendous cost. Now the main usage scenario of multipath transport is based on multi-homed host, which requires at least one of communicating endpoints is multi-homed. The multi-homed host multipath usage scenario relies on the end host equipped with several access networks, which is hard to be satisfied in practical applications.

An alternative approach for establishing multiple paths between source and destination is to provide support mechanisms in the application layer while retaining the underlying network infrastructure and end devices. This scheme of multipath transport is a kind of overlay network technologies actually. Overlay networks have emerged as an effective way to support new applications, as well as protocols without any changes in the underlying network layer. Multiple disjoint (partially disjoint) transmission paths, which pass one or more application-level overlay nodes, can be established between end hosts. This method is compatible with existing protocol stack, and gets rid of the restrictions of physical network conditions. Therefore, it is relatively easier to establish multipath transport scenario.

This document defines a framework of multipath transport system based on application-level relay (MPTS-AR). A large amount of application-level overlay nodes are deployed to provide relay service to the communicating endpoints. The upper application programs are provided

with opportunities to autonomously select one or multiple paths, including the default IP path and relay paths, to transport data in a session. A relay path may go through one or multiple overlay nodes which provide relay services for endpoints. This framework defines three kinds of logical entities including user agent, relay server and relay controller, and describes their function blocks and behaviors. The framework also defines a relay service control protocol named OpenPath protocol in control plane to manage relay paths, and a profile of multipath transport protocol suite in data plane to facilitate multipath data transport.

Relay controller and relay servers constitute a relay service system, which provides relay service to user agents. Relay controller is responsible for managing relay servers, allocating relay paths, QoS condition evaluation and so on. It also provides a unified access interface of relay service to user agents or out-of-band signaling entities. Main function of relay server is to complete the application-level data forwarding, by receiving media stream from the address and port of last hop, and then forwarding to the address and port of next hop. A relay path may pass one or more relay servers.

OpenPath, relay service control protocol, mainly includes two kinds of messages: the first is exchanged between relay controller and relay server which is used to manage relay servers and relay paths. Any application software or special server, which implements data relay services and supports OpenPath protocol as described in this document, can dynamically register to a relay controller and provide relay service for user agents in the region of this relay controller. The second is exchanged between relay controller and user agent or out-of-band signaling entity which is used to manage relay path including relay path inquiry, establishment, teardown, etc.

Transport requirements of various applications may be quite diverse. These applications are sensitive to different routing metrics such as latency, loss, throughput and so on. In order to support a variety of applications, the proposed MPTS-AR needs to work with a suite of multipath transport protocol (MPTP) which consists of multiple application-specific MPTPs. Each application-specific MPTP is aimed to meet the transmission requirements of a specific category of upper applications. In order to extend easily a new application-specific MPTP for an emerging application and simplify implementation of user agents, this document gives a common profile for all application-specific MPTPs. MPTP profile provides application-level multipath routing mechanism which is common in all application-specific MPTPs. All application-specific MPTPs MUST follow the common rules defined by MPTP profile. This document gives the definition of MPTP profile. Application-specific MPTPs are defined in companion documents.

Main principles for designing this framework include:

1) Establishment of multipath transport scenarios depends on the relay service provided by relay service system. Relay server does not care end-to-end transmission characteristics of data flows forwarded by it. Therefore, a variety of upper applications can use multipath transport services provided by the relay service system.

2) Management and service access interfaces of relay service are standardized so that any organizations and individuals can provide specialized relay services.

3) The MPTP profile defines common rules of multipath transport based on application-level relay for various upper applications. To support a specific category of upper applications, a corresponding application-specific MPTP should be defined in an additional document.

2. Terminology

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].

3. Definitions

1) Subflow: flow of data packets along a specific path, i.e., a subset of the packets belonging to a data flow. The combination of all subflows forms the complete original data flow. Typically, a subflow should map to a unique path.

2) Session: an association between two participants transmitting data. An endpoint may be involved in multiple sessions at the same time. For example, in a multimedia session, each medium is typically carried in a separate real-time transport protocol (RTP) [3] session which is a type of 'session' defined here. Another typical example of session is to transfer one or multiple files between two endpoints. This framework allows the variations defined here for different applications.

3) Multipath Session: a special type of session in which the original data flow is split into multiple subflows and each subflow is forwarded along a unique path.

4) Path: sequence of logical links between a source and a destination. We define it by a set of addresses. All available paths for a session include a default path and one or more relay paths.

5) Default path: a path between a sender and a receiver, which is same as the path negotiated and established by a normal session. In Session Initiation Protocol (SIP) [4] and Session Description Protocol (SDP) [5] case, the default path is determined by the c= and m= lines in SDP during session setup. If either the source or the destination is not behind a symmetric NAT, the default path may be the direct network path between the source and the destination. Otherwise, it may traverse a third-party node, such as a TURN server or a media server which is responsible for relaying packets.

6) Relay path: a path via one or multiple relay servers between a source and a destination. A relay path is defined by a sequence of (S, R1, ..., Rm, D), where Ri denotes the address of the i-th relay server and m denotes the number of relay servers in this path.

7) Candidate path: a path that is either a default path or a relay path.

8) Active path: a path that carries media data.

9) User Agent: a logical entity that can act as either a sender or a receiver. A sender is responsible for managing all candidate paths and multipath session, partitioning and scheduling subflows, and packaging MPTP packets. A receiver is responsible for recombinating subflow packets and sending back QoS feedback to the sender for each subflow. The behavior of a user agent is further defined in Section 6.

10) Relay server: an intermediary entity that primarily plays the role of forwarding subflow packets according to a Path-Table. Relay server receives subflow packets from its upstream entity of the subflow that the received packets belong to, obtains the next-hop transport address based on the matching results in the Path-Table, and forwards the packets to the corresponding next-hop transport address. The behavior of a relay server is further defined in Section 7.

11) Path-Table: a table consisting of a set of path entries, which may be added, updated or deleted by a remote relay controller via OpenPath protocol. Each relay server maintains its own path-table.

12) Relay Controller: a logical entity that manages all relay servers in its region and allocates relay paths for each session. The behavior of a relay controller is further defined in Section 8.

13) Overlay Link: the connection between two relay servers. It is usually composed of one or more physical links.

4. Overview

In the multipath transport system based on application-level relay, relay server provides application-level data forwarding service, which can used to establish multiple relay paths between communication endpoints to achieve multipath transport. With appropriate multipath transport control mechanisms and protocols, not only can it achieve higher quality of end-to-end transmission service, but it balance network load and enhance efficiency of the network resource usage.

Figure 1 illustrates the structure of the proposed multipath transport system framework and the relationship among user agent, relay server and relay controller. Relay controller and relay server constitute the relay service system which provides relay service to user agent. User agent maintains multiple end-to-end relay paths, and dispatches data flow along multipath paths following MPTP protocol suite.

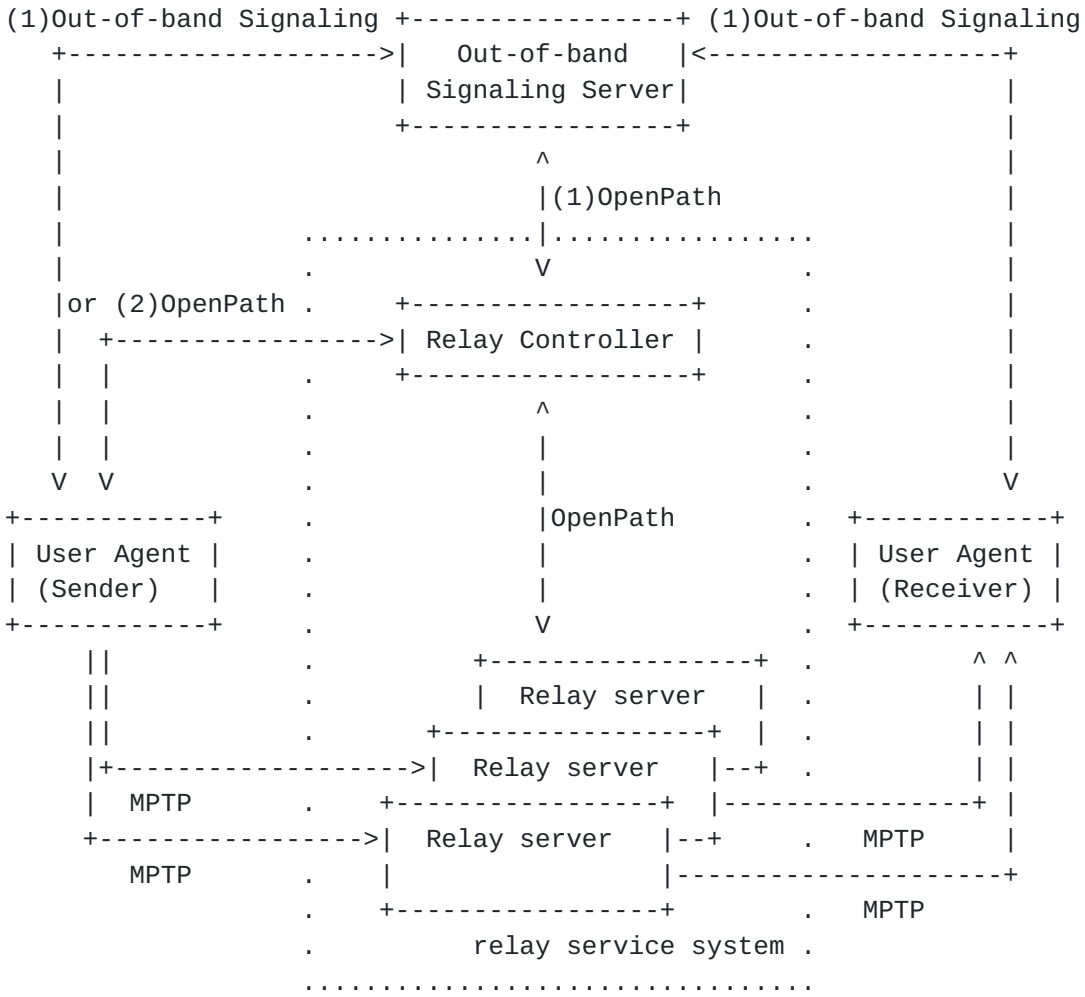


Figure 1. The structure of the multipath transport system framework based on application-level relay (MPTS-AR)

4.1 Deployment and organization of relay controller and relay server

The relay controller is the central component of the relay service system. The main functions of relay controller are to manage relay servers, evaluate the QoS conditions of relay paths, and provide relay path service to user agents (or provide relay path service to user agent indirectly through out-of-band signaling server). Relay controller that is usually deployed in the core of the network by the overlay service provider usually possess high-capacity computing, storage, and bandwidth resources. When there are a large number of user agents, multiple controllers can be deployed and each of them provides management service for part of user agents.

Relay servers provide data relay service to the communicating agents. Considering the efficiency of the forwarding service, MPTS-AR adopts UDP as the underlying transport protocol. Relay servers may be deployed in a variety of ways. In one way, a number of Internet service providers (ISP) can actively deploy proprietary overlay nodes with high network bandwidth and computing performance in their domains and offer them to overlay service providers (OSP). ISPs may also provide additional support for the operation of the overlay network to the OSP and charge OSP for the resources and support provided. The OSP can lease a number of overlay nodes from multiple ISPs and deploy multipath transport system on top of them.The OSP offers multipath transport service to interested users and charges the users for the services. In this way, overlay networks can be enhanced with available routing information to reduce path quality measurement costs and to provide best routes. In another way, the participating nodes that access the same application may also self-organize to form a dynamic overlay network among which the nodes with higher performance can provide relay service to others. The organization of relay servers is a kind of typical overlay network technology. Overlay network model of relay servers is closely related to the generation and selection of relay paths so that this document does not recommend any specific overlay network model, and only rules that all relay servers need to register to a relay controller and provide data relay service to user agents.

4.2 Relay path service provided by relay controller

All relay servers register to a relay controller in their region. Relay controller is responsible for managing the relay servers and providing direct or indirect relay path services to the user agent by OpenPath protocol including establishing, inquiring, and removing of relay paths.

Considering the execution efficiency, the relay path is designed to be unidirectional. A bidirectional data flow, such as in a conversational use-case, is regarded as two independent unidirectional flows in opposite directions. Relay paths are assigned for each unidirectional flow.

User agent sender, who is responsible for sending out data flow, and relay servers need to know the relay path information so they can correctly forward subflow data along a particular relay path. An alternative approach, similar to source routing, is that the user agent sender can store the entire route in MPTP packet headers. Each intermediate node will simply examine the headers of a received packet and forward it to the next node as indicated in the source route. The advantage of the method is to simplify the implementation of relay servers. They need not store any path information and can perform routing of MPTP packets only based on MPTP packet headers. But this method introduces traffic overhead considerably, especially when the payload traffic is relatively small.

In practice, the user agent sender and relay servers need not to know the complete information of the associated path. They just need to know the next-hop transport address for each path associated with them. A pair of transport address comprises one network address plus one port. When receiving OpenPath path allocation request messages from user agents directly or indirectly (for example via signaling server such as SIP or RTSP), relay controller generates a set of disjoint relay paths based on information carried by the OpenPath path allocation request message including the media types, quality of experience (QoE) expectations or requirements, the number of relay paths expected, and so on. Relay controller associates a unique path identifier to each relay path. On the one hand, relay controller instructs the corresponding relay servers to assign the appropriate resources for incoming data forwarding. On the other hand, relay controller sends OpenPath response message back to the service requester. Usually, the response message includes the path identifier, the relay service address of first-hop relay server and instant QoS of each path. This document neither defines a uniform relay path generation algorithm, nor recommends QoS evaluation method of relay path. These methods can be designed by the overlay service provider.

Relay controller needs to maintain the mapping between the user agent and the allocated relay paths. When receiving OpenPath path removal request messages, relay controller instructs the associated relay servers to release resources and generates a response message back to the requester.

4.3 End-to-end transmission paths managed by user agent

End-to-end transmission paths between user agents includes two types: 1) default path (DP), which does not go through any relay server; 2) relay path (RP), which goes through one or more relay servers.

In order to establish multiple end-to-end transmission paths, user agent sender, needs to collect candidate paths before transmitting data. As shown in figure 1, this document provides two alternative ways to be used for collecting candidate paths by the user agent sender. In the first way, the user agent sender obtains candidate paths from the out-of-band signaling used for establishing an association between the user agent sender and user agent receiver. More specifically, user agents use a kind of out-of-band signaling (e.g., SIP, Real-Time Streaming Protocol (RTSP) [6]) to negotiate and establish a session with the remote peer before transmitting data. The signaling server, which out-of-band signaling passes through during session setup, is extended to support the access interfaces of relay path service provided by OpenPath protocol. The signaling server requests the relay controller to allocate relay paths for the session to be established on behalf of the user agent sender. If successful, the signaling server inserts the information of the allocated relay paths into corresponding signaling messages to inform the participating user agents. In the second way, the user agent sender obtains candidate paths through direct interaction with the relay controller using OpenPath Protocol. The advantage of the first way is to avoid a large number of connections of the relay controller with user agents, and improve the security of the relay controller through limiting communication only with the trusted signaling server.

As mentioned above, relay path is designed to be unidirectional. A bidirectional data flow is regarded as two independent unidirectional flows in opposite directions. Two participating nodes of the bidirectional session are the user agent sender of the corresponding unidirectional flows respectively. Both user agent senders needs to collect candidate paths for corresponding unidirectional flow.

After collecting multiple end-to-end transmission paths, user agent sender needs to evaluate the quality of the part of a relay path from the user agent sender itself to the first-hop relay server (the first relay server on a relay path) for each relay path. Combined with path QoS information carried by OpenPath path allocation response message, user agent sender then calculates end-to-end transmission qualities of all relay paths, which are used as the basis of subsequent path selection and load distribution.

During the lifecycle of multipath transport session, user agent sender usually needs to dynamically evaluate QoS condition of all paths including default path and relay paths using MPTP and

corresponding measurement mechanism. Dynamic path quality information can be used to optimize load distribution process and achieve a better transmission service quality.

User agents should establish a multipath session before using multiple paths to transport data flow. It can be set up in many possible ways e.g., during session establishment, or at anytime during the session. To reduce session startup time, the user agent sender can start transporting data via the default path and then perform path connectivity checks for relay paths. If there are one or multiple available relay paths, the use agent sender updates the single-path session to a multipath session.

4.4 Relay service control protocol

Different from multipath transport scenario based on multi-homed hosts, in the multipath transport scenario based on application-level relay, the generation and normal operations of relay paths depend on the relay service system which includes relay controller and relay server.

Relay controller provides control functions of relay service, and relay sever provides executive functions of relay service. Relay server needs to report the status to the relay controller, and relay controller needs to control the behaviors of relay servers. In addition, in order to provide relay path service to user agents or out-of-band signaling server, relay controller needs to provide relay service access interfaces. Therefore, this document defines a relay service control protocol named OpenPath, which provides interfaces among relay controller, relay server and user agents or out-of-band signaling server.

The definition of OpenPath protocol will promote the progress of standardization of the application-level multipath relay service. As long as OpenPath protocol and MPTP are followed, third-party relay servers implemented and deployed by any organization and individual can interact with relay controller and provide relay service to user agents. This document defines OpenPath protocol in detail.

4.5 Multipath transport protocol suite and profile

Existence of multiple end-to-end transmission paths between participating nodes is only the basis of enabling multipath transmission. User agents need to use a kind of multipath transport protocol to exchange data via multiple transmission path in a session. The design of this multipath transport protocol needs to focus on considering the following factors: 1) express multipath transport control mechanism fully, such as subflow partition and

recombination mechanism, multipath routing, etc. 2) meet various transport requirements of different overlay applications, such as real-time transport requirement, reliable transport requirement, etc. 3) meet transport requirements of relay server. As relay server only supports efficient UDP forwarding, multipath transport protocol is required to be a UDP-based application-layer protocol. 4) minimize the overhead of multipath transport control. The ratio of transport control messages should be as small as possible. Some existing multipath transmission control protocols, including MPTCP [7], SCTP [8], can not meet the aforementioned requirements. In addition, multiple multipath transport protocols need to be designed to meet different transport requirement of overlay applications.

In this document, multipath transport protocol (MPTP) is designed to be a protocol suite which consists of one MPTP profile and multiple application-specific MPTPs. The MPTP profile is aimed to provide application-level multipath routing mechanism and each application-specific MPTP is aimed to meet the transmission requirements of a specific category of upper applications. All application-specific MPTPs MUST follow MPTP profile defined in this document. Relay servers need to support MPTP profile, and user agents need to support MPTP profile and one or multiple application-specific MPTPs.

MPTP is a UDP-based application-layer transport protocol. The protocol stack architecture of MPTP is shown in figure 2.

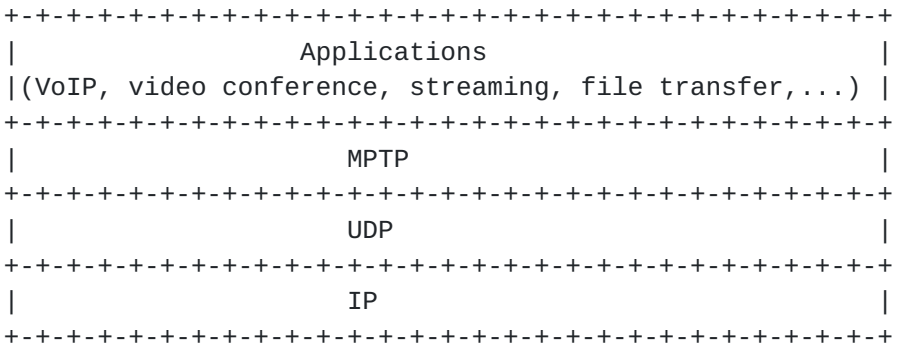


Figure 2. The protocol stack architecture of MPTP

In network protocol stack, the transport layer provides end-to-end communication services for applications. Reliable transport protocols, such as TCP and SCTP, are not particularly suitable for real-time applications such as streaming media and real-time multiplayer games. And they are complex and can be a problem for relay servers which provide relay service for huge numbers of user agents. In contrast, UDP only provides checksums for data integrity, and multiplexing for different applications. It delegates other functions to the above application programs. Therefore it typically

gives higher throughput and shorter latency. Based on these considerations, MPTP uses UDP as underlying transport protocol. Each UDP datagram carries one MPTP packet. On the one hand, this design decision simplifies the behaviors and enhances service capabilities of relay servers. In other words, relay server works in a stateless manner and provides the UDP-based relay service. On the other hand, all upper applications, that need either timely delivery or reliable delivery, can use the transport service provided by MPTP. The delivery requirements of upper application need to be meet by MPTP profile and application-specific MPTPs.

In addition to carrying payload data passed from upper application programs through multiple paths, MPTP also need to provide path control functions including keeping path alive and monitoring the quality of data delivery on each path.

5. Usage Scenarios

The multipath transport system framework based on application-level relay can provide many application scenarios including point-to-point, many-to-one and one-to-many communications. These applications could be delay sensitive or reliability sensitive, such as real-time communication, parallel streaming media, file transfer or file sharing.

Figure 3 illustrates a point-to-point multipath session. There are three paths between source and destination, including the default path, one relay path via one relay server and another relay path via two relay servers. User agent sender can choose a data partitioning method according to its particular requirements. Then, each flow is assigned to a path. The user agent receiver reassembles the received flows using a resequencing buffer to retrieve the reconstructed flow which is delivered to the above application.

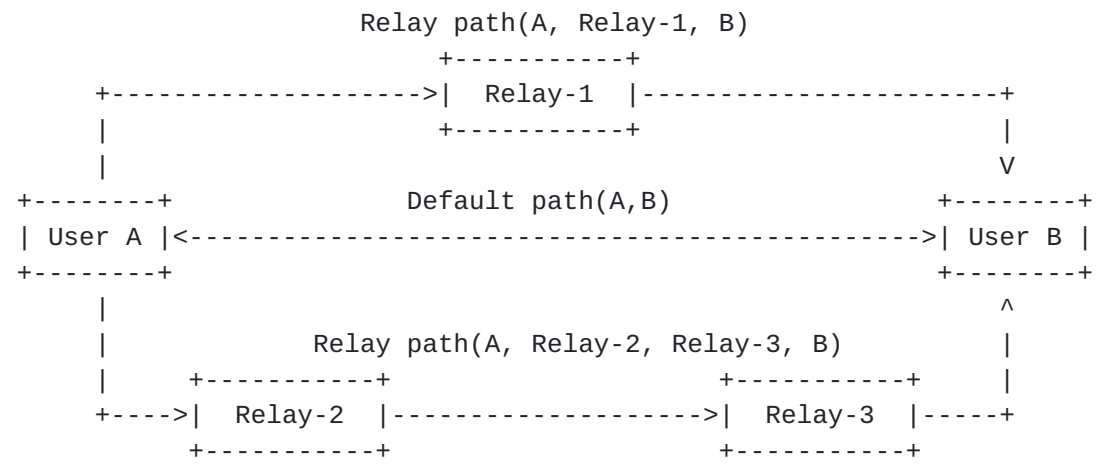


Figure 3. A point-to-point multipath session

A wide range of applications require data transmission from geographically distributed sources to one destination. For instance, large volume data of high definition movies are stored at multiple geographically distributed locations. The audiences need to retrieve and integrate data from several locations. This usage scenario can be completed by a many-to-one multipath session, which is depicted in Figure 4. In the figure, the user agent receiver streams different portions of a video from three servers concurrently. The path between the server and the user agent receiver may go through one or more relay servers.

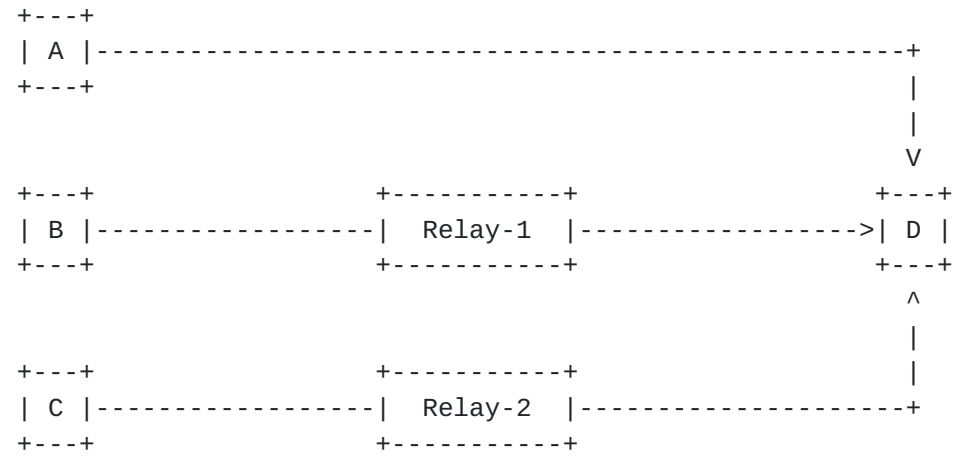


Figure 4. A many-to-one multipath session

Many video applications are typically characterized by a wide range of connection qualities and receiving devices which are with different capabilities ranging from cell phones with small screens and restricted processing power to high-end PC with high-definition display. These systems are usually adopting layered coding. Layered

coding enables the encoding of a high-quality video bit stream containing one or more subset bit streams that can themselves be decoded independently. This usage scenario can be completed by a one-to-many multipath session, which is depicted in figure 5. A video source is encoded into multiple streams, each of which is transported by a source tree for video multicast.

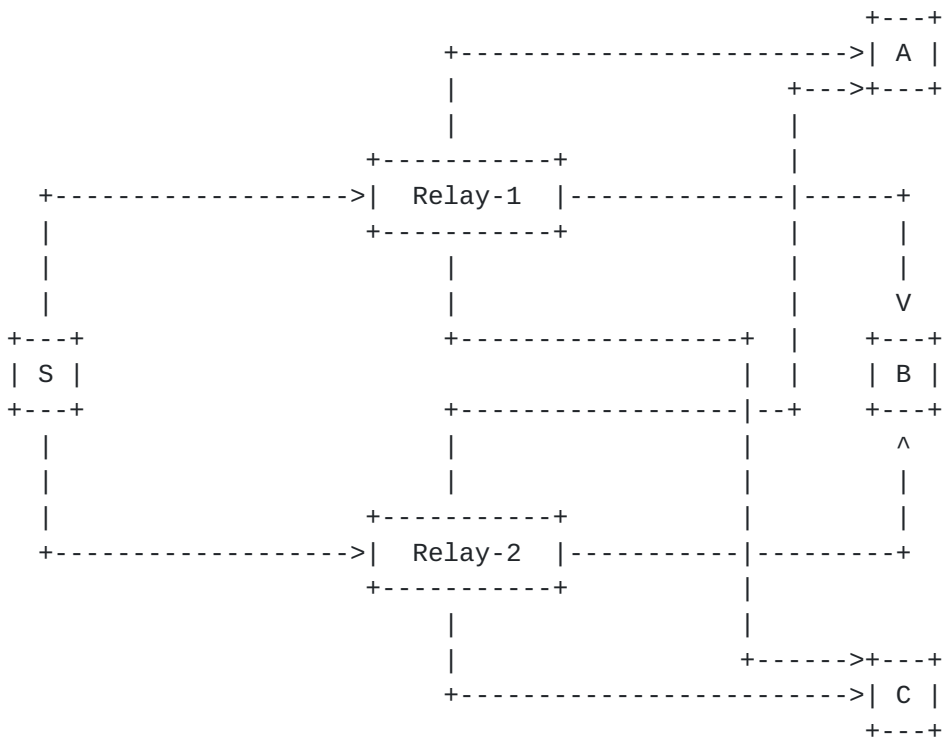


Figure 5. A one-to-many multipath session

5.1 Usage Scenario in SIP system

Figure 6 depicts a kind of usage scenario in which SIP is used as a separate signaling to advertise relay paths information.

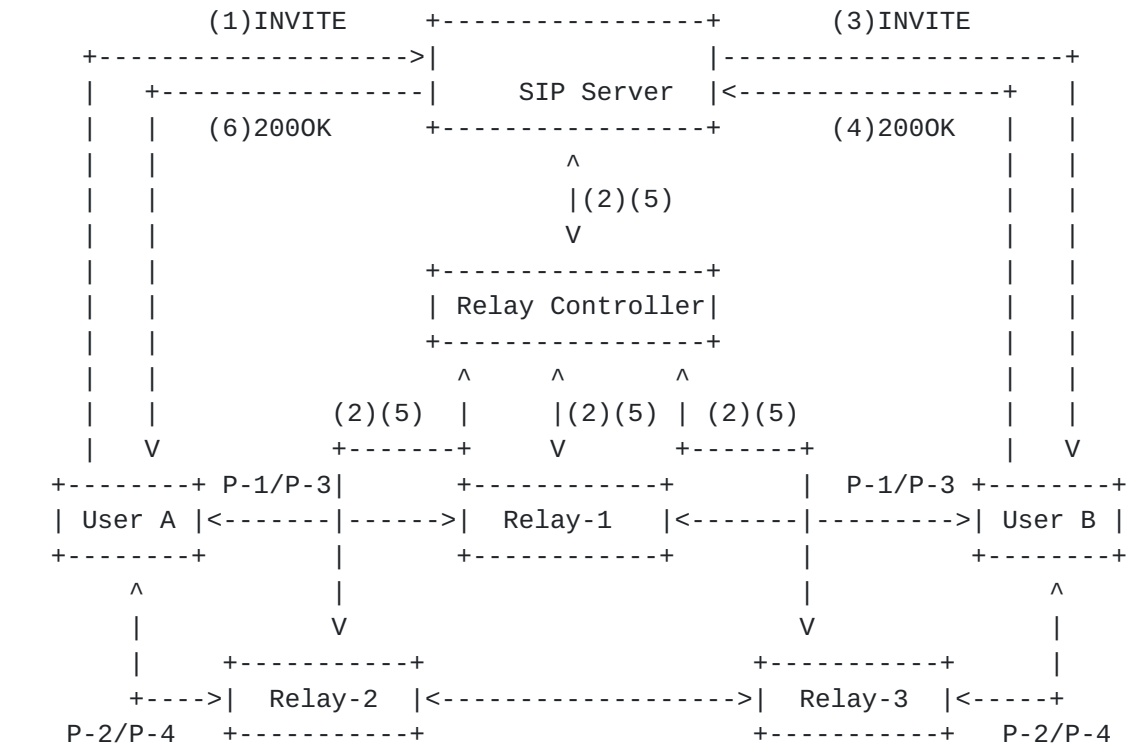


Figure 6. Usage scenario for multipath transport system framework in SIP System

(1) User A sends an INVITE to the SIP server that serves her domain.

(2) SIP server extracts the current addressable locations of the caller and the callee, and the media information including media type s and listed media codecs. Then SIP server sends an Openpath path allocation request, ALLOCATE_PATH, carrying the above information to the relay controller and requests it to assign the appropriate relay paths for the media flow from user B to user A. In this example, the relay controller selects two relay paths for the media flow from user B to user A. They are (B, Relay-1, A) and (B, Relay-3, Relay-2, A), named P-1 and P-2 respectively. And each relay path is associated with a globally unique path identifier, named PID-1 and PID-2 respectively. The relay controller sends each of the three selected relay server an ADD_PATH request message of OpenPath protocol. ADD_PATH request includes the corresponding path identifier and next-hop transport address of the receiving relay server. For Relay-1, the path identifier is PID-1 and the next-hop transport address is the current addressable location of user A. For Relay-2, the path identifier is PID-2 and the next-hop transport address is the current addressable location of user A. For Relay-3, the path identifier is PID-2 and the next-hop transport address is Relay-2's IP address and relay service port.

(3) SIP server inserts the path identifiers and the next-hop transport addresses of user B into SDP body of INVITE and forward the modified INVITE to user B. The next-hop transport addresses of user B for P-1 and P-2 are separately the rely address of Relay-1 and Relay-3.

(4) User B answers the call and sends back a 200 OK response.

(5) In the same way, SIP server obtains the allocated relay paths for the media flow from user A to user B from the relay controller. In this example, the relay controller assigns the same relay paths with opposite direction for the media flow from user A to user B based on symmetry principle. They are (A, Relay-1, B) and (A, Relay-2, Relay-3, B), named P-3 and P-4 respectively, associated with the path identifiers of PID-3 and PID-4. The relay controller sends an ADD_PATH request message to each of the three selected relay servers. For Relay-1, the path identifier is PID-3 and the next-hop transport address is the current addressable location of user B. For Relay-2, the path identifier is PID-4 and the next-hop transport address is Relay-3's IP address and relay service port. For Relay-3, the path identifier is PID-4 and the next-hop transport address is the current addressable location of user B.

(6) SIP server inserts the path identifiers and the next-hop transport addresses of user A into SDP body of 200 OK response and forwards it to user A. The next-hop transport addresses of user A for P-3 and P-4 are separately the rely address of Relay-1 and Relay-2.

Through the signaling process above, user A and user B separately obtain three candidate paths including the default path and two relay paths as the sending peer of the corresponding media flow. After connectivity check, user A and user B can take advantage of multiple paths to transport the media flow.

6. User Agent Behavior

Given multiple paths, user agent needs to provide essential support for multipath transport, including session and path management, flow partitioning and scheduling, subflow recombination, QoS feedback for each subflow.

6.1 Multipath session management

In general, a session needs to be established between two participants before transmitting data. At the same way, a multipath session needs to be established before transporting data on multiple paths. The multipath session setup uses a way of out-of-band (e.g., SDP in SIP or RTSP). The capability exchange and relay path

advertisement should be done during the signaling process. A multipath session may be setup from the beginning, or may be upgraded from a normal single path session.

6.2 Path management

The user agent sender obtains the default path from the out-of-band signaling for the session setup. In SIP/SDP case, the default path is determined by the c= and m= lines in SDP. It may be the direct network path between the sender and the receiver, or may traverse a third-party node, such as a TURN server or a dedicated relay server. For the latter, IP address and port of the third-party node is in the c= and m= lines in SDP. The path identifier of the default path is set to zero.

As stated in Section 5, the user agent sender can use two alternative ways to collect candidate relay paths. One way is that the user agent sender obtains candidate relay paths from the out-of-band signaling for the session setup. The signaling server, which out-of-band signaling passes through during session setup, requests the relay controller to allocate relay paths for the session to be established using OpenPath protocol. And then the signaling server inserts the information of the allocated relay paths into corresponding signaling messages to inform the participating user agents. Another way is that the user agent sender obtains candidate paths through direct interaction with the relay controller using OpenPath Protocol.

The user agent sender needs to perform path probes to determine if the path is available and if so, obtains the transmission quality of the path at the same time. After obtaining a new candidate path, the user agent sender first performs path probe process, if the path is available, marks it available and puts it into the available path list ordered based on a decreasing priority level; if the path is not available, marks it unavailable and puts it into the unavailable path list. The user agent sender will periodically perform path probes for all the paths in available path list to actively detect path failure and perceive the dynamic changes to the path. If the probe process for a particular path fails, the path will be marked unavailable and removed from the available path list to the unavailable path list. The user agent sender should also perform path probes for the paths in unavailable path list in a longer cycle. If the probe process for a particular path successes, the path will be marked available and removed from the unavailable path list to the available path list.

If no data is received on a relay path within a given time, relay servers will withdraw the corresponding resources allocated for this path. Therefore, all the relay paths should be kept alive actively by the user agent sender. To meet this requirement, the user agent

sender SHOULD send MPTP keep-alive packets periodically for both active paths and non-active paths.

Using the information in the subflow MPTP reports, a user agent sender can monitor delivery quality of active paths. If failure (e.g., errors, unreachability, and congestion) happens in an active path, the user agent sender may perform flow repartitioning and spread the payload across other active paths, or may select a new path from the available path list to replace the failure path.

6.3 Flow partitioning and scheduling

This document does not limit the usage of multiple paths. User agent sender may concurrently use multiple paths to obtain higher throughput, or may send all traffic on a specified path while all other available paths are used only rarely to enhance resilience (e.g., for retransmission, for error-repair, or for redundancy packets). User agent sender MUST only use the default path and the relay paths in available path list as the active path. How to select multiple paths among all available paths and how many paths to use concurrently are outside the scope of this document.

If multiple paths are used concurrently, the original data stream should be divided into several substreams. Flow partitioning methods are outside the scope of this document. Application-specific MPTP documents may introduce some flow partitioning methods for specific applications. A simple flow partitioning scheme is to assign packets to multiple subflows using the round-robin algorithm. Specifically, the original data stream is first divided into blocks of equal-sized temporal or spacial length. Then, a subflow is assembled by picking blocks periodically from the original blocks in an increasing order. This method is simple and can be applied to all of the upper applications. However, the data are treated equally and not distinguished based on their different importance in terms of the whole data flow. And great correlations exist among subflows which are sent along paths with different transport capacity.

User agent sender should associate a subflow with an active path based on a scheduling strategy. A scheduling policy should jointly consider various factors including the estimated path bandwidth information, the path reception statistics (e.g. RTT, loss-rate, delay etc.), the relative importance of subflow data and so on. Due to the changes in path characteristics, user agent sender should be able to change its scheduling strategy during an ongoing session to fully explore the potential of multipath transport.

6.4 Subflow packaging

In a multipath session, user agent sender formats data flow into MPTP data packets following MPTP profile defined in this document and the corresponding application-specific MPTP defined in one application-specific MPTP documents. Then the user agent sender dispatches MPTP data packets along corresponding relay paths.

The common header of MPTP packets includes two key fields: path identifier and subflow-specific sequence number (SSSN). The path identifier of a relay path, which is globally unique, is generated by the relay controller. The path identifier of the default path is fixed to zero. The path identifier is used to identify a specific path by the user agent sender and relays. Meanwhile, the path identifier is also used to identify a subflow by the user agent receiver.

Subflow-specific sequence number is the sequence number of a MPTP data packet in a specific subflow. It is used to help calculate the quality of data delivery such as fractional losses, jitter, RTT, etc, for each path. The initial subflow sequence number is randomly generated when the subflow is first established in the multipath session.

6.5 Subflow recombination

The user agent receiver recombines the original data flow according to MPTP data packets received from multiple paths. The user agent receiver first restores the order of each subflow using path identifier and subflow sequence number in MPTP data packet headers. Subsequent work is different according to specific applications. Subsequent work SHOULD be defined in detail in application-specific MPTP documents.

6.6 Subflow reporting

In order to monitor the quality of path delivery and to meet specific requirements of some kind of applications, user agents SHOULD generate MPTP reports for per subflow to provide subflow information. The user agent sender generates MPTP Sender Reports (SR) for each unique subflow and sends them along the same path as the subflow MPTP data packets. As the relay path is unidirectional and the default path is bidirectional, the user agent receiver generates MPTP Receiver Reports (RR) for each unique subflow and sends them along the default path.

Although subflow MPTP SRs and RRs are not sent along the same path, they still can be used to measure the quality of path delivery. For example, the calculated round-trip propagations to the user agent receiver along different paths using subflow MPTP SRs and RRs still

can correctly represent relative size of transmission delays of different paths.

When user agent generates MPTP report packets is outside the scope of this document. It SHOULD be defined in detail in application-specific MPTP documents. In general, timely MPTP reports are necessary for multipath transport environments. MPTP reporting interval should be frequent enough for user agents to quickly adapt to path fluctuation and transmission errors. Meanwhile the traffic overhead introduced by MPTP reporting SHOULD also be taken into account when designing an application-specific MPTP for some specific applications.

In addition, what MPTP report packets contain and what to do when receiving an MPTP report packet are related to the requirements of specific application programs. They are outside the scope of this document and SHOULD also be defined in detail in application-specific MPTP documents.

7. Relay Behavior

Relay server performs MPTP packets lookups and forwarding based on a local Path-Table. Relays are managed by the relay controller over connections using a protocol referred to as OpenPath protocol.

7.1 Connection Management and Registrations

Relay server communicates with the relay controller over a connection which may be encrypted using TLS or run directly over TCP. Relay server initiates a standard TLS or TCP connection to the relay controller. Relay server can get the IP address of the relay controller in a number of ways, such as manual configuration, DNS domain name queries and so on.

When a connection is established, relay server completes the registration process by exchanging HELLO messages. The version field in HELLO request is set to the highest OpenPath protocol version supported by the relay server. Relay controller needs to check the included OpenPath protocol version in HELLO request. If the version is not supported, relay controller responds to the HELLO with a failure response.

The relay server identifier field is set to zero in the initial HELLO request. This indicates that it is the first time for this relay server to register with the relay controller. The relay controller needs to assign a unique identifier for this relay server and insert the assigned relay server identifier in the corresponding HELLO response message. Subsequent HELLO request messages may carry the relay server identifier directly.

The HELLO request contains the IP address and port of the relay server for the relay service.

If a failure response to the HELLO message is received, relay server then terminates the connection. Otherwise, the connection proceeds and the relay server may start relay service.

During the lifetime of the connection, ECHO requests should be sent periodically from either relay server or relay controller, and the request receiver must return an ECHO reply.

After connection establishment, relay server may receive FEATURES requests from relay controller. It must respond with a FEATURES reply that specifies its capabilities.

During the lifetime of the connection, relay controller may periodically collect statistics from a relay server by STATISTICS requests. The relay server must respond with a STATISTICS reply that specifies its current statistics.

7.2 Path-Table Management

The Path-Table contains a set of path entries each of which corresponds to an associated relay path. Each path entry consists of match fields, result fields and counters (see table 1).

Match fields are used to match against MPTP packets. Match fields only consist of path identifier in this document.

Path Identifier: a 32 bit unsigned integer uniquely identifying the associated relay path.

Result fields include next-hop transport address, idle timeout and hard timeout. Next-hop transport address is to determine where the matched packets are forwarded. Idle timeout and hard timeout are used for relay to actively clear those expired path entries.

Next-hop Transport Address Type: corresponds to the type of the next-hop transport address. Namely:

1: IPv4 address
2: IPv6 address

Next-hop Transport IP Address: the address to which the relay server forwards the matched packets.

Next-hop Transport Port: the port number to which the relay server forwards the matched packets.

Counters are maintained for each path entry and updated for matching MPTP packets.

Received packets: the amount of packets the path has matched.

Received bytes: the amount of bytes the path has matched.

Duration: the amount of time the path has been installed in the relay server.

Table 1: Fields in a path entry.

Fields	Bits
Match fields	
Path Identifier	32
Result fields	
Next-hop transport address type	8
Next-hop transport IP address	For an IPv6 address, this is 128;for an IPv4 address, this is 32.
Next-hop transport port	16
Idle timeout	16
Hard timeout	16
Counters	
Received packets	64
Received bytes	64
Duration (seconds)	32
Duration (nanoseconds	32

Path-Table of relay server is managed by relay controller through Path-Table modification messages. For ADD_PATH requests, relay server must first check if any path entry with the same path identifier has existed in the Path-Table. If an overlap conflict exists between an

existing path entry and the ADD_PATH request, relay server must refuse the addition and respond with a failure response. For valid ADD_PATH requests, relay server must insert a new path entry in the Path-Table and respond to the relay controller with a success response. The counters of received packets and received bytes in the new inserted entry are set to zero.

For DELETE_PATH requests, if a matching entry exists in the Path-Table,it must be removed, and a success response is returned to the relay controller. For DELETE_PATH requests, if no path entry matches, no path entry modification occurs, and a failure response is returned to the relay controller.

For UPDATE_PATH requests, if a matching entry exists in the Path-Table, the result fields of this entry is updated with the value from the request, and counter fields are left unchanged. For UPDATE_PATH requests, if no path entry matches, no path entry modification occurs, and a failure response is returned to the relay controller.

7.3 Path Validity Management

Each path entry has an idle timeout and a hard timeout associated with it. These two fields control how quickly a path entry expires. When a path entry is inserted by an ADD_PATH request or modified by a UPDATE_PATH request, its idle timeout and hard timeout are set with the values from the message.

If either value is non-zero, relay server must note the arrival time of MPTP packet on the associated path, as it may need to evict the path entry later. A non-zero hard timeout field causes the path entry to be removed after the given number of seconds, regardless of how many ackets it has matched. A non-zero idle timeout field causes the path entry to be removed when it has matched no packets in the given number of seconds. Using these two fields, relay server can actively clear those expired path entries. In addition, relay controller may actively remove path entries by sending DELETE_PATH messages.

If a relay server actively removes a path entry, it must send a NOTIFY message to relay controller. The NOTIFY message contains a complete description of the path entry including the reason for removal, the path entry duration and statistics at the time of removal.

7.4 Relay Service Management

After successful registration, relay server may send a START message to relay controller to indicate that it has enough capacity to

provide relay service.

In the case that a relay server is overloaded, or under other some situations, the relay server may send a STOP message to relay controller to indicate that it will no longer receive new relay service requests (i.e. ADD_PATH messages) for a while. During this period, the relay server still provides relay service for those existing relay paths. And ECHO messages still need to be sent periodically between the relay server and the relay controller.

When a relay server recovers from an overloaded state, it may send a START message to relay controller to indicate that it has additional capacities to provide new relay services.

When a relay server wants to stop relay service permanently, it should actively send a BYE message to relay controller before terminating the connection. In this way, relay controller can be ready in time to do some remedial measures. For instance, relay controller may assign new relay paths for the affected media flow.

7.5 MPTP Packet Processing

The main function of a relay server is to provide relay service to the associated subflows. All subflows associated with a relay server share a common relay port of this relay server.

After receiving a MPTP packet, relay server extracts the path identifier from the received MPTP packet and does matching in the Path-Table. If the received MPTP packet does not match a path entry in the Path-Table, relay server has nothing to do but to drop the packet. If the received packet matches a path entry in the Path-Table, relay server forwards it to the associated next-hop transport address in the matched path entry. Meanwhile, relay server updates the associated statistical counters in the matched path entry.

7.6 Topology Discovery and Performance Measurement

In this document, the term topology refers to the topology that connects all the relay servers in a MPTS-AR. The connection between relay servers can be based on the manual configuration or other mechanism. For instance, during the registration process, relay controller may select several registered relay servers randomly or according to certain strategy, and insert their information in the corresponding HELLO success response message. ECHO messages can be used to accomplish the same function.

After obtaining other relay servers information by manual

configuration or being provisioned by relay controller, relay server performs topology discovery process. Topology discovery process is mainly to decide whether there is an overlay link between two relay servers. An alternative way may be based on the following rules. If two relay servers are located in the same autonomous domain, there will be an overlay link connecting the two relay servers. If there is an interdomain link between two domains, there is an overlay link which connects the two relay servers from the two domains. The global overlay topology is formed based on the topology observation.

Relay server should report the result of topology discovery process to relay controller via ECHO messages periodically.

An overlay link is usually composed of multiple physical links. Besides overlay traffic, other nonoverlay traffic would be using the same physical links. In addition, relay functions provided by relay server work in application layer, that is, on top of transport layer. Thus, relay server cannot control or manage the IP-layer resource. Relay server can only rely on measurement mechanism to obtain the performance of overlay links associated to it. With performance measurement, relay server can track dynamicly overlay link capacity. Overlay link capacity can be expressed in terms of the quality metrics such as loss rate, delay, available bandwidth and so on. The concrete measurement methods that can be adopted are outside the scope of this document.

Relay server should report dynamic capacity of overlay links associated with it to relay controller via ECHO messages periodically.

8. Relay Controller Behavior

Relay controller is responsible for managing relay servers and selecting one or multiple relay paths for a data flow.

8.1 Relay Server Management

Relay controller manages all relay servers in its region, and maintains registration and status information of relay servers such as relay capacity, availability and work load. For each relay server, relay controller collects its capabilities information by FEATURES messages, and periodically collects its statistics information by STATISTICS messages.

8.2 Topology Maintenance

Relay controller manages the topology of a network of relay servers in its region. Topology informations include if an overlay link

exists between any two relay servers and if so, how about the quality of the overlay link.

Relay controller can obtain topology informations from relay servers in its regin via ECHO messages.

8.3 Relay Path Allocation

During establishing a multipath session, the user agent sender or the signaling server which out-of-band signaling pass through during session setup, requests relay controller to allocate relay paths by ALLOCATE_PATH messages. The ALLOCATE_PATH request carries the related information of the data flow which is going to establish, such as location information of participants, transport requirements of the data flow and so on. Relay controller selects one or multiple relay paths according to the information carried in ALLOCATE_PATH request and the status information of the registered relay servers. For each selected relay path, the relay controller sends an ADD_PATH request to each relay server on the selected relay path. The ADD_PATH request carries the path identifier, next-hop transport address of the receiving relay server, etc. A relay path is assigned successfully only when each relay server on this relay path replies with an ADD_PATH success response. After successful path allocation, relay controller replies an ALLOCATE_PATH success response to inform the user agent sender or the signaling server about the allocated relay path information including the path identifier, user agent sender's next-hop transport address, etc.

9. OpenPath Protocol

9.1 Protocol Overview

OpenPath is based on a request/response transaction model. Each transaction consists of a request and a response. A response uses the same transaction id as is in the associated request to facilitate pairing. The transaction id is a 32-bit identifier generated by the request sender.

OpenPath messages are guaranteed delivery over a connection-oriented channel. All integer fields are carried in network octet order, that is, most significant byte first. Octets designated as padding have the value zero.

All OpenPath messages begin with an OpenPath common header. OpenPath messages MAY contain a message body. The structure and interpretation of a body depends on the message type.

OpenPath message types fall into four classes: relay-to-controller,

controller-to-relay, user agent-to-controller and symmetric.

Relay-to-controller messages are initiated by relay server and used to manage the channel connection and update relay controller of changes to the relay server. Controller-to-relay messages are initiated by relay controller and used to manage the Path-Table or inspect the state of relay servers. User agent-to-controller messages are initiated by user agent or out-of-band signaling server, and used for relay path allocation. Symmetric messages are initiated by either relay controller or relay server and used to keep alive the channel connection and topology maintenance.

9.1.1 Relay-to-Controller

Relay-to-controller message is initiated by a relay server and requires a response message from relay controller.

HELLO: Relay server registers with relay controller by sending a HELLO request. Relay controller must respond with a HELLO response that indicates the outcome of registration. This is commonly performed upon establishment of the OpenPath connection channel.

The relay server identifier field is set to zero in the initial HELLO request. Relay controller must assign a unique identifier for this relay server and insert the assigned relay server identifier in the corresponding HELLO response. Subsequent HELLO requests may carry the assigned relay server identifier directly.

The HELLO request contains a message body. The body contains the IP address and port of relay server for the relay service.

The HELLO success response may contains a message body. The body contains the information of one or more other registered relay servers. The information of a relay server here include the IP address for relay service and the relay server identifier.

START: Relay server starts relay service by sending a START request. Relay controller must respond with a START response that indicates the outcome of relay service startup.

STOP: Relay server may stop relay service temporarily by sending a STOP request. For instance, when a relay server is overloaded, it may want to refuse accepting any new relay service requests for a while. Relay controller must respond with a STOP response that indicates the outcome of relay service pause. During relay service pause, this relay server still provides relay service for those existing relay paths. This relay server may restart relay service by sending a START request after a while.

BYE: Relay server may terminate the relay service permanently by sending a BYE request, such as before terminating the OpenPath connection channel or exiting. Meanwhile, this relay server ceases relay service for all existing relay paths. If this relay server wants to start relay service again, it must first perform registration with relay controller.

NOTIFY: When a relay server actively removes a path entry, it may notify relay controller by sending a NOTIFY request. For instance, in order to save the resource, relay server actively removes those path entries which lack activity.

9.1.2 Controller-to-Relay

Controller-to-relay message is initiated by a relay controller and require a response message from relay server.

FEATURES: Relay controller may request the capabilities of a relay server by sending a FEATURES request. This relay server must respond with a FEATURES response that specifies its capabilities. This is commonly performed after successful registration of this relay server.

STATISTICS: Relay controller may periodically collect statistics from relay servers by sending STATISTICS requests. Relay server must respond with a STATISTICS response that specifies its current statistics.

ADD_PATH: When relay controller assigns relay paths for a data flow, it must send an ADD_PATH request to each relay server on the assigned relay path. The relay server must respond with an ADD_PATH response that specifies the outcome of adding a new path entry. A relay path is assigned successfully only when each relay server replies with an ADD_PATH success response.

UPDATE_PATH: Relay controller may want to modify an existing path entry in the Path-Table of a relay server by sending a UPDATE_PATH request. For instance, a data flow may be "put on hold" and data transmission may be ceased for a while. In this case, relay controller may update the idle timeout and hard timeout of the corresponding path entries of all the relay servers on the affected relay paths to a longer time. Relay server must respond with an UPDATE_PATH response that specifies the outcome of updating an existing path entry.

DELETE_PATH: Relay controller may delete an existing path entry in the Path-Table of a relay server by sending a DELETE_PATH request, such as when a data flow ends normally. Relay server must respond

with a DELETE_PATH response that specifies the outcome of deleting an existing path entry.

9.1.3 User agent-to-Controller

User agent-to-controller message is initiated by a user agent or an out-of-band signaling server, and requires a response message from relay controller.

ALLOCATE_PATH: During establishing a multipath session, user agent sender may request relay controller to allocate candidate relay paths by exchanging ALLOCATE_PATH messages directly with relay controller. Or the signaling server requests relay controller to allocate candidate relay paths for user agents, and inserts the information of the allocated relay paths into corresponding signaling messages to inform user agents. Relay controller must respond with an ALLOCATE_PATH response that specifies the outcome of path allocation and the information of the assigned relay path if exists.

RELEASE_PATH: Before tearing down a multipath session, user agent sender may request relay controller to release the relay paths assigned for the corresponding media flow by exchanging RELEASE_PATH messages directly with relay controller. RELEASE_PATH request message carries the information of one or more relay path identifiers. Or the signaling server requests relay controller to release the assigned relay paths on behalf of the user agent sender. Relay controller must respond with an RELEASE_PATH response that specifies the outcome of deleting relay paths.

9.1.4 Symmetric

Symmetric message is sent in either direction between relay server and relay controller.

ECHO: ECHO requests MUST be sent periodically from either relay controller or relay server. ECHO messages can be used to measure the latency of the connection between relay controller and relay server, as well as verify the peer's liveness. The receiver of an ECHO request must respond with an ECHO response.

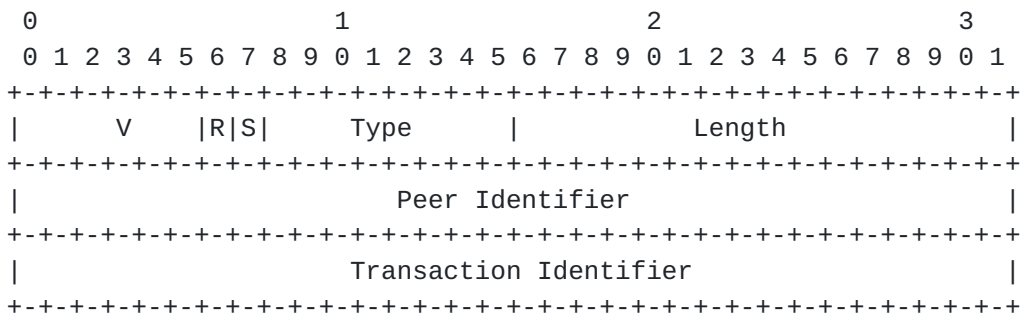
Relay server can insert the information of topology and overlay link capacity in ECHO requests or ECHO responses.

9.2 Common Structures

This section describes several common structures used by multiple messages.

9.2.1 OpenPath Common Header

All OpenPath messages begin with an OpenPath common header which is defined below:



Version (V): 6 bits
This field identifies the version of OpenPath protocol version being used. The version defined by this document is one (1).

R: 1 bit
If the R bit is set, the message is an OpenPath request; otherwise, the message is an OpenPath response.

S: 1 bit
When the message is a request, this bit is reserved. When the message is a response, if the S bit is set, the message is a success response; otherwise, the message is a failure response.

Type: 8 bits
This field identifies the type of the OpenPath messages. This document defines 13 OpenPath message types:

Type Value	Type Name	Sending Direction
1	HELLO	RTP relay -> Controller
2	BYE	RTP relay -> Controller
3	ECHO	RTP relay -> Controller Or Controller -> Relay
4	START	RTP relay -> Controller
5	STOP	RTP relay -> Controller
6	NOTIFY	RTP relay -> Controller

7	FEATURES	Controller -> Relay
8	STATISTICS	Controller -> Relay
9	ADD_PATH	Controller -> Relay
10	DETELE_PATH	Controller -> Relay
11	UPDATE_PATH	Controller -> Relay
12	ALLOCATE_PATH	User Agent -> Controller
13	RELEASE_PATH	User Agent -> Controller

Length: 16 bits

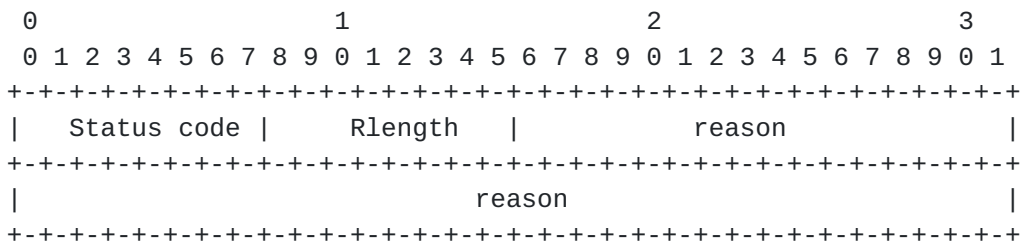
The length field indicates the total length of the message in 32-bit words including the header and any padding.

Peer Identifier: 32 bits
This field is a 32-bit identifier that corresponds to a globally unique relay server, user agent or out-of-band signaling server. It is generated by relay controller during registration process of a relay server, user agent or out-of-band signaling server. This identifier remains unchanged during the entire lifecycle of the OpenPath connection with relay controller. For OpenPath messages exchanged between relay controller and relay server, this field corresponds to a relay server identifier. For OpenPath messages exchanged between relay controller and user agent, this field corresponds to a user agent identifier. For OpenPath messages exchanged between relay controller and out-of-band signaling server, this field corresponds to a signaling server identifier. For the last two cases, it is uniformly called user agent identifier.

Transaction Identifier: 32 bits
This field identifies the transaction id associated with this message. It is randomly generated by the request sender and discarded when the associated response message is received. The transaction identifier of a response message must always match the requests that prompted them.

9.2.2 Common Body of OpenPath Failure Responses

OpenPath failure responses of all message types MAY contain an optional body after the OpenPath common header. If present, the body conforms to a common structure defined below.



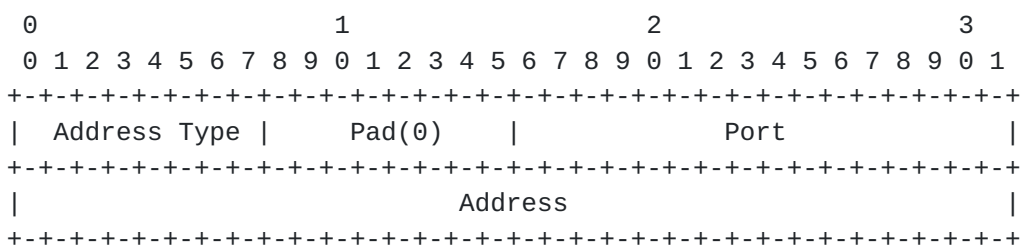
Status code: 8 bits
This field is a numeric result code that indicates the outcome of a request processing.

Rlength: 8 bits
This field is the length of the reason phrase in 16-bit word length.

Reason: variable size
This field is a short textual description of the status code.

9.2.3 Transport Address Structure

Relay server needs to advertise relay controller about its transport address for relay service. Relay controller also needs to tell relay server about the transport address of its next-hop node for each associated path. A transport address is defined as follow.



Address Type: 8 bits
This field is the type of transport address. Namely:
1: IPv4 address
2: IPv6 address

Port: 16 bits
This field is the port number part of transport address.

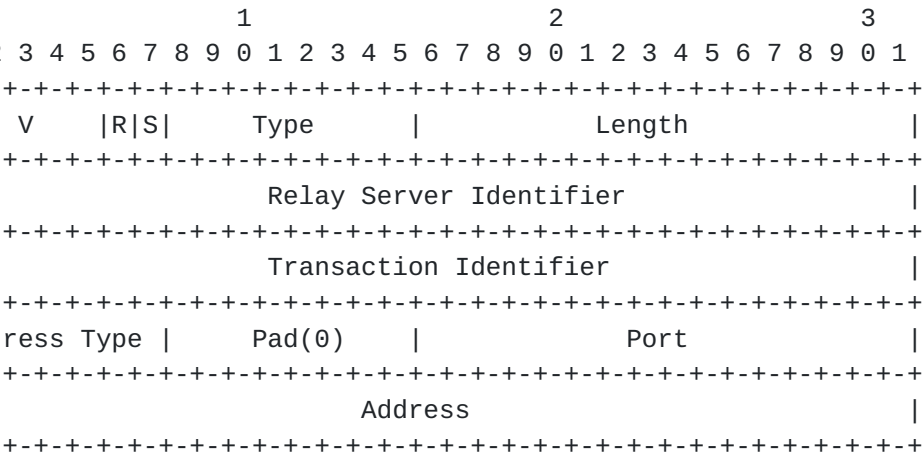
Address: 4 octets (IPv4), 16 octets (IPv6)
The IP address part of transport address.

9.3 Message Format of OpenPath Request and Success Response

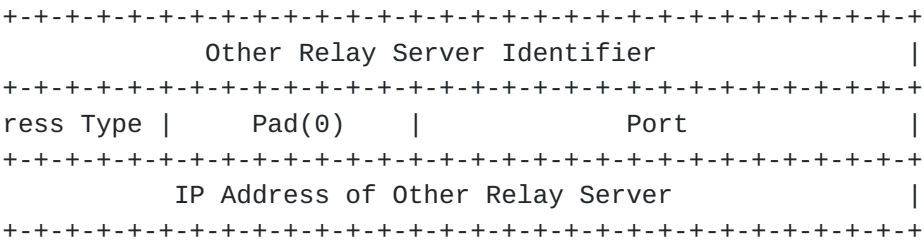
9.3.1 HELLO

A HELLO request contains a body beyond an OpenPath common header.

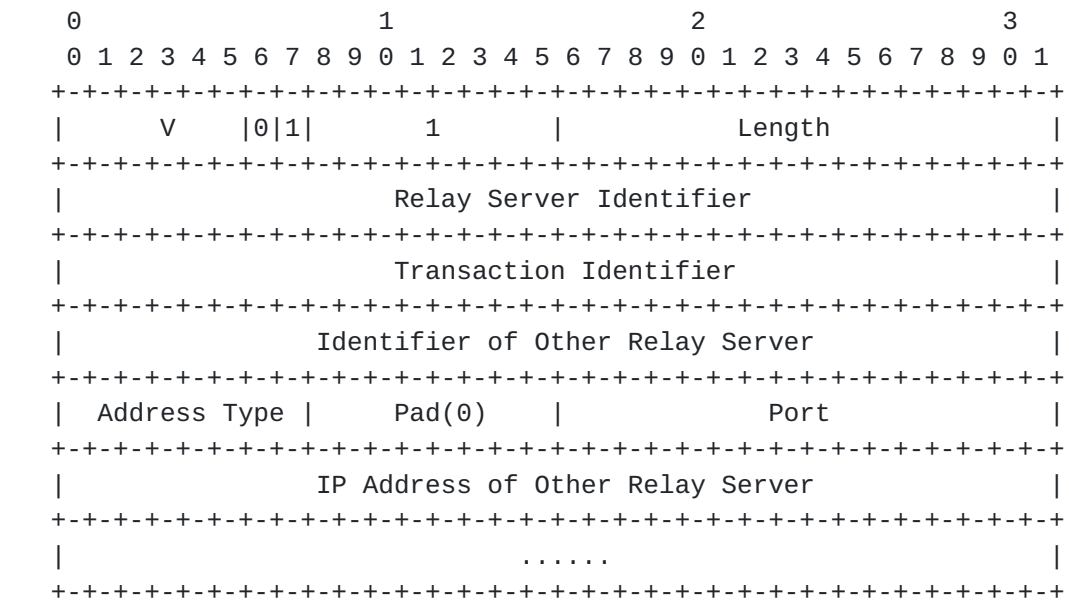
The body contains the transport address of the relay server to provide relay service.



A HELLO success response may contains a message body. The body contains information of one or more other registered relay servers. The information of a relay server here include the address for relay service and the relay server identifier.



An example of A HELLO success response is:

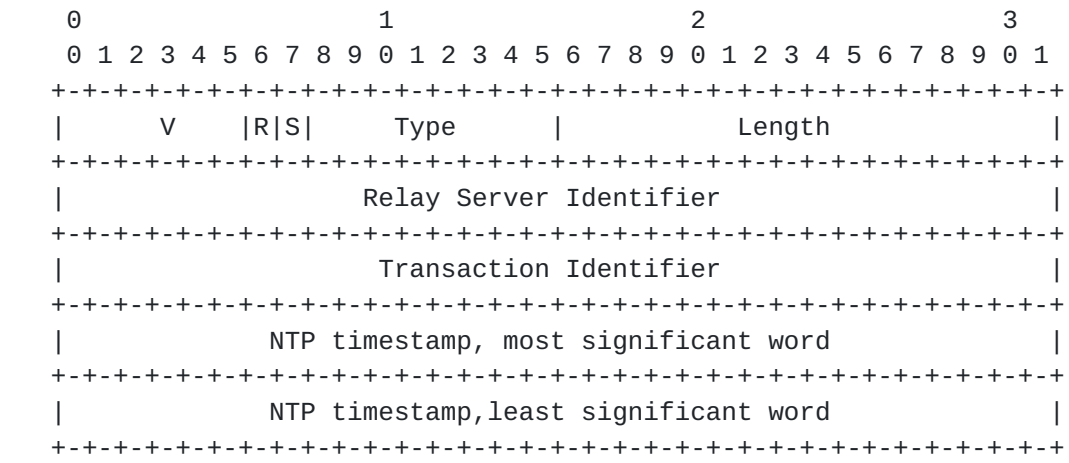


9.3.2 START/STOP/BYE

START/STOP/BYE requests and their success responses have no body; that is, they only contain an OpenPath common header.

9.3.3 ECHO

An ECHO request consists of an OpenPath common header and an optional body. If the body is absent, an ECHO transaction is used to simply verify liveness between relay controller and relay server. If the body is present, it may contain a timestamp field to check latency between relay controller and relay server. Example:

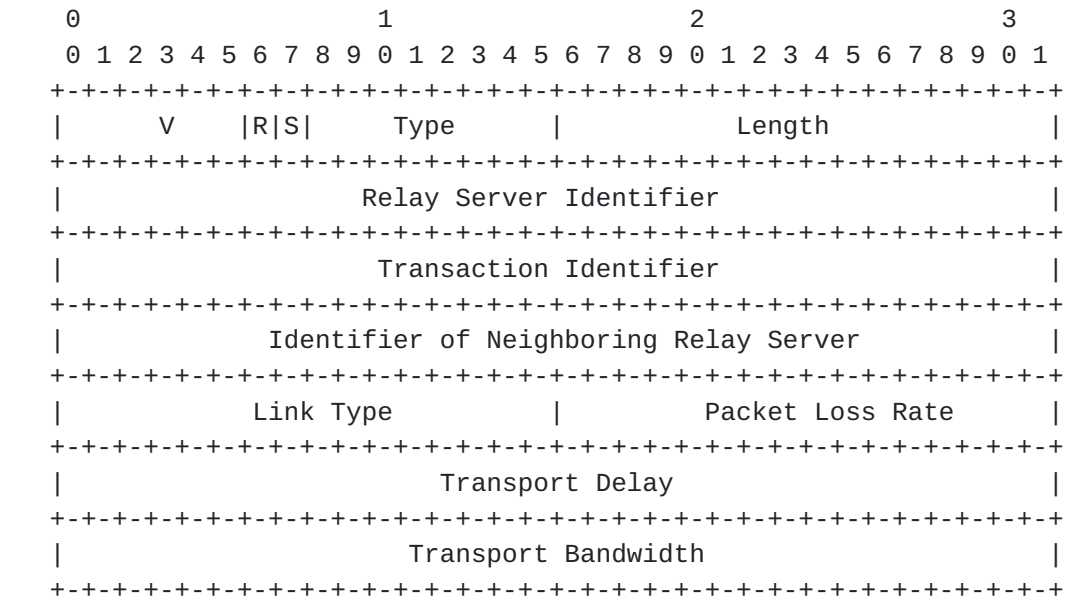


NTP timestamp: Using the timestamp format of the Network Time Protocol (NTP), which is in seconds relative to 0h UTC on 1 January 1900 [2]. The full resolution NTP timestamp is a 64-bit unsigned

fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits.

If the body of an ECHO request only contain a timestamp field, its response has the same message format as the associated ECHO request.

An ECHO message (request message or response message) generated by a relay server may contain the information of one or more overlay link capacities in its body.



Identifier of Neighboring Relay Server:
the identifier of the relay server which is connected to this ECHO message sender by a overlay link.

Link Type: 16 bits
This field identifies the type of the overlay link between this ECHO message sender and the relay server which is identified by the identifier of neighboring relay server field. This document defines 2 overlay link types:

Type Value	Description
1	The two relay servers which are associated with the overlay link are located in the same autonomous domain.
2	The two relay servers which are associated with the overlay link are from two different autonomous domains. And there is an interdomain link between these two domains.

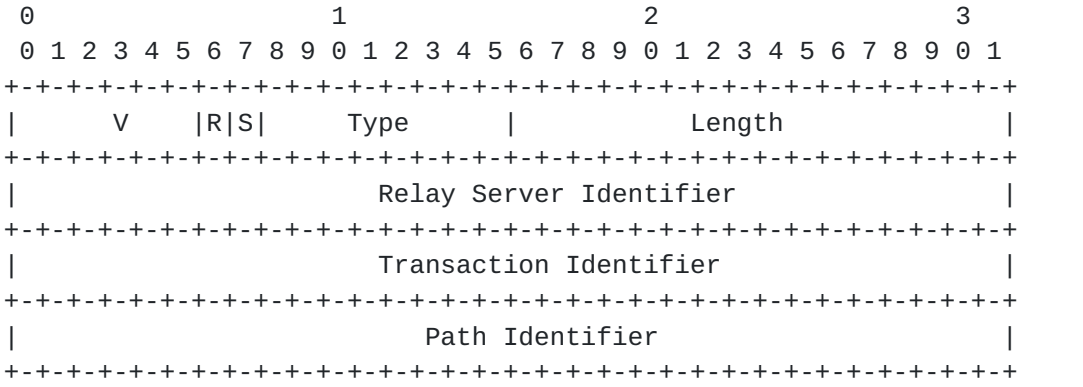
Packet Loss Rate: 16 bits
This field indicates the estimated packet loss rate of the overlay link since the last ECHO message which contains the information report of the same overlay link. This field is expressed in units of per thousand.

Transport Delay: 32 bits
This field indicates the estimated unidirectional transport delay of the overlay link from this ECHO message sender to the relay server which is identified by the identifier of neighboring relay server field. This field is expressed in units of 1/65536 seconds.

Transport Bandwidth: 32 bits
This field indicates the estimated transport bandwidth of the overlay link. This field is expressed in units of kilo-bits per second (kpbs).

9.3.4 NOTIFY/DELETE_PATH

NOTIFY/DELETE_PATH requests contain a body beyond an OpenPath common header. The body only contains a path identifier field.



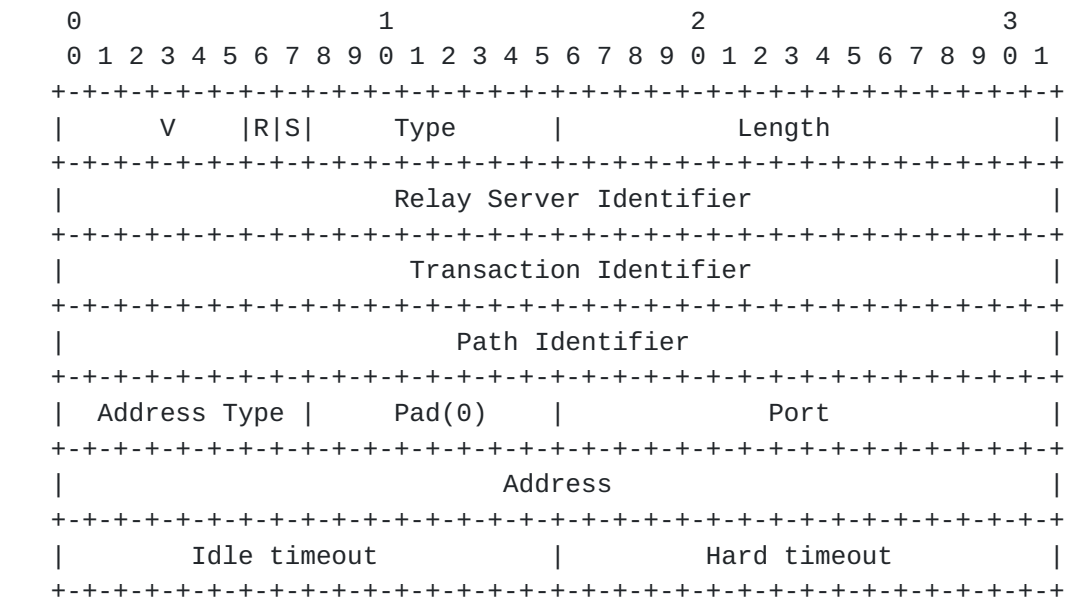
Path Identifier: 32 bits

This field is a 32-bit identifier that corresponds to a globally unique path. It is generated by relay controller when assigning relay paths for a data flow.

NOTIFY/DELETE_PATH success responses only contain an OpenPath common header.

9.3.5 ADD_PATH/UPDATE_PATH

ADD_PATH/UPDATE_PATH requests contain a body beyond an OpenPath common header. The body contains match fields and result fields of a path entry. The match fields contain a single path identifier field. The result fields contain next-hop transport address and idle/hard timeout fields.



Idle timeout: 16 bits
This field is the number of seconds after which the path will timeout if no traffic.

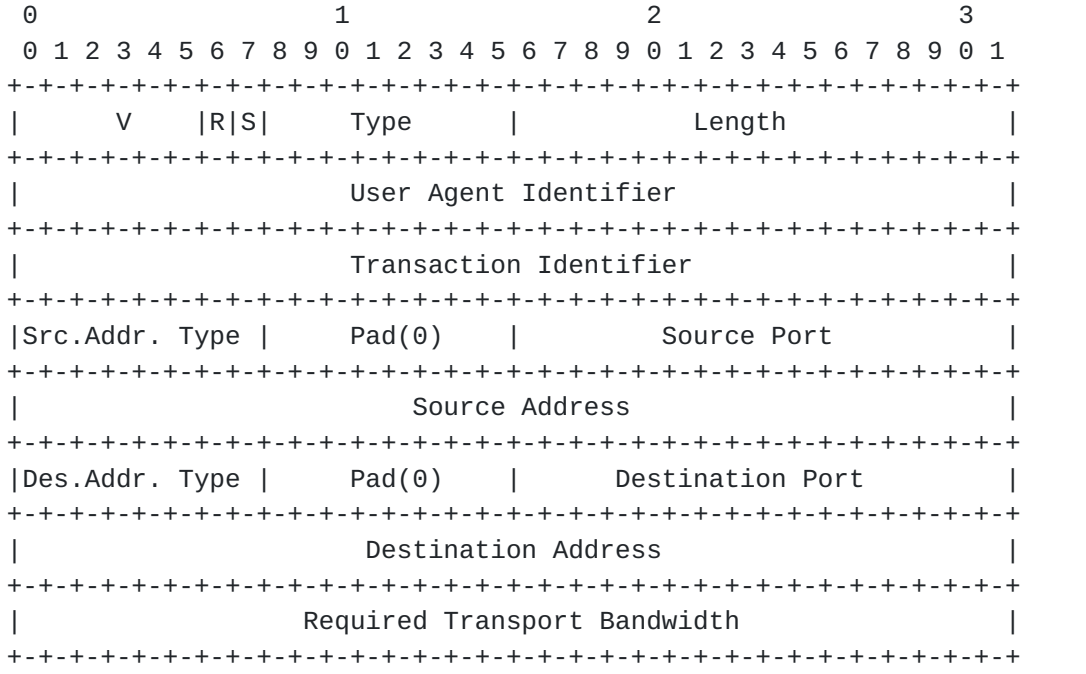
Hard timeout: 16 bits
This field is the number of seconds after which the path must expire regardless of whether or not packets go through the path.

ADD_PATH/UPDATE_PATH success responses only contain an OpenPath common header.

9.3.6 ALLOCATE_PATH

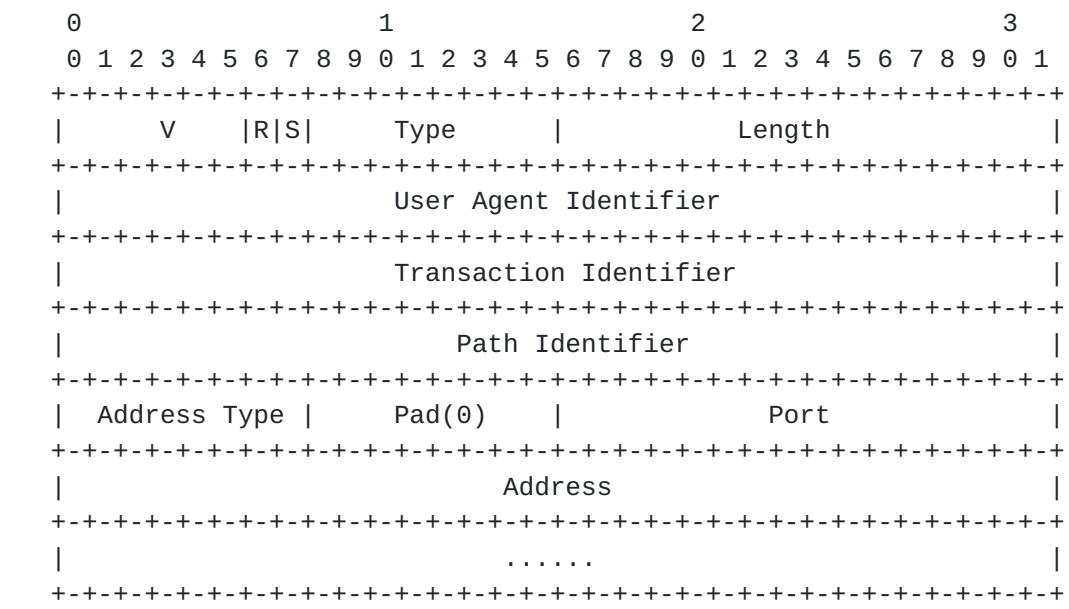
ALLOCATE_PATH requests contain a body beyond an OpenPath common header. The body contains information of the media flow which is

going to be established. Information of a media flow include source and destination addresses and the transport requirements.



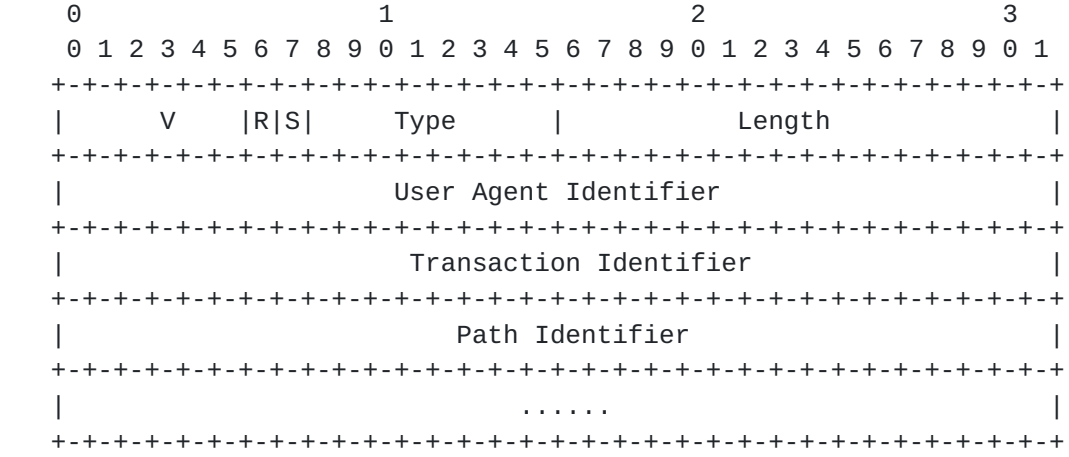
Required Transport Bandwidth:
This field indicates the required transport bandwidth of the media flow which is going to be established. This field is expressed in units of kilo-bits per second (kpbs).

An ALLOCATE_PATH success response contains a body beyond an OpenPath common header. The body contains information of one or more allocated relay path, including the path identifier and user agent sender's next-hop transport address.



9.3.7 RELEASE_PATH

RELEASE_PATH requests contain a body beyond an OpenPath common header. The body contains one or more path identifier fields. Each path identifier corresponds to one relay path which is being requested to be released.



RELEASE_PATH success responses only contain an OpenPath common header.

9.3.8 FEATURES

A FEATURES request only contains an OpenPath common header.

A FEATURES success response contains a body beyond an OpenPath common header.

(to be continued)

9.3.9 STATISTICS

A STATISTICS request only contains an OpenPath common header.

A STATISTICS success response contains a body beyond an OpenPath common header.

(to be continued)

10. MPTP Profile

10.1 Overview

As stated in Section 5, MPTP obtain a simple, unreliable datagram service from the underlying UDP. MPTP SHOULD meet various delivery requirements of upper applications. As stated in the introduction, in order to be extensible and suitable for various applications, MPTP is designed to be a protocol suite which consists of one MPTP profile and multiple application-specific MPTPs. The MPTP profile only defines common rules of multipath transport based on application-level relay for various upper applications. It is aimed to provide application-level multipath routing mechanism. Other transmission requirements of upper applications, such as timely or reliable delivery, SHOULD be met in corresponding application-specific MPTP documents.

In addition to carrying payload data passed from upper application programs through multiple paths, MPTP also need to provide path control functions including keeping path alive and monitoring the quality of data delivery on each path. Therefore, MPTP packets are divided into two types: MPTP data packets and MPTP control packets. MPTP control packets include MPTP keep-alive packets and MPTP report packets.

User agent sender formats the payload data passed from upper application programs into MPTP data packets which are sent along the associated path.

User agent sender SHOULD send MPTP keep-alive packets periodically for each path including both active path and non-active path.

To monitor the transport quality of a path, user agents generate MPTP report packets for each subflow. Due to the content of MPTP report packets depends on the delivery requirements of upper application programs, this document does not give concrete content and processing of the MPTP report packets, which should be described in

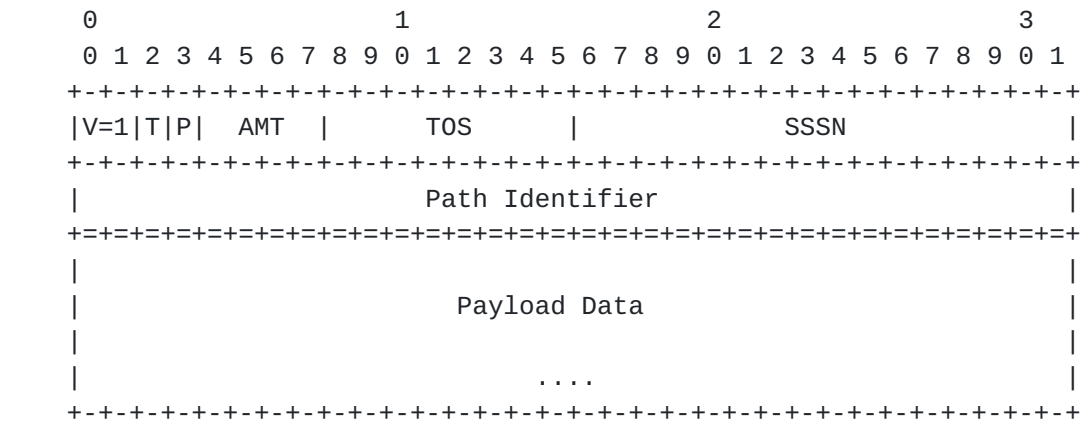
application-specific MPTP documents. Here, we only outline the delivery methods of MPTP report packets. The user agent sender generates MPTP sender reports for each subflow and sends them along the same path as the subflow MPTP data packets. The user agent receiver responds with MPTP receiver reports which are sent along the default path.

10.2 MPTP Fixed Header Fields

All MPTP packets have a fixed eight octet-length header. The first four octets except for the first four bits correspond to different fields for MPTP data packets and MPTP control packets.

10.2.1 Fixed Header Fields of MPTP Data Packet

Fixed header of MPTP data packet is defined below:



The fields have the following meaning:

version (V): 2 bits
This field identifies the version of MPTP. The version defined by this document is one (1).

MPTP packet type (T): 1 bit
This field identifies the type of a MPTP packet. If the MPTP packet type bit is set, this packet is a MPTP data packet; otherwise, this packet is a MPTP control packet.

padding (P): 1 bit
If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored, including itself. Padding may be needed by some encryption algorithms with fixed block sizes.

Application-specific MPTP type (AMT): 4 bits
This field identifies the type of application-specific MPTP that this packet format conforms to. This document defines two application-specific MPTP types: AMT = 1, indicates that this MPTP packet conforms to RTP-based multimedia application-specific MPTP; AMT = 2, indicates that this MPTP packet conforms to reliable application-specific MPTP.

type of service (TOS): 8 bits
The last four bits of the type of service is reserved. The first four bits of the type of service, similar to TOS field in internet packet header, is specified along the abstract parameters precedence, delay, throughput, and reliability. These abstract parameters embody the delivery requirements of upper application programs. The values of these abstract parameters should be specified in application-specific MPTP documents.

Precedence: An independent measure of the importance of the payload data.

Delay: Prompt delivery is important for the payload data with this indication.

Throughput: High data rate is important for the payload data with this indication.

Reliability: A higher level of effort to ensure delivery is important for the payload data with this indication.

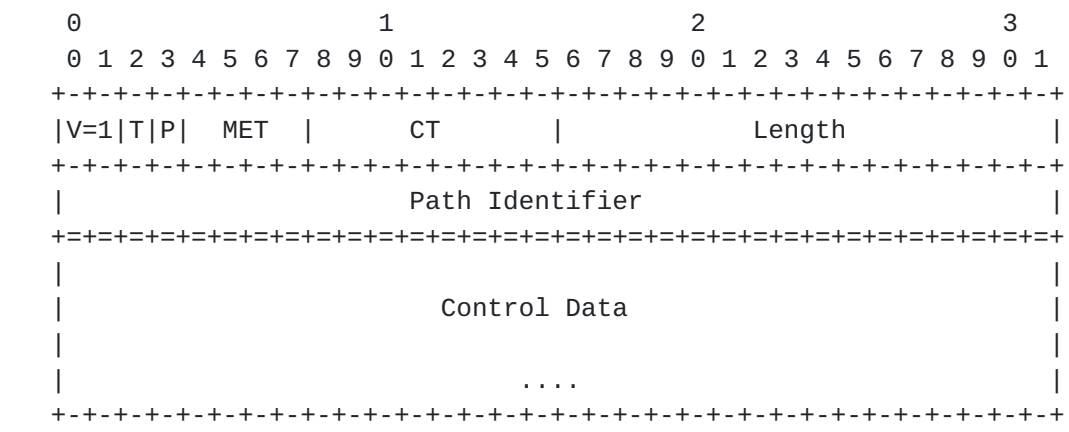
Subflow-Specific Sequence Number (SSSN):
This field is the sequence of the MPTP data packet in the subflow. Each subflow has its own strictly monotonically increasing sequence number space.

Path Identifier:
This field is the identifier of the path that the MPTP packet is associated with. For a relay path, the path identifier is globally unique; for the default path, the path identifier is fixed to zero.

Payload Data: variable size
The content of payload data should be described in application-specific MPTP documents.

10.2.2 Fixed Header Fields of MPTP Control Packet

Fixed header of MPTP control packet is defined below:



The fields of V, T, P, MET and Path Identifier have the same meaning as those fields in MPTP data packet. Other fields have the following meaning:

MPTP control packet type (CT): 8 bits

This field identifies the type of MPTP control packet. Namely:

MPTP control packet type (CT)	Use
1	Subflow Sender Report, for transmission statistics from the user agent sender
2	Subflow Receiver Report, for reception statistics from the user agent receive
3	Keep Alive, for keeping a path alive

Length: 8 bits
This field is the length of the encapsulated control data after the MPTP fixed header in byte length.

Control Data: variable size
For MPTP keep-alive packet, control data may be empty.
For MPTP report packet, the content of control data should be described in application-specific MPTP documents.

11. SDP Considerations

The advertisement of MPTP capability and relay paths MUST be

performed by out-of-band signaling, for example, as part of a SIP offer/answer exchange using SDP. This section defines dedicated extensions in SDP. SDP syntax and procedures are available that can be re-used or easily adapted to provide the necessary capabilities.

11.1 Signaling MPTP Capability in SDP

A communication participant, who follows the framework of multipath transport system based on application-level relay, MUST use SDP to indicate that it supports and wants to use this framework. The mptp-relay attribute defined here will be used to indicate the support for this framework. The mptp-relay attribute is a media level parameter. The syntax of the mptp-relay attribute is defined using the following Augmented BNF, as defined in [RFC5234].

mptp-relay-attrib = "a=" "mptp-relay" ":" mptp-name CRLF
The mptp-name field indicates an application-specific MPTP. In an application-specific MPTP document, the value of mptp-name field MUST be given for that application-specific MPTP.

When SDP is used following the offer/answer mode [RFC3264], the "mptp-relay" attribute indicates the desire to transport media flow on multiple paths. If the offerer supports and wishes to use MPTP, it MUST include a media-level "a=mptp-relay" attribution in the initial SDP offer. If the initial SDP offer contains "a=mptp-relay" attribution and if the answerer supports and wishes to use MPTP, it MUST include this attribute in the SDP answer.

When SDP is used in a declarative manner, the "mptp-relay" attribute indicates that the message sender can send or receive media data over multiple paths. The message receiver SHOULD be capable to stream media to multiple paths or be prepared to receive media from multiple paths.

11.2 Relay Path Advertisement in SDP

The information of candidate relay paths MUST be advertised to the user agent sender in the out-of-band signaling. The relay-path attribute is extended to advertise candidate relay paths in SDP. The syntax of the relay-path attribute is defined using the following Augmented BNF, as defined in [RFC5234]. The definitions of DIGIT, SP and CRLF are according to RFC4234.

relay-path-attrib = "a=" relay-path-label ":" counter SP path-id SP transport-address CRLF
relay-path-label = "relay-path"
counter = 1*DIGIT
path-id = 32token-char

```
transport-address = addrtype ":" unicast-address ":" port
  ; addrtype from RFC4566, typically "IP4" | "IP6"
  ; port from RFC4566
unicast-address = IP4-address / IP6-address
  ; IP4-address from RFC4566
  ; IP6-address from RFC4566
token-char = %x21 / %x23-27 / %x2A-2B / %x2D-2E / %x30-39 / %x41-5A /
%x5E-7E
```

The path identifier and the next-hop transport address of the user agent sender for each candidate relay path is encapsulated in a relay-path attribute.

The <counter> parameter indicates the priority of the associated relay path and it is a monotonically increasing positive integer starting at 1. Number 1 is the highest priority. The counter must be reset for each media line. The relay-path attributes are ordered based on a decreasing priority level.

The <path-id> parameter indicates the path identifier of the associated relay path.

The <transport-address> is the next-hop transport address of the user agent sender for associated candidate relay path. The <addrtype> is the address type. This document only defines IP4 and IP6 for IPv4 addresses and IPv6 addresses respectively.

Example:

```
a=relay-path:1 1a3b6c9d0e2f6g1qazxsw23edcvfr45t IP4:192.0.3.2:10000
```

12. IANA Considerations

The following contact information shall be used for all registrations in this document:

Contact: Weimin Lei
mailto:leiweimin@ise.neu.edu.cn
tel:+86-24-8368-3048

Note to the RFC-Editor: When publishing this document as an RFC, please replace "RFC XXXX" with the actual RFC number of this document and delete this sentence.

12.1 SDP Attributes

In the registry for SDP parameters, the attributes named "mptp-relay" and "relay-path" have been registered as follows:

SDP Attribute ("att-field"):

Attribute Name:	mptp-relay
Long form:	Multipath transport system framework based on application-level relay
Type of name:	att-field
Type of attribute:	Media level only
Subject to charset:	No
Purpose:	This attribute is used to indicate support for multipath transport system framework based on application-level relay.
Reference:	See this document
Values:	See this document.

SDP Attribute ("att-field"):

Attribute Name:	relay-path
Long form:	Relay Path of multipath transport system framework based on application-level relay
Type of name:	att-field
Type of attribute:	Media level only
Subject to charset:	No
Purpose:	This attribute is used to describe the information of a relay path including the path identifier and the next-hop transport address of the user agent sender.
Reference:	See this document
Values:	See this document.

13. Security Considerations

TBD

14. References

14.1 Normative References

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

[2] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

14.2 Informative References

[3] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[5] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[6] Schulzrinne, H., Rao, A., Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC2326, April 1998.

[7] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC6182, March 2011.

[8] Stewart, R., "Stream Control Transmission Protocol", RFC4960, September 2007.

Authors' Addresses

Weimin Lei
Northeastern University
Institute of Communication and Information System
College of Information Science and Engineering
Shenyang, China, 110819
P. R. China

Phone: +86-24-8368-3048
Email: leiweimin@ise.neu.edu.cn

Wei Zhang
Northeastern University
Institute of Communication and Information System
College of Information Science and Engineering
Shenyang, China, 110819
P. R. China

Email: zhangwei1@ise.neu.edu.cn

Shaowei Liu
Northeastern University
Institute of Communication and Information System
College of Information Science and Engineering
Shenyang, China, 110819
P. R. China

Email: liushaowei.ise.neu@gmail.com
liu_nongfu@163.com