Internet Engineering Task Force
Internet Draft
Intended status: experimental
Expires: January 2011
A.Singh
University of Bremen
M. Scharf
Alcatel-Lucent Bell Labs
August 6, 2010

PayLoad Multi-connection Transport using Multiple Addresses
draft-singh-mptcp-plmt-00.txt

Abstract

The single path transport provided by the Transmission Control Protocol (TCP) can be extended to a multipath transport session for multi-homed end hosts by coupling several TCP connections over multiple interfaces of the end hosts. Payload Multi-connection Transport (PLMT) is a multipath protocol variant that encodes all the control/signaling information in the payload of TCP connections and therefore requires no additional TCP options. PLMT allows for the simultaneous use of the multiple connections over potentially disjoint paths while being mostly backward compatible to single path transport of TCP. PLMT operates as an additional protocol layer between the network stack and the application layer. This document describes PLMT as an example for a multipath mechanism that could possibly be realized entirely in the user-space of an operating system.

Table of Contents

1. Introduction

The objective of a multipath transport mechanism is to allow the simultaneous use of multiple connections over multiple paths. A multipath transport mechanism is expected to be beneficial since it enhances the network resource utilization and since it provides resilience to node failures in the network [5].

One key mechanism that aims to provide multipath transport is Multipath TCP (MPTCP). MPTCP enables multipath transport by utilizing multiple addresses of the end host to establish multiple paths (subflows) for a TCP connection [6]. MPTCP extends the standard Transmission Control Protocol (TCP) [2] to add the multipath capability and uses several new TCP options to encode control/signaling information.

Another multipath transport solution, MCTCP [9] uses the new TCP options only during connection setup to transport signaling information. Afterwards the additional signaling information is sent together with the application data in the payload using a type-length-value (TLV) framing format.

This document presents the Payload Multi-connection Transport (PLMT) protocol design as a further alternative multipath transport mechanism. PLMT also uses a type-length-value (TLV) framing format to send application data and control/signaling information. However, in order to transmit control/signaling information; PLMT does not use new TCP options, unlike other multipath transport solutions. Instead, PLMT sets up a control connection to a well-known port for the signaling information exchange, and it uses payload encoding over standard TCP connections. The control connection can either be set up before starting the data transport, or afterwards. In either case, it is possible to implement the PLMT signaling without changing the network stack. Each of the multiple PLMT connections is a standard TCP connection that transports TLV encoded data segments and that are coupled together to the PLMT session.

Therefore, PLMT is easily deployable and extensible. PLMT is also transparent to applications and offers reliable transport similar to a standard TCP connection. PLMT is also mostly backward compatible to single path standard TCP. By design, PLMT robustly operates in environments with middleboxes that prevent the use of new TCP options. But the use of out-of-band signaling also comes at some cost concerning complexity, fall-back options, and security. However, as outlined in this document, PLMT is designed to minimize these risks and is rather robust. This document presents PLMT and discusses both the advantages and drawbacks of its design.

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 [1].

This document uses the terminology defined in [5][6], though some of the terms are re-defined.

Session: A connection over which an application can communicate between two hosts. For an application, there is a one-to-one mapping between a session and the socket. If a session includes only the initial connection, it is almost identical to a standard TCP connection.

PLMT Control Port: A port allocated to accept the PLMT control connections.

PLMT Layer: A protocol layer implementing the multi-connection capability of the PLMT. It can for instance be realized in the user space of an operating system.

Initial Connection: A TCP connection established by an application request. If both ends are PLMT capable, the first subflow uses this connection.

Additional Subflow Connection: A new TCP connection established for a subsequent subflow.

Control Connection: A TCP connection that is established to the PLMT Control Port. The IP addresses are identical to the Initial Connection.

PLMT Data Segment: The segmented application data with TLV header.

Active Opener: Refers to the TCP client for a Session with PLMT Layer.

Passive Opener: Refers to the TCP server for a Session with PLMT Layer.

Legacy End-host: Refers to a host without PLMT Layer.

Token: A 64-bit number that is unique on a host.

Signature: A long bit pattern that is used to identify PLMT messages inside TCP connections. The length is 16 byte (128 bit). It MUST be

selected in a way such that it is unlikely to occur in application protocols. Guidelines how to determine a Signature are explained in section 4.3.1. .

Session Sequence Number: The sequence number of a byte inside a byte stream of a session, determined by the PLMT Layer.

3. Design Considerations

This section gives a high-level overview of PLMT's design.

3.1. Goals

Important design assumptions and goals of the PLMT design are:

- No change of network stack: PLMT is designed to minimize the impact on the network stack implementation. The signaling can be completely implemented in the user-space of an operating system.

- Backward compatible: The PLMT should be backward compatible to standard TCP. A single connection PLMT should be exactly similar to the standard TCP connection. As long as only one connection exists, it is not necessary to use TLV framing on that connection.

- Co-existence with standard TCP connections: A PLMT capable end host must be able to differentiate between PLMT connections and regular TCP connections. This is crucial, since PLMT connections use TLV encoding.

- Multihomed and multiaddressed end hosts: PLMT assumes that for the establishment of multiple connections at least one of the end hosts must be multihomed and multiaddressed.

- Middlebox compatibility: PLMT should be compliant to the vast majority of middleboxes, such as NAPT middleboxes and firewalls. Therefore, PLMT should not rely on TCP extensions. PLMT should also allow a middlebox to identify that a host establishes PLMT connections, and prevent this.

o Transparency: PLMT should be transparent to the legacy application i.e., it should provide the same API and services (of the standard TCP) to the application.

3.2. Layered Representation

PLMT operates as an additional protocol layer (shim layer) between the application layer and the transport layer. It is designed to be transparent to both higher and lower layers and to be implemented in the user space. It can be used by legacy applications without any changes. Figure 1 illustrates this layering.

```
                                    +-------------------------------+
                                    |          Application          |
+-----------------------------+     +-------------------------------+
|         Application         |     |             PLMT              |
+-----------------------------+     +---------------+---------------+
|             TCP             |     |      TCP      |      TCP      |
+--------------+--------------+     +---------------+---------------+
|             IP              |     |      IP       |      IP       |
+--------------+--------------+     +---------------+---------------+
```

Figure 1 Comparison of Standard TCP and PLMT Protocol Stacks

3.3. Operation Summary

This section gives an outlook to the overall high-level operation of PLMT. Figure 2 depicts a simple scenario to illustrate the basic PLMT operation. A detailed PLMT protocol specification and operation description is provided in section 4.

o A legacy application, unaware of the presence of PLMT will initiate a standard TCP connection by opening a TCP socket for a Session. PLMT-aware applications MAY use a new application interface [8] to control the functioning of PLMT.

o The PLMT Layer then manages the connection establishment of Initial Connection, Control Connection and additional Subflow Connections.

o In order to enable PLMT, the Active Opener opens a PLMT control connection to a well-known port at the Passive Opener. The control connection is used to determine whether the remote end supports PLMT, and to exchange the necessary control information such as the Tokens. The Control Connection, as well as Subflow Connections, are established in the standard TCP way by the PLMT Layer.

- A node may either set up a Control Connection before or in parallel to the setting up of the Initial Connection (refer Figure 2). Alternatively, it may first use the Initial Connection and decide later to open the Control Connection. The latter case is discussed in section 4.3.3. . The control connection must be set up using the same IP source and destination addresses like the Initial Connection, and use the PLMT control port. If the setup of the Control Connection fails, PLMT will not be enabled and fall back to standard TCP.

- If the Passive Opener supports PLMT and TLV transport is successfully enabled, the Initial Connection will use a TLV framing for data transmission. Then, the Initial Connection is also termed first Subflow Connection. The setup of the TCP connections between two hosts A and B is illustrated in Figure 2. PLMT signals the use of TLV encoding by sending the Signature in the payload of the TCP byte stream. The Signature is a long bit pattern that is selected in such a way that it is unlikely to occur in a TCP connection not using PLMT. Furthermore, Tokens are used to verify that the Initial and Control Connection originate indeed at the same hosts. A detailed analysis of the security implications of PLMT and the resulting very small risk of false positives when detecting its connections are provided in section 6. .

- If multiple interfaces are present, PLMT can establish multiple Subflow Connections to allow data transport over multiple paths. Once TLV encoded data transport is activated, a Session level data sequence number is used for in-order delivery of the Data Segments over multiple Subflow Connections. The PLMT Layer manages the multiple interfaces and connections and delivers the packets over the different connections. At the receiver, the PLMT Layer reassembles the byte stream and transparently delivery them to the application.

- As the Subflow Connections are standard TCP connections, they are terminated as a regular TCP connection with the 4-way FIN handshake. The Session is terminated with the termination of the last subflow.

```
            End-host A                               End-host B
     ---------------------------             ---------------------------
      Address A1     Address A2               Address B1     Address B2
     ------------   ------------             ------------   ------------
           |               |                          |              |
           |     (Initial Connection setup)           |              |
           |---------------SYN----------------------->|              |
           |<------------SYN/ACK----------------------|              |
           |---------------ACK----------------------->|              |
           |               |                          |              |
           |     (Control Connection setup)           |              |
           |~~~~~~~~~~~~~~~SYN~~~~~~~~~~~~~~~~~~~~~~~>|              |
           |<~~~~~~~~~~~~SYN/ACK~~~~~~~~~~~~~~~~~~~~~~|              |
           |~~~~~~~~~~~~~~~ACK~~~~~~~~~~~~~~~~~~~~~~~>|              |
           |               |                          |              |
           | (Token exchange over Control Connection  |              |
           |       as detailed in Figure 3)           |              |
           |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>|              |
           |<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|              |
           |               |                          |              |
           |           Signature+Token                |              |
           |----------------------------------------->|              |
           |           Signature+Token                |              |
           |<-----------------------------------------|              |
           |               |                          |              |
           |     (TLV-encoded Data Segments           |              |
           |     over the Initial Connection)         |              |
           |----------------------------------------->|              |
           |<-----------------------------------------|              |
           |               |                          |              |
           |(Address Exchange over the Control or Initial Connection)|
           |               |                          |              |
           |       (Additional Subflow Connection setup (TCP))       |
           |==========================SYN===========================>|
           |<=======================SYN/ACK==========================|
           |==========================ACK===========================>|
           |               |                          |              |
           |        (Signature and TLV-encoded Data Segments         |
           |              over the Subflow Connection)               |
           |=======================================================>|
           |<=======================================================|
           |               |            |             |              |
```

Figure 2 PLMT Connections Establishment in case that the Control Connection
is set up in parallel to the Initial Connection

3.4. Compatibility

PLMT uses the Control Connection to detect whether a Passive Opener indeed supports its operation. If the setup of the Control Connection fails, it falls back to standard TCP transport, and does not use any additional PLMT signaling. PLMT is thus compatible with legacy TCP stacks and is able to detect them.

The PLMT Layer is transparent to applications, i. e., it is compatible with legacy applications unaware of PLMT.

The PLMT protocol does not require extensions of the TCP protocol and reuses the standard TCP mechanisms for the reliable, in-order operation of its connections. PLMT uses its own frame format based on the TLV encoding to send the application data and the control information. The use of TLV encoding is known from other TCP-based protocols such as TLS [3]. Therefore, PLMT should pass most middleboxes, in particular all middleboxes that would block TCP options. An exception is the case of middleboxes that parse the byte stream and block TLV content. In this case, PLMT transport may fail in certain cases, as discussed in section 5.3. and 5.4. .

The signaling and message transport of PLMT can be implemented on a host without changing the network stack, i. e., as a library in the user space. With a combination of scheduling and rate shaping mechanisms, the PLMT Layer can also try to emulate congestion control coupling algorithms such as [4]. In this case, it may be possible to implement PLMT entirely in the user space of a host.

3.5. Advantages and Drawbacks of PLMT

PLMT follows the principles outlined for a multipath transport solution based on TCP in [5]. PLMT uses the TCP payload to transport signaling messages and requires no new TCP options. Thus, PLMT brings along all advantages of the payload encoding mechanism (cf. [9]):

- o PLMT does not use any TCP option to setup its connections. Therefore, it might be possible to implement PLMT entirely in the user-space, which would significantly facilitate deployment of PLMT.

- o In addition, the signaling messages are not constrained with the limited size of the TCP options, and PLMT does not consume further option space in SYN segments.

- o PLMT does not modify TCP and is therefore compatible with many middleboxes, especially ones which do not allow unknown TCP options to get through, or ones that re-write the TCP options.

- o Middleboxes can very easily identify the setup of a PLMT Control Connection due to the use of a well-known port. If a middlebox on the path of the Initial Connection wants to prevent the use of multipath transport, it can simply block the connection setup to that port. Then, multipath transport will not be used for the corresponding connection.

PLMT is developed as an example for a multipath transport protocol that does not use any new TCP option, or other TCP extensions, and that is still backward compatible. Still, due to the use of payload encoding and an out-of-band control channel for the exchange of control information, a number of issues arise. The following text discusses these problems (some of which may exist for other multipath transport solutions as well) and possible solutions.

- o PLMT opens a Control Connection per PLMT Session, i. e., an additional TCP connection. If a host opens Control Connections for every short TCP-based transfer, this would result in a large number of additional connection setups, which would consume bandwidth, processing resources, and port numbers. The worst case is that a PLMT Control Connection is set up for every Initial Connection, but additional subflows are never established. Then, the number of TCP connection doubles without any performance benefit. As a remedy, PLMT can also first use an Initial Connection without Control Connection, and try to establish the Control Connection after some time. Once PLMT capability is detected and additional signaling information has been exchanged, the Initial Connection as well as potential additional Subflow Connections can then be used to transport PLMT TLV-encoded data traffic. This mechanism avoids needless Control Connection setups for short transfers.

- o PLMT needs a well known, dedicated port for the Control Connections, similar to TLS [3]. If PLMT is enabled on a host, it may try to establish Control Connections to that port for all communication partners. Even if heuristics can be used to learn whether servers are supporting PLMT, or not, and thus reduce the connection setup attempts, numerous legacy hosts in the Internet will receive connection setups on that port. To legacy systems, this may look similar to a SYN flooding attack. As a counter measure, network administrators may configure firewalls to block the PLMT Control Port, which prevents the usage of the protocol once it is more widely deployed.

- Middlebox that transparently change the length of content are a problem for multipath transport protocols. When using TLV-based transport, PLMT could detect such middleboxes by using a checksum, or by observing broken TLV headers, and try retransmissions. However, if the byte stream is transparently changed before switching to TLV encoding, difficulties can arise. For instance, the Signature may not be at the position where it is expected. In this case, PLMT cannot enter the TLV mode, but it can also not necessarily fall back, and it may either have to cancel that transfer by closing the PLMT Session, or, in the worst case, it may even deliver corrupted data to an application.

- PLMT delays the setup of connections in various scenarios. If an Active Opener wants to use TLV encoding immediately on the Initial Connection, it must await the setup of the control connection. If there is no response (no SYN/ACK), the Active Opener may either retransmit the SYN, i. e., wait for a longer time, or give up. Then, multipath transport is not possible. In all cases, there is at least a small delay before the data transport over the Initial Connection can start. If the Active Opener decides to setup the Control Connection later, this delay is avoided. But then the Active Opener must stop data transmission after the setup of the Control Connection, in order to ensure a safe exchange of tokens, which interrupts the data transport.

- The Passive Opener has a significant processing overhead due to PLMT. First and most obviously, there is the overhead of maintaining the Control Connections, which can be significant for a highly-loaded server with thousands of connections.

- The second and trickier challenge is the distinction between legacy TCP connections and connection originating from hosts that use PLMT. PLMT Subflow Connections are characterized by the presence of the Signature in the byte stream. This means that the PLMT layer must accept all incoming connections, parse for the presence of a valid Signature, and then decide whether it is a legacy connection or a connection transporting PLMT content with TLV encoding. The parsing for Signatures is difficult if an incoming connection sends less data than the length of the Signature. If the first bytes match a valid Signature, or if no bytes are received at all, the PLMT layer must wait for the arrival of further data, or time out, e. g., if the corresponding application does not send enough bytes. If it times out, the only safe option is to close the connection. This means that the PLMT layer may reject not only PLMT connections that suffered from retransmissions within the first byte, but also valid TCP connection setup from legacy stacks if they happen to (partly) match a Signature. If the delayed setup of Control Connections is allowed, the parsing overhead is even larger. The PLMT layer must then parse all established TCP connections for all valid Signatures at the negotiated positions in the byte stream, which may also require temporary buffering of data, if only parts of a valid Signature are received, or if the rest of the first TLV message is missing. In all cases, the delivery of data to applications may be delayed.

- On a Passive Opener, the PLMT layer has to accept incoming connections in order to parse the payload, before it can hand over the connection to the application. This can delay data delivery, and also may result in inconsistent views when the connection is indeed established. Further studies are needed to understand whether the delay of connection establishment as seen by applications, which does not occur in case of option-based multipath protocols, could break existing applications.

- Due to the processing and buffer overhead required to identify connections by payload parsing, the Passive Opener is vulnerable to a Denial-of-Service (DoS) attack: An attacker can open a large number of Control Connections, which will consume resources on a server and slow down data delivery on other connections. Passive Openers can reduce the risk by only accepting Coupled Connections from source IP addresses that originate also an existing connection, but this does not offer a complete protection, in particular if an attacker is sitting behind a large NAPT middlebox. Another remedy is to limit the amount of allowed Control Connections, but then other users of PLMT suffer from the effects of Control Connection setup failures.

- PLMT must exchange the Token information in the payload of the Initial Connection, in order to verify that an Initial Connection and a Coupled Connection indeed have the same endpoints. This requires the transport of a TLV-encoded message. As a consequence, unlike other multipath transport protocols [6] [9], PLMT cannot fall back to a backward compatible byte stream transport if a middlebox on the path should block the TLV transport.

- If there is a single-homed Active Opener and a multi-homed Passive Opener, PLMT cannot indicate to the Active Opener that multipath transport may make sense, i. e., that it could establish a Control Connection, before that connection actually exists. Other multipath transport protocols [6] [9] have a signaling mechanism for this. PLMT can only detect this situation if it blindly opens Control Connections in all cases.

- If a middlebox does not intercept the information on the Control Connections, or if it does not know the Signature by other means, it cannot determine if a given TCP connection transports PLMT data, or not. If a middlebox is not on the path of the Control Connection, it cannot prevent the usage of TLV encoding. For the latter case, a possible remedy would be that Additional Subflow Connections use another well-known port, which could then be blocked.

- A Passive Opener can accept with a certain, small probability erroneously a connection from a legacy host as PLMT Subflow Connection, if an application happens to send a bit pattern that is identical to one of the valid Signature of that Passive Opener, plus the valid Tokens. This may either happen if the first bytes of a standard TCP connection match an active Signature, or if a corresponding bit pattern is present exactly at the same sequence position as negotiated on a control connection. In that case, TLV-encoded content will be injected into a legacy connection, which will be corrupted. Due to the length of the Signature, this error probability is very small.

- An attacker can abuse PLMT to break legacy TCP connections to a PLMT-enabled Passive Opener, if it is sitting behind the same NAPT middlebox like another Active Opener, as already explained. In this case, the attacker can open multiple Control Connections, not only as a DoS attack, but also to attack other users. With a very small probability, the Signature and Tokens negotiated over the Control Connection will match another connection. If so, TLV content will be injected on that connection, and it will break, too. Again, the success probability of this attack is very small.

In summary, PLMT is a multipath protocol that is designed as a payload-only solution. It is useful for controlled and trusted environments, for networks with middleboxes that affect the use of TCP options, and for use cases where it is impossible to change the network stack.

4. PLMT Protocol

This section details the operations of PLMT protocol.

4.1. Session Initiation

A session initiation begins with an application request for a new TCP connection, upon which the PLMT protocol performs the following actions.

4.2. Exchange of PLMT Signaling Over the PLMT Control Channel

A node MAY setup a TCP Control Connection before or in parallel to the setting up of the Initial Connection (Parallel Setup), or it MAY set up the Control Connection at a later point in time (Late Setup). Both variants have advantages and drawbacks and affect the way how the Initial Connection is used.

4.2.1. Establishment of the Control Connection

The Active Opener Must set up the TCP Control Connection using the same source and destination IP addresses, and it MUST be destined to the PLMT Control Port. If the TCP connection is successfully set up, this is a first indication that the Passive Opener indeed supports

PLMT. In order to exclude the case that another service is accidentally running on that port, PLMT support is further verified by PLMT Capable Messages.

A Passive Opener SHOULD verify whether there are already established TCP connections from the same Active Opener, in order to reduce the vulnerability to DoS attacks.

4.2.2. PLMT Capable Messages

If the Control Connection is set up successfully, the two hosts can be expected to have an operational PLMT Shim Layer. The End-host MUST exchange the Tokens as shown in Figure 3 for further validation of the existence of PLMT Shim layer and the willingness of the Passive Opener to use PLMT. Note that at this stage of the signaling the Passive Opener cannot safely identify the Initial Connection that this Control Connection shall be associated with.

```
            End-host A                                 End-host B
     ---------------------------             ---------------------------
      Address A1     Address A2               Address B1     Address B2
     ------------   ------------             ------------   ------------
           |               |                           |              |
           |     (Control Connection setup (TCP))      |              |
           |~~~~~~~~~~~~~~~SYN~~~~~~~~~~~~~~~~~~~~~~~>|              |
           |<~~~~~~~~~~~~SYN/ACK~~~~~~~~~~~~~~~~~~~~~~|              |
           |~~~~~~~~~~~~~~~ACK~~~~~~~~~~~~~~~~~~~~~~~>|              |
           |               |                           |              |
           |       (PLMT Capable Signaling)            |              |
           |~~~~~~~~~~PLMT Token Indication~~~~~~~~~~>|              |
           |<~~~~~~~~PLMT Token Confirmation~~~~~~~~~~|              |
           |               |                           |              |
```

Figure 3 PLMT Signaling Exchange over the Control Connection

The frame format of the PLMT Token Indication message is shown in Figure 4. The Token is a unique number for a host and is used to identify a particular PLMT Session. To make it harder for an attacker to guess the Token by brute-force method, a 64-bit Token SHOULD be generated randomly [7]. Furthermore, the PLMT Token Indication message includes the Signature of the Active Opener, as well as the byte position in the Initial Connection where this Signature will be present on the Initial Connection. The byte position is provided in the Token Indication in order to reduce the parsing overhead of a Passive Opener, and the risk that an attacker can hijack a connection by negotiation of a large number of Signatures and Tokens with a Passive Opener. This implies that an

Active Opener can only send data up to this position before it receives a PLMT Token Confirmation message. In case of a Parallel connection setup, this position is set to 0, as the Signature is set at the beginning of the connection. As a side note, the whole mechanism can fail if the bytestream length is affected by a middlebox.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+-------------------------------+---------------+
|Kind=TOKENIND  |           Length=32           |   reserved    |
+---------------+-------------------------------+---------------+
:          Active Opener Signature (in total 16 byte)           :
+---------------------------------------------------------------+
:          Active Opener Token (in total 8 bytes)               :
+---------------------------------------------------------------+
|                    Signature offset (4 byte)                  |
+---------------------------------------------------------------+
```

Figure 4 PLMT Token Indication message (sent via the Control Connection)

As a response to the reception of the PLMT Token Indication from the Active Opener, the Passive Opener SHOULD either send back an own Token in a PLMT Token Confirmation message shown in Figure 5, or it SHOULD immediately close the Control Connection instead. This message also echoes back the Active Opener's Token, in order to verify that the reply is indeed sent by a PLMT layer.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+-------------------------------+---------------+
|Kind=TOKENCONF |            Length=36          |  reserved     |
+---------------+-------------------------------+---------------+
:         Passive Opener Signature (in total 16 byte)           :
+---------------------------------------------------------------+
:              Passive Opener Token (in total 8 byte)           :
+---------------------------------------------------------------+
:       Echo of Active Opener Token (in total 8 bytes)          :
+---------------------------------------------------------------+
```

Figure 5 PLMT Token Confirmation message (sent via the Control Connection)

Upon reception of that message, the Active Opener MUST first verify the validity of the message (in particular the echoed Token). If the message is valid, it MUST send the Signature provided from the Passive Opener at the indicated byte position in the Initial Connection, directly followed by a PLMT Token Message. Afterwards, TLV framing has to be used. The Passive Opener must similarly react: After having received the Signature and Token on an Initial Connection, the Passive Opener MUST send the Active Opener's Signature and a PLMT Token Message over the Initial Connection, too, and use TLV framing afterwards. Thus, after having sent the Signature, the Active Opener must parse all incoming bytes on the Initial Connection for the Signature of the Passive Opener, in order to detect the begin of TLV transfer in the reverse direction. In the simplest case, the Passive Opener has not sent any data in the meantime, i. e., the Signature is received immediately. However, other cases are possible, too.

Note that this method is inefficient and also has a very small risk of false positives, as it requires byte-wise parsing of the byte stream. Yet, the fundamental problem is that the Passive Opener cannot provide a byte offset for the Signature over the Control Channel during the PLMT Capability Signaling phase, as the Initial Connection and the Control Connection cannot be associated at that time. As an optimization, the Passive Opener could provide a bytestream offset by a separate signaling message once it has received the Token on the Initial Connection, but PLMT cannot rely on this, as the Control Connection could fail or stall in the meantime and then the PLMT session would not be in consistent state. The PLMT signaling exchange is designed to reflect an atomic transaction.

4.2.3. Further Usage of the Control Connection

The Control connection is only needed to exchange token information and to verify the association with the Initial Connection. After the PLMT capability exchange has been completed, the control connection is actually not needed any more, and it MAY be closed. All further control information, such as additional addresses etc., can also be exchanged over the Subflow Connections, by corresponding TLV messages. However, the Control Connection MAY also be kept established and used for further PLMT signaling. In particular, it could be useful to exchange address information over the Control Connection instead of the Subflow Connections. This would enable future NAPT helper for the PLMT protocol that could try to translate private to public addresses. A detailed discussion of this is outside the scope of this document.

4.2.4. Discussion of Control Connection Failure Cases

A failure to setup a Control Connection is an indication that the other end host does not have a PLMT Layer, or that middleboxes do not allow the establishment of a PLMT Control Connection. An Active Opener MUST await the successful PLMT capability exchange on the Control Connection before starting to send the Signature and TLV encoded content. An Active Opener MAY also give up after a certain waiting time. Then, it MUST close the Control Connection, and use backward compatible bytestream transport on the Initial Connection.

The PLMT capability exchange requires a single exchange of messages on the Control Connection only. If the Connection fails afterwards, all control information can be exchanged over Subflow Connections. If the control connection fails and the Active Opener does not receive the Token Confirmation message, without that the Passive Opener detects this, there may be a synchronization mismatch and the Passive opener may inject a Signature and a Token to the Initial Connection even if this is not expected by the Active Opener. In order to avoid data corruption, the Active Opener could parse all incoming data for the Signature after failure of a Control Connections, but this may increase the processing overhead.

If a Control Connection fails after the exchange of the tokens, PLMT could in principle continue to operate, since TLV encoded data can be transported over the established Subflow Connections, and since the Signatures and Tokens are already known.

4.3. PLMT Data Connection Setup and Operation

PLMT provides two modes of operation, which differ by the time when the control connection is established: Parallel Setup and Late Setup. The Parallel Setup is significantly simpler for a Passive Opener, as Signatures are sent in the first bytes of a connection and therefore are simple to identify. But, unfortunately, the setup of a Control Connection for every data transfer with a short duration results in overhead and additional delay without any performance gains. This mode is therefore mainly useful if it is known in advance that a TCP connection will transport a large amount of data. In order to reduce the overhead for short connection, PLMT also allows that the Control Connection is established later than the Initial Connection. In this case, the PLMT Layer on a host MUST not initiate the TLV data encoding before the PLMT capability of the other host has been determined through the Control Connection, (cf. Figure 3).

4.3.1. Guidelines for selection of a Signature

To allow for a simple identification of where exactly the TLV encoding inside the byte stream starts, a 128-bit Signature is used, which is used as a delimiter between bytestream and TLV encoding (cf. Figure 6). The Signature is selected by the hosts that must parse it, and MUST be chosen such that collisions with existing application protocols are minimal. Note that it is up to the hosts to decide what Signature to use for different connections The most secure solution is to use a different Signature for every Control Connection, but then the parsing effort is the largest. For performance optimization, the PLMT Layer at a host MAY use the same Signature in more than one connection, but it MUST change the value on a regular basis.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------------------------------------------------------+
|                       Signature (16 byte)                       :
+-----------------------------------------------+-----------------+
:                           Signature                             :
+-----------------------------------------------+-----------------+
:                           Signature                             :
+-----------------------------------------------+-----------------+
:                           Signature                             |
+-----------------------------------------------------------------+
```

Figure 6 PLMT Signature (sent on Subflow Connections)

4.3.2. Bundling of Initial Connection to the Control Connection in Parallel Setup

The Control Connection is used to determine the PLMT capability of the end hosts. The Initial Connection MUST not transport any data before the Control Connection is established and the PLMT Capability Exchange is completed. If the Control Connection setup or PLMT Capability Exchange fails, then the Initial Connection MUST not transmit data with TLV encoding but the legacy TCP bytestream.

Before using TLV encoding, a host must first send the Signature on the Initial Connection as depicted in Figure 7. The first TLV-encoded messages after that delimiter must exchange the tokens to bundle the Initial Connection with the Control Connection, and to verify at both endpoints that the Initial Connection and the Control Connection indeed terminate at the same host. The tokens are exchanged by a Token Indication and a Token Confirmation message. After these messages, both sides are allowed to send other PLMT messages in TLV encoding over the Connection, or to establish

further Subflow Connections. Both Active and Passive Opener must verify the Tokens. If the Tokens do not match the ones exchanged over the control connection, the PLMT session must be closed, as apparently an error has occurred.

```
            End-host A                                 End-host B
     ---------------------------             ---------------------------
      Address A1     Address A2               Address B1     Address B2
     ------------   ------------             ------------   ------------
           |               |                          |              |
           |      (Initial Connection setup (TCP))    |              |
           |---------------SYN----------------------->|              |
           |<------------SYN/ACK----------------------|              |
           |---------------ACK----------------------->|              |
           |               |                          |              |
           |    (PLMT Capability of the Other End-host has been      |
           |         determined over the Control Connection)         |
           |               |                          |              |
           |    (First TLV encoded message exchange   |              |
           |         over the Initial Connection)     |              |
           |---B's Signature + Token B Verification-->|              |
           |               |                          | Token        |
           |               |                          | verif.       |
           |<--A's Signature + Token A Verification---|              |
    Token  |               |                          |              |
    verif. |               |                          |              |
           |               |                          |              |
           |     (TLV encoded data transport          |              |
           |     over the Initial Connection)         |              |
           |---------------TLV----------------------->|              |
           |<--------------TLV------------------------|              |
           |               |                          |              |
```

Figure 7 Bundling of Initial PLMT Subflow Connection and Control Connection for Parallel Setup

```
                                 1                   2                   3
           0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
         +---------------+---------------------------------+-------------+
         |Kind=TOKEN     |                 Length=12       |  reserved   |
         +---------------+---------------------------------+-------------+
         |                         Token (8 byte)                        |
         +---------------------------------------------------------------+
```

Figure 8 PLMT Token Verification Message (sent over the Initial Connection)

4.3.3. Bundling of Initial Connection to the Control Connection in Late Setup

In order to avoid the setup overhead of control Connections for short-lived transfers, the PLMT protocol MAY establish the Control Connection after data has already been exchanged on the Initial Connection. This document does not describe heuristics when to set up the Control connection. They may take into account factors such as number of bytes transferred, cached information about support of PLMT, or user preferences.

A receiver MUST assume that all bytes received on an incoming TCP connection are sent by legacy end system, before a match with a valid Signature is possible. Until then, all data must be passed to the application in unmodified form. Thus, PLMT risks with a very small probability that corrupted data is delivered to an application.

Once the Control Connection is established and the PLMT capability information of the end hosts has been exchanged, the Active Opener can send the Passive Opener's Signature and a PLMT Token Verification message over the Initial Connection, at the position in the byte stream that has been advertised over the control channel. The mechanism of token exchange in the payload of the Initial Connection is used to verify that the Initial Connection and Control Connection actually involve the same hosts.

```
       End-host A                           End-host B
 ---------------------------             ---------------------------
  Address A1     Address A2               Address B1     Address B2
 ------------   ------------             ------------   ------------
       |               |                          |              |
       |      (Initial Connection setup (TCP))    |              |
       |---------------SYN----------------------->|              |
       |<------------SYN/ACK----------------------|              |
       |---------------ACK----------------------->|              |
       |               |                          |              |
       |               (Data Segments             |              |
       |     sent over the Initial Connection)    |              |
       |----------------------------------------->|              |
       |<-----------------------------------------|              |
       |               |                          |              |
       |     (Control Connection setup (TCP))     |              |
       |~~~~~~~~~~~~~~~SYN~~~~~~~~~~~~~~~~~~~~~~~>|              |
       |<~~~~~~~~~~~~SYN/ACK~~~~~~~~~~~~~~~~~~~~~~|              |
       |~~~~~~~~~~~~~~~ACK~~~~~~~~~~~~~~~~~~~~~~~>|              |
```

```
           |               |                          |              |
           |   (TLV-Enabled PLMT Control Signaling    |              |
           |     sent over the Control Connection)    |              |
           |~~~Sign. indic. (A's sign., A's token)~~~>|              |
           |<~~Sign. confirm. (B's sign., B's token)~~|              |
           |               |                          |              |
           |       (Message exchange over the         |              |
           |           Initial Connection)            |              |
           |---B's Signature + Token B verification-->|              |
           |                                          | Token        |
   ........|..........................................| verif.       |
           |<--A's Signature + Token A verification---|              |
    Token  |               |                          |              |
    verif. |     (TLV encoded data transport          |              |
           |     over the Initial Connection)         |              |
           |---------------TLV----------------------->|              |
           |<--------------TLV------------------------|              |
           |               |                          |              |
```

Figure 9 Bundling of PLMT First Subflow Connection and Control Connection for Delayed Setup

4.4. Additional Subflow Connections Initiation and Operation

4.4.1. Address Advertisement

The Initial Subflow Connection, as well as the Control Connection, is established by the Active Opener. Once TLV encoding is enabled on the Initial Subflow Connection, and it is thus verified that the two end-hosts are PLMT capable, any of the end-hosts MAY initiate further Subflow Connections. PLMT assumes that at least one of the two connection endpoints is multihomed, i. e., has at least two IP addresses. The end-hosts MAY exchange these addresses via the Control Connection or via any Subflow Connection, once TLV transport is enabled. The frame format of advertising and releasing addresses is given in Figure 10 and 11, respectively.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+-------------------------------+-------+-------+
| Kind=ADD_ADDR |             Length            | IPVer | (res) |
+---------------+-------------------------------+-------+-------+
|          Address (IPv4 - 4 octets / IPv6 - 16 octets)         |
+---------------------------------------------------------------+
```

Figure 10   PLMT Add Address

```
                             1                   2                   3
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
        +---------------+-------------------------------+-------+-------+
        | Kind=DEL_ADDR |               Length          | IPVer | (res) |
        +---------------+-------------------------------+-------+-------+
        |          Address (IPv4 - 4 octets / IPv6 - 16 octets)         |
        +---------------------------------------------------------------+
```

Figure 11   PLMT Remove Address

4.4.2. Subflow Connection Setup

For each initiation of an additional Subflow Connection, a new TCP connection is initiated with a three-way handshake (SYN, SYN/ACK, ACK). The Signatures are used by both ends to distinguish Subflow Connections from normal TCP connection, and to detect the start of TLV encoding. If a Subflow Connection is established that shall carry TLV Data Segments, a sender MUST send the Signature first before starting to send TLV Data Segments. In all cases, the first Data Segment after the Signature MUST be a Token Indication (from Active Opener) or Token Confirmation message (from Passive Opener). This setup of an additional Subflow Connection is illustrated in Figure 12.

```
         End-host A                                End-host B
     ---------------------------             ---------------------------
      Address A1     Address A2               Address B1     Address B2
     ------------   ------------             ------------   ------------
           |               |                          |              |
           |     (TLV encoded Data Segments)          |              |
           |----------------------------------------->|              |
           |<-----------------------------------------|              |
           |               |                          |              |
           |   (Over Subflow or Control Connection)   |
           |<--------------ADD_ADDR-B2----------------|              |
           |               |                          |              |
           |      (Additional Subflow Connection Setup (TCP))        |
           |***************************SYN**************************>|
           |<************************SYN/ACK*************************|
           |***************************ACK**************************>|
           |               |                          |              |
           |***B's Signature + Token B verification*****************>|
           |                                          | Token        |
   ........|..........................................| verif.       |
           |<**A's Signature + Token A verification******************|
    Token  |               |                          |              |
    verif. |     (TLV encoded data transport          |              |
```

```
|  over the additional Subflow Connection) |              |
|***************TLV**************************************>|
|               |                          |              |
|<**************TLV***************************************|
```

Figure 12 Additional Subflow Connection setup

4.4.3. TLV Encoding of Data Segments

TLV encoded Data Segments can be sent on each Subflow Connection. Each Data Segment carries a 64-bit Session Sequence Number. A PLMT-capable host must maintain a Session Sequence Number in addition to the TCP sequence numbers of TCP on a Subflow Connection.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------+-------------------------------+--------------+
|   Kind = DATA   |         Length=20+n           |  reserved    |
+-----------------+-------------------------------+--------------+
:                   Session Sequence Number (8 byte)             :
+----------------------------------------------------------------+
:                    Data Segment (n bytes total)                |
+----------------------------------------------------------------+
```

Figure 13 TLV encoded Data Segment message

Session Sequence Numbers are used to reorder the data inside the PLMT session that arrives over multiple Subflow Connections. The Session Sequence Number is thus similar to the TCP sequence number and identifies each byte of data. Each Data Segment carries the Session Sequence Number, which refers to the byte number of the first byte in the Data segment.

Even when a PLMT-capable host is not transmitting TLV data segments, the end host MUST store Session Sequence Numbers for all ongoing TCP connections, in order to be able to deal with late setups of a Control Connection.

4.4.4. Data Acknowledgments

In addition to the regular Subflow Connection TCP acknowledgements, session-level Data Acknowledgements are used to cumulatively acknowledge the data received over the different Subflow Connections. A Data Acknowledgement that acknowledges the reception of a Data Segment message includes the next expected byte of Data Segments. In a normal operation, session-level Data Acknowledgements are actually not needed, but certain performance enhancing proxies

or middlebox failures may result in situations in which the acknowledgments on a SubFlow Connection erroneously allows release of data in the sender, even if it is not yet received.

The Data Acknowlegdements also include a session-level receive window to correctly perform flow control at session level, and to avoid deadlocks.

Since the use of data acknowledgements is only a mechanism to increase robustness, the data acknowledgements SHOULD be sent at bigger intervals of time. It is left for further study how often they should be sent. Another open question is on which of the connections the messages should be sent.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+-------------------------------+--------------+
|Kind=SESS_ACK  |           Length=12           |  reserved    |
+---------------+-------------------------------+--------------+
:    Next expected Session Sequence Number (8 byte in total)    :
+--------------------------------------------------------------+
:          Session receive window (8 bytes in total)            :
+--------------------------------------------------------------+
```

Figure 14 Data Acknowledgement message

4.5. Other Aspects

4.5.1. Congestion Control

One of the goals for having a multi-connection transport solution is to enhance the usage of network resources, commonly known as resource pooling principle. In order to achieve resource pooling, the congestion windows of the different Subflow Connections of the Session should be coupled together. The coupling should lead to transmission of more Data Segments over the less congested connections as compared to the more congested connections.

Different congestion control algorithms may be implemented for multipath transport mechanisms to achieve the goals of resource pooling and fairness. One such algorithm is presented in [4]. The algorithm offers a potential solution in the current Internet by controlling the Subflow Connection congestion window increase as a function of the performance of other Subflow Connections of a session.

PLMT could use this algorithm for congestion control as well. If PLMT is entirely implemented in the user space, an alternative algorithm could be used that runs a corresponding scheduler, which uses own estimates for the path characteristics. The design of alternative algorithms for congestion control coupling is beyond the scope of this document.

4.5.2. Path Management and Scheduling

The establishment of multiple Subflow Connections to different addresses aims at a better utilization of the network resources. PLMT could use cross-layer information from the network layer for path management.

The scheduling of TLV-encoded Data Segments over the different Subflow Connections is based on the local policy. PLMT can use different algorithms to control the splitting of the data stream from the application over the different Subflow Connections. PLMT uses the standard TCP mechanisms for reliable transport of data on its Subflow Connections.

The retransmission strategy for lost Data Segments is a local policy. The session sequence number allows lost Data Segments to be sent over another Subflow Connection in addition to the retransmission over the same Subflow Connection. How often a Data Segment is sent over another Subflow Connection is again a design choice of the local policy.

4.5.3. Closing Connections and Sessions

A Subflow Connection is a standard TCP connection. To close a Subflow Connection the TCP 4-way FIN handshake mechanism is used.

When the Session needs to be closed, it means that all the PLMT Connections need to be closed, including the Control Connection.

5. Interaction with Middleboxes

The Internet consists of many different types of middleboxes, some parse the contents of the stream of a TCP connection, rewrite the content of packet headers or rewrite even the payload. For a new multipath transport like PLMT to be successfully deployable, its operation should be understood and tested against such middleboxes. Examples for well-known middleboxes are Network Address and Port Translators (NAPT). PLMT is designed to be compatible with middleboxes that have problems with TCP options. But there are also some problems with other types of middleboxes.

5.1. Middleboxes that Translate Address/Ports

Middleboxes that perform Network Address and Port Translations (NAPT) may cause problems for the creation of multiple connections (this is a potential issue for all multipath transport protocols). Hosts behind the NAPT know their local addresses but might not be aware of the global addresses that the NAPT uses. Therefore, the hosts MUST NOT advertise their multiple local addresses to the other host. The host behind the NAPT MAY still be multipath capable and MAY open a PLMT connection to the other host if the other host is also PLMT capable. Over the established PLMT connection, the other host MAY advertise its multiple addresses. These addresses will be used by the host behind the NAPT to open further Subflow Connections.

5.2. Middleboxes that Manipulate TCP Options

The multipath solutions that use TCP options field for their operation may suffer from middleboxes that may remove or modify the TCP options. Some middleboxes may even drop packets with unknown TCP options, and this may happen for the connection establishment packets as well. PLMT does not employ any new TCP option and hence it would not be affected by such a middlebox behavior.

5.3. Middleboxes that Parse Content

Current middleboxes in the Internet are not aware of multipath transport. Therefore, middleboxes will identify the single Subflow Connection to be a standard TCP connection. The TLV encoding of the payload may confuse the middlebox and may lead the middlebox to stall the connection in case that the middlebox parses the content.

If a middlebox blocks TLV encoding, PLMT can try to transmit data over another path. However, PLMT cannot fall back to a mode that does not use TLV transport, since it must send the Signature and tokens in TLV encoding over the Initial Subflow Connection.

Middleboxes that want to prevent multipath transport can block connection setups to the well-known port. This prevents the use of multipath transport if a middlebox is both on the path of the Initial Subflow Connection and the Control Connection. A middlebox that is not on the path of the Control Connection cannot safely distinguish normal TCP connections and PLMT Subflow Connections with TLV transport.

5.4. Middleboxes that Change content

Middleboxes may also modify the payload and not only the packet headers. All the multipath solutions require a session-level data sequence number to re-order/combine the data stream received over the Subflow Connections. The PLMT design allows detecting such a middlebox behavior by identifying the connection which gets stalled due to undecodable TLV framing. In addition, checksums could be used. The Data Acknowledgements will identify the holes in the session sequence numbers so that a retransmission of the missing segments over other Subflow Connections will be initiated. This allows working around content-modifying middleboxes, unless they are present on all paths.

If this type of middlebox is present on the Initial Connection, then the Signature matching may fail. This means that data transport over the Initial Connection may be corrupted, as, e. g., the Signature may be delivered to the application as part of the byte stream.

6. Security Considerations

The Signature-based method to identify the setup of a new TLV-enabled data flow has two security issues: First, an application can accidentally generate a bit pattern that is equal to the Signature. Second, due to the use of out-of-band signaling, PLMT's method must be robust against malicious attacks that try to break or hijack PLMT sessions or normal connections. Unlike other multipath transport protocols, it is theoretically possible to attack a normal TCP connection to a PLMT-enabled server, even if it does not use multipath transport.

6.1. Reappearance of Signature in Application Data

The Signature (and the tokens) is sent in two different contexts:

- A connection which was started as a single legacy TCP connection is later switched to PLMT/TLV-enabled operation. In this case, the Active Opener provides the Session sequence number over the control connection of the last byte that is not TLV encoded. This way, the PLMT Layer of the Passive Opener knows how much user data has been transmitted through the legacy TCP connection and when to expect the Signature. Given the length of the Signature, as well as the following token exchange, it is extremely unlikely that a normal TCP connection is wrongly classified as a Subflow Connection. A similar problem occurs at the Active Opener.

- The Signature can also be present in the first bytes of a new PLMT Subflow Connection, if it is an additional Subflow Connection, or if the Control Connection is established first. In these cases, the Subflow Connection is characterized by the Signature being present in the first bytes of a connection. In case that an application itself opens an additional TCP connection to the same corresponding end host, a problem could occur if the Signature pattern (and follow-up token messages) is contained in the first data packet of the connection.

Because of both effects, there is a residual probability that PLMT accepts a connection erroneously, if an application accidentally sends a bit pattern that is identical to the Signature (plus the Tokens), of if an attacker manages to guess the pattern. This probability is very small as the Signature is a long, random bit pattern.

This probabilistic approach of a token-based identification is general practice in challenge-response authentication methods, where there is also an extremely small residual probability that an unauthorized (malicious) node guesses the response correctly.

6.2. Resilience against Malicious Attacks

One aspect of address-agile multi-path transport mechanisms are possible malicious attacks. PLMT suffers from a DoS vulnerability, but it has protection methods against other attacks.

PLMT uses the same token mechanism like other multipath transport protocols, but with much longer tokens. An attacker must not only correctly guess the Tokens, but also the Signature. As a consequence, the probability of blind guess attacks on PLMT is extremely small.

7. Open Issues

This PLMT protocol specification is a work-in-progress, and there are still remaining unsolved issues that need further considerations.

8. IANA Considerations

This document will make a request to IANA to allocate a new TCP/UDP port value for the PLMT Control Connection.

9. Conclusion

PLMT is a user-space solution to enable reliable, in-order data transfer over multiple paths. This specification defines the PLMT protocol. PLMT is defined as a worked example for a payload-based multipath transport, as an alternative to TCP option based signaling mechanisms. Due to some security vulnerabilities, it is mainly suitable for controlled and trusted environments.

10. References

10.1. Normative References

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

[2] J. Postel, ''Transmission Control Protocol'', STD 7, RFC 793, September 1981.

[3] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

10.2. Informative References

[4] Raiciu, C., Handley, M. and D. Wischik, ''Coupled Multipath-Aware Congestion Control'', draft-ietf-mptcp-congestion-00 (work in progress), July 2010.

[5] Ford, A., Raiciu, C., Barre, S. and J. Iyengar, ''Architectural Guidelines for Multipath TCP Development'', draft-ietf-mptcp-architecture-01 (work in progress), June 2010.

[6] Ford, A., Raiciu, C. and M. Handley, ''TCP Extensions for Multipath Operation with Multiple Addresses'', draft-ietf-mptcp-multiaddressed-01 (work in progress), July 2010.

[7] M. Bagnulo, ''Threat Analysis for Multi-addressed/Multi-path TCP'', draft-ietf-mptcp-threat-02 (work in progress), March 2010.

[8] Scharf, M. and A. Ford, ''MPTCP Application Interface Considerations'', draft-scharf-mptcp-api-02 (work in progress), July 2010.

[9] M. Scharf, ''Multi-Connection TCP (MCTCP) Transport'', draft-scharf-mptcp-mctcp-00 (work in progress), July 2010.

11. Acknowledgments

The authors are supported by the German-Lab project (http://www.german-lab.de/), a research project funded by the German Federal Ministry of Education and Research (BMBF). The views expressed here are those of the author(s) only. The BMBF is not liable for any use that may be made of the information in this document.

The authors gratefully acknowledge significant input into this document from Koojana Kuladinithi, Asanga Udugama, Andreas Koensgen, Andres Toro (all from University of Bremen), Andreas Timm-Giel (Hamburg University of Technology), Thomas-Rolf Banniza and Peter Schefczik (all from Alcatel-Lucent Bell Labs).

Authors' Addresses

Amanpreet Singh
University of Bremen
Otto-Hahn-Allee 1
28359 Bremen
Germany

Email: aps@comnets.uni-bremen.de

Michael Scharf
Alcatel-Lucent Bell Labs
Lorenzstrasse 10
70435 Stuttgart
Germany

EMail: michael.scharf@alcatel-lucent.com