

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: July 25, 2011

A. Ford  
Roke Manor Research  
C. Raiciu  
M. Handley  
University College London  
S. Barre  
Universite catholique de  
Louvain  
J. Iyengar  
Franklin and Marshall College  
January 21, 2011

Architectural Guidelines for Multipath TCP Development  
draft-ietf-mptcp-architecture-05

Abstract

Hosts are often connected by multiple paths, but TCP restricts communications to a single path per transport connection. Resource usage within the network would be more efficient were these multiple paths able to be used concurrently. This should enhance user experience through improved resilience to network failure and higher throughput.

This document outlines architectural guidelines for the development of a Multipath Transport Protocol, with references to how these architectural components come together in the development of a Multipath TCP protocol. This document lists certain high level design decisions that provide foundations for the design of the MPTCP protocol, based upon these architectural requirements.

Status of this Memo

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

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

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

This Internet-Draft will expire on July 25, 2011.

## Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	4
1.1. Requirements Language . . . . .	5
1.2. Terminology . . . . .	5
1.3. Reference Scenario . . . . .	6
2. Goals . . . . .	6
2.1. Functional Goals . . . . .	6
2.2. Compatibility Goals . . . . .	7
2.2.1. Application Compatibility . . . . .	7
2.2.2. Network Compatibility . . . . .	8
2.2.3. Compatibility with other network users . . . . .	9
2.3. Security Goals . . . . .	10
2.4. Related Protocols . . . . .	10
3. An Architectural Basis For Multipath TCP . . . . .	10
4. A Functional Decomposition of MPTCP . . . . .	12
5. High-Level Design Decisions . . . . .	14
5.1. Sequence Numbering . . . . .	14
5.2. Reliability and Retransmissions . . . . .	15
5.3. Buffers . . . . .	17
5.4. Signalling . . . . .	18
5.5. Path Management . . . . .	19
5.6. Connection Identification . . . . .	20
5.7. Congestion Control . . . . .	21
5.8. Security . . . . .	21
6. Software Interactions . . . . .	22
6.1. Interactions with Applications . . . . .	22
6.2. Interactions with Management Systems . . . . .	23
7. Interactions with Middleboxes . . . . .	23
8. Contributors . . . . .	25
9. Acknowledgements . . . . .	25
10. IANA Considerations . . . . .	25
11. Security Considerations . . . . .	25
12. References . . . . .	26
12.1. Normative References . . . . .	26
12.2. Informative References . . . . .	26
Appendix A. Changelog . . . . .	28
A.1. Changes since draft-ietf-mptcp-architecture-04 . . . . .	28
A.2. Changes since draft-ietf-mptcp-architecture-03 . . . . .	28
A.3. Changes since draft-ietf-mptcp-architecture-02 . . . . .	28
A.4. Changes since draft-ietf-mptcp-architecture-01 . . . . .	28
A.5. Changes since draft-ietf-mptcp-architecture-00 . . . . .	28
Authors' Addresses . . . . .	28

## 1. Introduction

As the Internet evolves, demands on Internet resources are ever-increasing, but often these resources (in particular, bandwidth) cannot be fully utilised due to protocol constraints both on the end-systems and within the network. If these resources could be used concurrently, end user experience could be greatly improved. Such enhancements would also reduce the necessary expenditure on network infrastructure that would otherwise be needed to create an equivalent improvement in user experience. By the application of resource pooling [3], these available resources can be 'pooled' such that they appear as a single logical resource to the user.

Multipath transport aims to realize some of the goals of resource pooling by simultaneously making use of multiple disjoint (or partially disjoint) paths across a network. The two key benefits of multipath transport are:

- o To increase the resilience of the connectivity by providing multiple paths, protecting end hosts from the failure of one.
- o To increase the efficiency of the resource usage, and thus increase the network capacity available to end hosts.

Multipath TCP is a modified version of TCP [1] that implements a multipath transport and achieves these goals by pooling multiple paths within a transport connection, transparently to the application. Multipath TCP is primarily concerned with utilising multiple paths end-to-end, where one or both end host is multi-homed. It may also have applications where multiple paths exist within the network and can be manipulated by an end host, such as using different port numbers with ECMP [4].

MPTCP, defined in [5], is a specific protocol that instantiates the Multipath TCP concept. This document looks both at general architectural principles for a Multipath TCP fulfilling the goals described in Section 2, as well as the key design decisions behind MPTCP, which are detailed in Section 5.

Although multihoming and multipath functions are not new to transport protocols (SCTP [6] being a notable example), MPTCP aims to gain wide-scale deployment by recognising the importance of application and network compatibility goals. These goals, discussed in detail in Section 2, relate to the appearance of MPTCP to the network (so non-MPTCP-aware entities see it as TCP) and to the application (through providing an service equivalent to TCP for non-MPTCP-aware applications).

This document has three key purposes: (i) it describes goals for a multipath transport - goals that MPTCP is designed to meet; (ii) it lays out an architectural basis for MPTCP's design - a discussion that applies to other multipath transports as well; and (iii) it discusses and documents high-level design decisions made in MPTCP's development, and considers their implications.

Companion documents to this architectural overview are those which provide details of the protocol extensions [5], congestion control algorithms [7], and application-level considerations [8]. Put together, these components specify a complete Multipath TCP design. We note that specific components are replaceable in accordance with the layer and functional decompositions discussed in this document.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2].

### 1.2. Terminology

Regular/Single-Path TCP: The standard version of the TCP [1] protocol in use today, operating between a single pair of IP addresses.

Multipath TCP: A modified version of the TCP protocol that supports the simultaneous use of multiple paths between hosts.

Path: A sequence of links between a sender and a receiver, defined in this context by a source and destination address pair.

Host: An end host either initiating or terminating a Multipath TCP connection.

MPTCP: The proposed protocol extensions specified in [5] to provide a Multipath TCP implementation.

Subflow: A flow of TCP segments operating over an individual path, which forms part of a larger Multipath TCP connection.

(Multipath TCP) Connection: A set of one or more subflows combined to provide a single Multipath TCP service to an application at a host.

### 1.3. Reference Scenario

The diagram shown in Figure 1 illustrates a typical usage scenario for Multipath TCP. Two hosts, A and B, are communicating with each other. These hosts are multi-homed and multi-addressed, providing two disjoint connections to the Internet. The addresses on each host are referred to as A1, A2, B1 and B2. There are therefore up to four different paths between the two hosts: A1-B1, A1-B2, A2-B1, A2-B2.

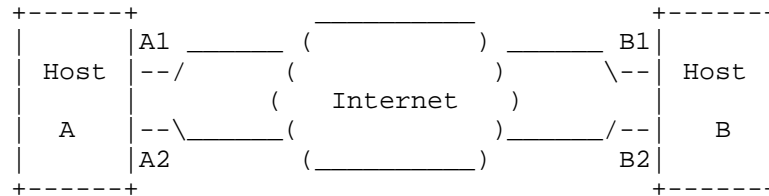


Figure 1: Simple Multipath TCP Usage Scenario

The scenario could have any number of addresses (1 or more) on each host, as long as the number of paths available between the two hosts is 2 or more (i.e.  $\text{num\_addr}(A) * \text{num\_addr}(B) > 1$ ). The paths created by these address combinations through the Internet need not be entirely disjoint - potential fairness issues introduced by shared bottlenecks need to be handled by the Multipath TCP congestion controller. Furthermore, the paths through the Internet often do not provide a pure end-to-end service, and instead may be affected by middleboxes such as NATs and Firewalls.

## 2. Goals

This section outlines primary goals that Multipath TCP aims to meet. These are broadly broken down into: functional goals, which steer services and features that Multipath TCP must provide; and compatibility goals, which determine how Multipath TCP should appear to entities that interact with it.

### 2.1. Functional Goals

In supporting the use of multiple paths, Multipath TCP has the following two functional goals.

- o Improve Throughput: Multipath TCP MUST support the concurrent use of multiple paths. To meet the minimum performance incentives for deployment, a Multipath TCP connection over multiple paths SHOULD achieve no lesser throughput than a single TCP connection over the best constituent path.

- o Improve Resilience: Multipath TCP MUST support the use of multiple paths interchangeably for resilience purposes, by permitting segments to be sent and re-sent on any available path. It follows that, in the worst case, the protocol MUST be no less resilient than regular single-path TCP.

As distribution of traffic among available paths and responses to congestion are done in accordance with resource pooling principles [3], a secondary effect of meeting these goals is that widespread use of Multipath TCP over the Internet should improve overall network utility by shifting load away from congested bottlenecks and by taking advantage of spare capacity wherever possible.

Furthermore, Multipath TCP SHOULD feature automatic negotiation of its use. A host supporting Multipath TCP that requires the other host to do so too must be able to detect reliably whether this host does in fact support the required extensions, using them if so, and otherwise automatically falling back to single-path TCP.

## 2.2. Compatibility Goals

In addition to the functional goals listed above, a Multipath TCP must meet a number of compatibility goals in order to support deployment in today's Internet. These goals fall into the following categories:

### 2.2.1. Application Compatibility

Application compatibility refers to the appearance of Multipath TCP to the application both in terms of the API that can be used and the expected service model that is provided.

Multipath TCP MUST follow the same service model as TCP [1]: in-order, reliable, and byte-oriented delivery. Furthermore, a Multipath TCP connection SHOULD provide the application with no worse throughput or resilience than it would expect from running a single TCP connection over any one of its available paths. A Multipath TCP may not, however, be able to provide the same level of consistency of throughput and latency as a single TCP connection. These, and other, application considerations are discussed in detail in [8].

A multipath-capable equivalent of TCP MUST retain some level of backward compatibility with existing TCP APIs, so that existing applications can use the newer transport merely by upgrading the operating systems of the end-hosts. This does not preclude the use of an advanced API to permit multipath-aware applications to specify preferences, nor for users to configure their systems in a different way from the default, for example switching on or off the automatic

use of multipath extensions.

It is possible for regular TCP sessions today to survive brief breaks in connectivity by retaining state at end hosts before a timeout occurs. It would be desirable to support similar session continuity in MPTCP, however the circumstances could be different. Whilst in regular TCP the IP addresses will remain constant across the break in connectivity, in MPTCP a different interface may appear. It is desirable (but not mandated) to support this kind of "break-before-make" session continuity. This places constraints on security mechanisms, however, as discussed in Section 5.8. Timeouts for this function would be locally configured.

### 2.2.2. Network Compatibility

In the traditional Internet architecture, network devices operate at the network layer and lower layers, with the layers above the network layer instantiated only at the end-hosts. While this architecture, shown in Figure 2, was initially largely adhered to, this layering no longer reflects the "ground truth" in the Internet with the proliferation of middleboxes [9]. Middleboxes routinely interpose on the transport layer; sometimes even completely terminating transport connections, thus leaving the application layer as the first real end-to-end layer, as shown in Figure 3.

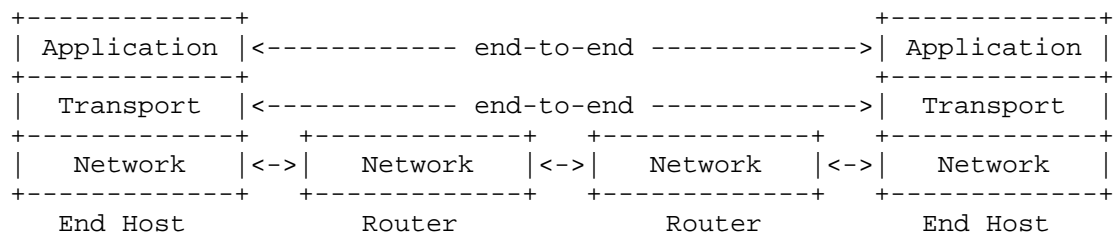


Figure 2: Traditional Internet Architecture

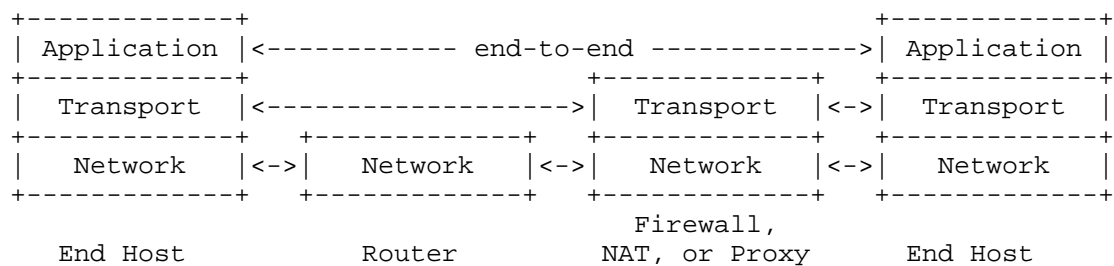


Figure 3: Internet Reality



Middleboxes that interpose on the transport layer result in loss of "fate-sharing" [10], that is, they often hold "hard" state that, when lost or corrupted, results in loss or corruption of the end-to-end transport connection.

The network compatibility goal requires that the multipath extension to TCP retains compatibility with the Internet as it exists today, including making reasonable efforts to be able to traverse predominant middleboxes such as firewalls, NATs, and performance enhancing proxies [9]. This requirement comes from recognizing middleboxes as a significant deployment bottleneck for any transport that is not TCP or UDP, and constrains Multipath TCP to appear as TCP does on the wire and to use established TCP extensions where necessary. To ensure end-to-endness of the transport, we further require Multipath TCP to preserve fate-sharing without making any assumptions about middlebox behavior.

A detailed analysis of middlebox behaviour and the impact on the Multipath TCP architecture is presented in Section 7. In addition, network compatibility must be retained to the extent that Multipath TCP MUST fall back to regular TCP if there are insurmountable incompatibilities for the multipath extension on a path.

Middleboxes may also cause some TCP features to be able to exist on one subflow but not another. Typically these will be at the subflow level (such as SACK [11]) and thus do not affect the connection-level behaviour. In the future, any proposed TCP connection-level extensions should consider how they can co-exist with MPTCP.

The modifications to support Multipath TCP remain at the transport layer, although some knowledge of the underlying network layer is required. Multipath TCP SHOULD work with IPv4 and IPv6 interchangeably, i.e. one connection may operate over both IPv4 and IPv6 networks.

### 2.2.3. Compatibility with other network users

As a corollary to both network and application compatibility, the architecture must enable new Multipath TCP flows to coexist gracefully with existing single-path TCP flows, competing for bandwidth neither unduly aggressively nor unduly timidly (unless low-precedence operation is specifically requested by the application, such as with LEDBAT). The use of multiple paths MUST NOT unduly harm users using single-path TCP at shared bottlenecks, beyond the impact that would occur from another single-path TCP flow. Multiple Multipath TCP flows on a shared bottleneck MUST share bandwidth between each other with similar fairness to that which occurs at a shared bottleneck with single-path TCP.

### 2.3. Security Goals

The extension of TCP with multipath capabilities will bring with it a number of new threats, analysed in detail in [12]. The security goal for Multipath TCP is to provide a service no less secure than regular, single-path TCP. This will be achieved through a combination of existing TCP security mechanisms (potentially modified to align with the Multipath TCP extensions) and of protection against the new multipath threats identified. The design decisions derived from this goal are presented in Section 5.8.

### 2.4. Related Protocols

There are several similarities between SCTP [6] and MPTCP, in that both can make use of multiple addresses at end hosts to give some multi-path capability. In SCTP, the primary use case is to support redundancy and mobility for multihomed hosts (i.e. a single path will change one of its end host addresses); the simultaneous use of multiple paths is not supported. Extensions are proposed to support simultaneous multipath transport [13], but these are yet to be standardised. By far the most widely used stream-based transport protocol is, however, TCP [1], and SCTP does not meet the network and application compatibility goals specified in Section 2.2. For network compatibility, there are issues with various middleboxes (especially NATs) that are unaware of SCTP and consequently end up blocking it. For application compatibility, applications need to actively choose to use SCTP, and with the deployment issues very few choose to do so. MPTCP's compatibility goals are in part based on these observations of SCTP's deployment issues.

## 3. An Architectural Basis For Multipath TCP

We now present one possible transport architecture that we believe can effectively support the goals for Multipath TCP. The new Internet model described here is based on ideas proposed earlier in Tng ("Transport next-generation") [14]. While by no means the only possible architecture supporting multipath transport, Tng incorporates many lessons learned from previous transport research and development practice, and offers a strong starting point from which to consider the extant Internet architecture and its bearing on the design of any new Internet transports or transport extensions.

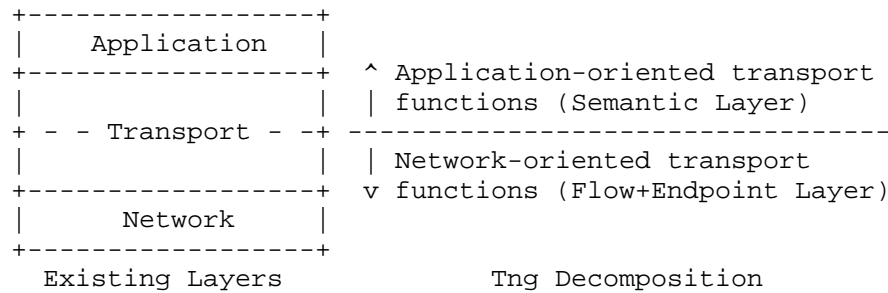


Figure 4: Decomposition of Transport Functions

Tng loosely splits the transport layer into "application-oriented" and "network-oriented" layers, as shown in Figure 4. The application-oriented "Semantic" layer implements functions driven primarily by concerns of supporting and protecting the application's end-to-end communication, while the network-oriented "Flow+Endpoint" layer implements functions such as endpoint identification (using port numbers) and congestion control. These network-oriented functions, while traditionally located in the ostensibly "end-to-end" Transport layer, have proven in practice to be of great concern to network operators and the middleboxes they deploy in the network to enforce network usage policies [15] [16] or optimize communication performance [17]. Figure 5 shows how middleboxes interact with different layers in this decomposed model of the transport layer: the application-oriented layer operates end-to-end, while the network-oriented layer operates "segment-by-segment" and can be interposed upon by middleboxes.

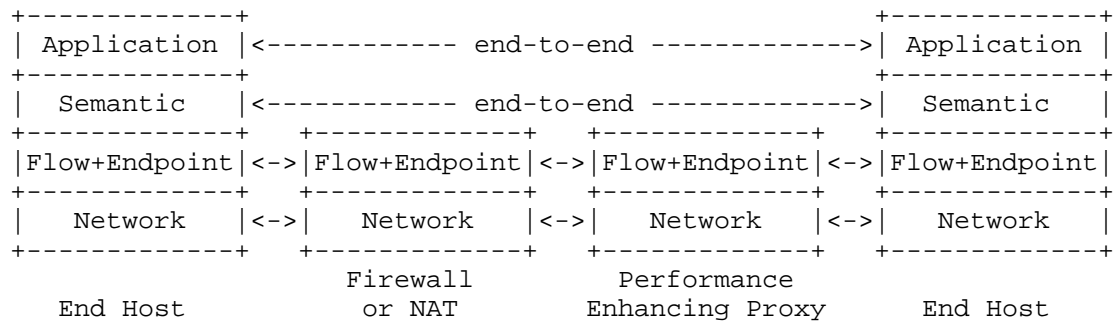


Figure 5: Middleboxes in the new Internet model

MPTCP's architectural design follows Tng's decomposition as shown in Figure 6. MPTCP, which provides application compatibility through the preservation of TCP-like semantics of global ordering of application data and reliability, is an instantiation of the

"application-oriented" Semantic layer; whereas the subflow TCP component, which provides network compatibility by appearing and behaving as a TCP flow in the network, is an instantiation of the "network-oriented" Flow+Endpoint layer.

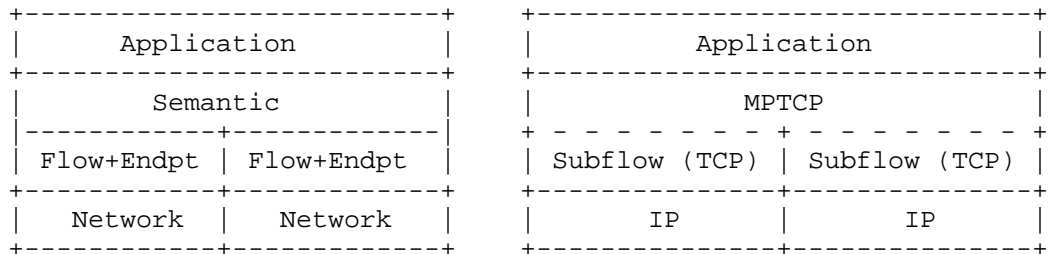


Figure 6: Relationship between Tng (left) and MPTCP (right)

As a protocol extension to TCP, MPTCP thus explicitly acknowledges middleboxes in its design, and specifies a protocol that operates at two scales: the MPTCP component operates end-to-end, while it allows the TCP component to operate segment-by-segment.

#### 4. A Functional Decomposition of MPTCP

The previous two sections have discussed the goals for a Multipath TCP design, and provided a basis for decomposing the functions of a transport protocol in order to better understand the form a solution should take. This section builds upon this analysis by presenting the functional components that are used within the MPTCP design.

MPTCP makes use of (what appear to the network to be) standard TCP sessions, termed "subflows", to provide the underlying transport per path, and as such these retain the network compatibility desired. MPTCP-specific information is carried in a TCP-compatible manner, although this mechanism is separate from the actual information being transferred so could evolve in future revisions. Figure 7 illustrates the layered architecture.

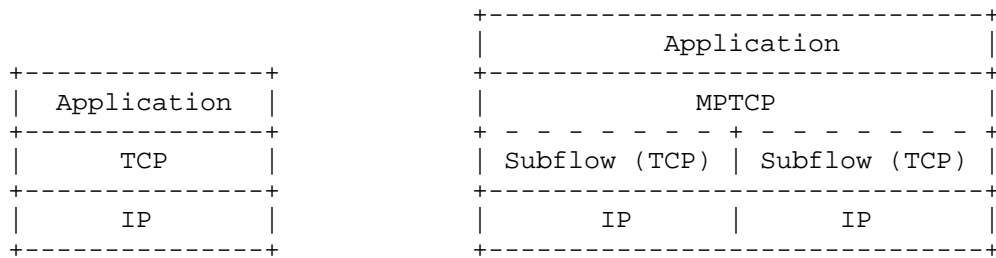


Figure 7: Comparison of Standard TCP and MPTCP Protocol Stacks

Situated below the application, the MPTCP extension in turn manages multiple TCP subflows below it. In order to do this, it must implement the following functions:

- o Path Management: This is the function to detect and use multiple paths between two hosts. MPTCP uses the presence of multiple IP addresses at one or both of the hosts as an indicator of this. The path management features of the MPTCP protocol are the mechanisms to signal alternative addresses to hosts, and mechanisms to set up new subflows joined to an existing MPTCP connection.
- o Packet Scheduling: This function breaks the bytestream received from the application into segments to be transmitted on one of the available subflows. The MPTCP design makes use of a data sequence mapping, associating segments sent on different subflows to a connection-level sequence numbering, thus allowing segments sent on different subflows to be correctly re-ordered at the receiver. The packet scheduler is dependent upon information about the availability of paths exposed by the path management component, and then makes use of the subflows to transmit queued segments. This function is also responsible for connection-level re-ordering on receipt of packets from the TCP subflows, according to the attached data sequence mappings.
- o Subflow (single-path TCP) Interface: A subflow component takes segments from the packet-scheduling component and transmits them over the specified path, ensuring detectable delivery to the host. MPTCP uses TCP underneath for network compatibility; TCP ensures in-order, reliable delivery. TCP adds its own sequence numbers to the segments; these are used to detect and retransmit lost packets at the subflow layer. On receipt, the subflow passes its reassembled data to the packet scheduling component for connection-level reassembly; the data sequence mapping from the sender's packet scheduling component allows re-ordering of the entire bytestream.

- o Congestion Control: This function coordinates congestion control across the subflows. As specified, this congestion control algorithm **MUST** ensure that a MPTCP connection does not unfairly take more bandwidth than a single path TCP flow would take at a shared bottleneck. An algorithm to support this is specified in [7].

These functions fit together as follows. The Path Management looks after the discovery (and if necessary, initialisation) of multiple paths between two hosts. The Packet Scheduler then receives a stream of data from the application destined for the network, and undertakes the necessary operations on it (such as segmenting the data into connection-level segments, and adding a connection-level sequence number) before sending it on to a subflow. The subflow then adds its own sequence number, ACKs, and passes them to network. The receiving subflow re-orders data (if necessary) and passes it to the packet scheduling component, which performs connection level re-ordering, and sends the data stream to the application. Finally, the congestion control component exists as part of the packet scheduling, in order to schedule which segments should be sent at what rate on which subflow.

## 5. High-Level Design Decisions

There is seemingly a wide range of choices when designing a multipath extension to TCP. However, the goals as discussed earlier in this document constrain the possible solutions, leaving relative little choice in many areas. Here, we outline high-level design choices that draw from the architectural basis discussed earlier in Section 3, which the design of MPTCP [5] takes into account.

### 5.1. Sequence Numbering

MPTCP uses two levels of sequence spaces: a connection level sequence number, and another sequence number for each subflow. This permits connection-level segmentation and reassembly, and retransmission of the same part of connection-level sequence space on different subflow-level sequence space.

The alternative approach would be to use a single connection level sequence number, which gets sent on multiple subflows. This has two problems: first, the individual subflows will appear to the network as TCP sessions with gaps in the sequence space; this in turn may upset certain middleboxes such as intrusion detection systems, or certain transparent proxies, and would thus go against the network compatibility goal. Second, the sender would not be able to attribute packet losses or receptions to the correct path when the

same segment is sent on multiple paths (i.e. in the case of retransmissions).

The sender must be able to tell the receiver how to reassemble the data, for delivery to the application. In order to achieve this, the receiver must determine how subflow-level data (carrying subflow sequence numbers) maps at the connection level. We refer to this as the Data Sequence Mapping. This mapping takes the form (data seq, subflow seq, length), i.e. for a given number of bytes (the length), the subflow sequence space beginning at the given sequence number maps to the connection-level sequence space (beginning at the given data seq number). This information could conceivably have various sources.

One option to signal the Data Sequence Mapping would be to use existing fields in the TCP segment (such as subflow seqno, length) and only add the data sequence number to each segment, for instance as a TCP option. This would be vulnerable, however, to middleboxes that resegment or assemble data, since there is no specified behaviour for coalescing TCP options. If one signalled (data seqno, length), this would still be vulnerable to middleboxes that coalesce segments and do not understand MPTCP signalling so do not correctly rewrite the options.

Because of these potential issues, the design decision taken in the MPTCP protocol is that whenever a mapping for subflow data needs to be conveyed to the other host, all three pieces of data (data seq, subflow seq, length) must be sent. To reduce the overhead, it would be permissible for the mapping to be sent periodically and cover more than a single segment. Further experimentation is required to determine what tradeoffs exist regarding the frequency at which mappings should be sent. It could also be excluded entirely in the case of a connection before more than one subflow is used, where the data-level and subflow-level sequence space is the same.

## 5.2. Reliability and Retransmissions

MPTCP features acknowledgements at connection-level as well as subflow-level acknowledgements, in order to provide a robust service to the application.

Under normal behaviour, MPTCP can use the data sequence mapping and subflow ACKs to decide when a connection-level segment was received. The transmission of TCP ACKs for a subflow are handled entirely at the subflow level, in order to maintain TCP semantics and trigger subflow-level retransmissions. This has certain implications on end-to-end semantics. It means that once a segment is ACKed at the subflow level it cannot be discarded in the re-order buffer at the

connection level. Secondly, unlike in standard TCP, a receiver cannot simply drop out-of-order segments if needed (for instance, due to memory pressure). Under certain circumstances, therefore, it may be desirable to drop segments after acknowledgement on the subflow but before delivery to the application, and this can be facilitated by a connection-level acknowledgement.

Furthermore, it is possible to conceive of some cases where connection-level acknowledgements could improve robustness. Consider a subflow traversing a transparent proxy: if the proxy ACKs a segment and then crashes, the sender will not retransmit the lost segment on another subflow, as it thinks the segment has been received. The connection grinds to a halt despite having other working subflows, and the sender would be unable to determine the cause of the problem. An example situation where this may occur would be mobility between wireless access points, each of which operates a transport-level proxy. Finally, as an optimisation, it may be feasible for a connection-level acknowledgement to be transmitted over the shortest Round-Trip Time (RTT) path, potentially reducing send buffer requirements (see Section 5.3).

Therefore, to provide a fully robust multipath TCP solution given the above constraints, MPTCP for use on the public Internet MUST feature explicit connection-level acknowledgements, in addition to subflow-level acknowledgements. A connection-level acknowledgement would only be required in order to signal when the receive window moves forward; the heuristics for using such a signal are discussed in more detail in the protocol specification [5].

Regarding retransmissions, it MUST be possible for a segments to be retransmitted on a different subflow to that on which it was originally sent. This is one of MPTCP's core goals, in order to maintain integrity during temporary or permanent subflow failure, and this is enabled by the dual sequence number space.

The scheduling of retransmissions will have significant impact on MPTCP user experience. The current MPTCP specification suggests that data outstanding on subflows that have timed out should be rescheduled for transmission on different subflows. This behaviour aims to minimize disruption when a path breaks, and uses the first timeout as indicators. More conservative versions would be to use second or third timeouts for the same segment.

Typically, fast retransmit on an individual subflow will not trigger retransmission on another subflow, although this may still be desirable in certain cases, for instance to reduce the receive buffer requirements. However, in all cases with retransmissions on different subflows, the lost segments SHOULD still be sent on the



path that lost them. This is currently believed to be necessary to maintain subflow integrity, as per the network compatibility goal. By doing this, some efficiency is lost, and it is unclear at this point what the optimal retransmit strategy is.

Large-scale experiments are therefore required in order to determine the most appropriate retransmission strategy, and recommendations will be refined once more information is available.

### 5.3. Buffers

To ensure in-order delivery, MPTCP must use a connection level receive buffer, where segments are placed until they are in order and can be read by the application.

In regular, single-path TCP, it is usually recommended to set the receive buffer to  $2 \times \text{BDP}$  (Bandwidth-Delay Product, i.e.  $\text{BDP} = \text{BW} \times \text{RTT}$ , where  $\text{BW}$  = Bandwidth and  $\text{RTT}$  = Round-Trip Time). One BDP allows supporting reordering of segments by the network. The other BDP allows the connection to continue during fast retransmit: when a segment is fast retransmitted, the receiver must be able to store incoming data during one more RTT.

For MPTCP, the story is a bit more complicated. The ultimate goal is that a subflow packet loss or subflow failure should not affect the throughput of other working subflows; the receiver should have enough buffering to store all data until the missing segment is retransmitted and reaches the destination.

The worst case scenario would be when the subflow with the highest RTT/RTO (Round-Trip Time or Retransmission TimeOut) experiences a timeout; in that case the receiver has to buffer data from all subflows for the duration of the RTO. Thus, the smallest connection-level receive buffer that would be needed to avoid stalling with subflow failures is  $\text{sum}(\text{BW}_i) \times \text{RTO\_max}$ , where  $\text{BW}_i$  = Bandwidth for each subflow and  $\text{RTO\_max}$  is the largest RTO across all subflows.

This is an order of magnitude more than the receive buffer required for a single connection, and is probably too expensive for practical purposes. A more sensible requirement is to avoid stalls in the absence of timeouts. Therefore, the RECOMMENDED receive buffer is  $2 \times \text{sum}(\text{BW}_i) \times \text{RTT\_max}$ , where  $\text{RTT\_max}$  is the largest RTT across all subflows. This buffer sizing ensures subflows do not stall when fast retransmit is triggered on any subflow.

The resulting buffer size should be small enough for practical use. However, there may be extreme cases where fast, high throughput paths (e.g. 100Mb/s, 10ms RTT) are used in conjunction with slow paths

(e.g. 1Mb/s, 1000ms RTT). In that case the required receive buffer would be 12.5MB, which is likely too big. In extreme cases such as this example, it may be prudent to only use some of the fastest available paths for the MPTCP connection, potentially using the slow path(s) for backup only.

**Send Buffer:** The RECOMMENDED send buffer is the same size as the recommended receive buffer i.e.,  $2 * \sum(BW_i) * RTT_{max}$ . This is because the sender must store locally the segments sent but unacknowledged by the connection level ACK. The send buffer size matters particularly for hosts that maintain a large number of ongoing connections. If the required send buffer is too large, a host can choose to only send data on the fast subflows, using the slow subflows only in cases of failure.

#### 5.4. Signalling

Since MPTCP uses TCP as its subflow transport mechanism, a MPTCP connection will also begin as a single TCP connection. Nevertheless, it must signal to the peer that it supports MPTCP and wishes to use it on this connection. As such, a TCP Option will be used to transmit this information, since this is the established mechanism for indicating additional functionality on a TCP session.

In addition, further signalling is required during the operation of a MPTCP session, such as that for reassembly for multiple subflows, and for informing the other host about potential other available addresses.

The MPTCP protocol design will, however, use TCP Options for this additional signalling. This has been chosen as the mechanism most fitting in with the goals as specified in Section 2. With this mechanism, the signalling required to operate MPTCP is transported separately from the data, allowing it to be created and processed separately from the data stream, and retaining architectural compatibility with network entities.

This decision is the consensus of the Working Group (following detailed discussions at IETF78), and the main reasons for this are as follows:

- o TCP options are the traditional signalling method for TCP;
- o A TCP option on a SYN is the most compatible way for an end host to signal it is MPTCP-capable;
- o If connection-level ACKs are signalled in the payload then they may suffer from packet loss and may be congestion-controlled,

which may affect the data throughput in the forward direction and could lead to head-of-line blocking;

- o Middleboxes, such as NAT traversal helpers, can easily parse TCP options, e. g., to rewrite addresses.

On the other hand, the main drawbacks of TCP options compared to TLV encoding in the payload are:

- o There is limited space for signalling messages;
- o A middlebox may, potentially, drop a packet with an unknown option;
- o The transport of control information in options is not necessarily reliable.

The detailed design of MPTCP alleviates these issues as far as possible by carefully considering the size of MPTCP options, and seamlessly falling back to regular TCP on the loss of control data.

Both option and payload encoding may interfere with offloading of TCP processing to high speed network interface cards, such as segmentation, checksumming, and reassembly. For network cards supporting MPTCP, signalling in TCP options should simplify offloading due to the separate handling of MPTCP signalling and data.

#### 5.5. Path Management

Currently, the network does not expose path diversity between pairs of IP addresses. In order to achieve path diversity from today's IP networks, in the typical case MPTCP uses multiple addresses at one or both hosts to infer different paths across the network. It is expected that these paths, whilst not necessarily entirely non-overlapping, will be sufficiently disjoint to allow multipath to achieve improved throughput and robustness. The use of multiple IP addresses is a simple mechanism that requires no additional features in the network.

Multiple different (source, destination) address pairs will thus be used as path selectors in most cases. Each path will be identified by a standard five-tuple (i.e. source address, destination address, source port, destination port, protocol), however, which can allow the extension of MPTCP to use ports as well as addresses as path selectors. This will allow hosts to use port-based load balancing with MPTCP, for example if the network routes different ports over different paths (which may be the case with technologies such as Equal Cost MultiPath (ECMP) routing [4]). It should be noted,

however, that ISPs often undertake traffic engineering in order to optimise resource utilisation within their networks, and care should be taken (by both ISPs and developers) that MPTCP using broadly similar paths does not adversely interfere with this.

For increased chance of successfully setting up additional subflows (such as when one end is behind a firewall, NAT, or other restrictive middlebox), either host SHOULD be able to add new subflows to a MPTCP connection. MPTCP MUST be able to handle paths that appear and disappear during the lifetime of a connection (for example, through the activation of an additional network interface).

The path management is a separate function from the packet scheduling, subflow interface, and congestion control functions of MPTCP, as documented in Section 4. As such it would be feasible to replace this IP-address-based design with an alternative path selection mechanism in the future, with no significant changes to the other functional components.

#### 5.6. Connection Identification

Since a MPTCP connection may not be bound to a traditional 5-tuple (source address and port, destination address and port, protocol number) for the entirety of its existence, it is desirable to provide a new mechanism for connection identification. This will be useful for MPTCP-aware applications, and for the MPTCP implementation (and MPTCP-aware middleboxes) to have a unique identifier with which to associate the multiple subflows.

Therefore, each MPTCP connection requires a connection identifier at each host, which is locally unique within that host. In many ways, this is analogous to an ephemeral port number in regular TCP. The manifestation and purpose of such an identifier is out of the scope of this architecture document.

Legacy applications will not, however, have access to this identifier and in such cases a MPTCP connection will be identified by the 5-tuple of the first TCP subflow. It is out of the scope of this document, however, to define the behaviour of the MPTCP implementation if the first TCP subflow later fails. If there are MPTCP-unaware applications that make assumptions about continued existence of the initial address pair, their behaviour could be disrupted by carrying on regardless. It is expected that this is a very small, possibly negligible, set of applications, however. MPTCP MUST NOT be used for applications that request to bind to a specific address or interface, since such applications are making a deliberate choice of path in use.

Since the requirements of applications are not clear at this stage, however, it is as yet unconfirmed whether carrying on in the event of the loss of the initial address pair would be a damaging assumption to make. This behaviour will be an implementation-specific solution, and as such it is expected to be chosen by implementors once more research has been undertaken to determine its impact.

#### 5.7. Congestion Control

As discussed in network-layer compatibility requirements Section 2.2.3, there are three goals for the congestion control algorithms used by a MPTCP implementation: improve throughput (at least as well as a single-path TCP connection would perform); do no harm to other network users (do not take up more capacity on any one path than if it was a single path flow using only that route - this is particularly relevant for shared bottlenecks); and balance congestion by moving traffic away from the most congested paths. To achieve these goals, the congestion control algorithms on each subflow must be coupled in some way. A proposal for a suitable congestion control algorithm is given in [7].

#### 5.8. Security

A detailed threat analysis for Multipath TCP is presented in a separate document [12]. This focuses on flooding attacks and hijacking attacks that can be launched against a Multipath TCP connection.

The basic security goal of Multipath TCP, as introduced in Section 2.3, can be stated as: "provide a solution that is no worse than standard TCP".

From the threat analysis, and with this goal in mind, three key security requirements can be identified. A multi-addressed Multipath TCP SHOULD be able to:

- o Provide a mechanism to confirm that the parties in a subflow handshake are the same as in the original connection setup (e.g. require use of a key exchanged in the initial handshake in the subflow handshake, to limit the scope for hijacking attacks).
- o Provide verification that the peer can receive traffic at a new address before adding it (i.e. verify that the address belongs to the other host, to prevent flooding attacks).
- o Provide replay protection, i.e. ensure that a request to add/remove a subflow is 'fresh'.

Additional mechanisms have been deployed as part of standard TCP stacks to provide resistance to Denial-of-Service attacks. For example, there are various mechanisms to protect against TCP reset attacks [18], and Multipath TCP should continue to support similar protection. In addition, TCP SYN Cookies [19] were developed to allow a TCP server to defer the creation of session state in the SYN\_RCVD state, and remain stateless until the ESTABLISHED state had been reached. Multipath TCP should, ideally, continue to provide such functionality and, at a minimum, avoid significant computational burden prior to reaching the ESTABLISHED state (of the Multipath TCP connection as a whole).

It should be noted that aspects of the Multipath TCP design space place constraints on the security solution:

- o The use of TCP options significantly limits the amount of information that can be carried in the handshake.
- o The need to work through middleboxes results in the need to handle mutability of packets.
- o The desire to support a 'break-before-make' (as well as a 'make-before-break') approach to adding subflows (within a limited time period) implies that a host cannot rely on using a pre-existing subflow to support the addition of a new one.

The MPTCP protocol will be designed with these security requirements in mind, and the protocol specification [5] will document how these are met.

## 6. Software Interactions

### 6.1. Interactions with Applications

In the case of applications that have used an existing API call to bind to a specific address or interface, the MPTCP extension MUST NOT be used. This is because the applications are indicating a clear choice of path to use and thus will have expectations of behaviour that must be maintained, in order to adhere to the application compatibility goals.

Interactions with applications are presented in [8] - including, but not limited to, performance changes that may be expected, semantic changes, and new features that may be requested through an enhanced API.

TCP features the ability to send "Urgent" data, the delivery of which

to the application may or may not be out-of-band. The use of this feature is not recommended due to security implications and implementation differences [20]. MPTCP requires contiguous data to support its Data Sequence Mapping over multiple segments, and therefore the Urgent pointer cannot interrupt an existing mapping. An MPTCP implementation MAY choose to support sending Urgent data, and if it does, it SHOULD send the Urgent data on the soonest available unassigned subflow sequence space. Incoming Urgent data SHOULD be mapped to connection-level sequence space and delivered to the application analogous to Urgent data in regular TCP.

## 6.2. Interactions with Management Systems

To enable interactions between TCP and network management systems, the TCP [21] and TCP Extended Statistics (ESTATS) [22] MIBs have been defined. MPTCP should share these MIBs for aspects that are designed to be transparent to the application.

It is anticipated that a MPTCP MIB will be defined in the future, once experience of experimental MPTCP deployments is gathered. This MIB would provide access to MPTCP-specific properties such as whether MPTCP is enabled, and the number and properties of the individual paths in use.

## 7. Interactions with Middleboxes

As discussed in Section 2.2, it is a goal of MPTCP to be deployable today and thus compatible with the majority of middleboxes. This section summarises the issues that may arise with NATs, firewalls, proxies, intrusion detection systems, and other middleboxes that, if not considered in the protocol design, may hinder its deployment.

This section is intended primarily as a description of options and considerations only. Protocol-specific solutions to these issues will be given in the companion documents.

Multipath TCP will be deployed in a network that no longer provides just basic datagram delivery. A myriad of middleboxes are deployed to optimize various perceived problems with the Internet protocols: NATs primarily address IP address space shortage [15], Performance Enhancing Proxies (PEPs) optimize TCP for different link characteristics [17], firewalls [16] and intrusion detection systems try to block malicious content from reaching a host, and traffic normalizers [23] ensure a consistent view of the traffic stream to Intrusion Detection Systems (IDS) and hosts.

All these middleboxes optimize current applications at the expense of

future applications. In effect, future applications will often need to behave in a similar fashion to existing ones, in order to increase the chances of successful deployment. Further, the precise behaviour of all these middleboxes is not clearly specified, and implementation errors make matters worse, raising the bar for the deployment of new technologies.

The following list of middlebox classes documents behaviour that could impact the use of MPTCP. This list is used in [5] to describe the features of the MPTCP protocol that are used to mitigate the impact of these middlebox behaviours.

- o NATs: Network Address Translators decouple the host's local IP address (and, in the case of NATs, port) with that which is seen in the wider Internet when the packets are transmitted through a NAT. This adds complexity, and reduces the chances of success, when signalling IP addresses.
- o PEPs: Performance Enhancing Proxies, which aim to improve the performance of protocols over low-performance (e.g. high latency or high error rate) links. As such, they may "split" a TCP connection and behaviour such as proactive ACKing may occur, and therefore it is no longer guaranteed that one host is communicating directly with another. PEPs, firewalls or other middleboxes may also change the declared receive window size.
- o Traffic Normalizers: These aim to eliminate ambiguities and potential attacks at the network level, and amongst other things are unlikely to permit holes in TCP-level sequence space (which has impact on MPTCP's retransmission and subflow sequence numbering design choices).
- o Firewalls: on top of preventing incoming connections, firewalls may also attempt additional protection such as sequence number randomization (so a sender cannot reliably know what TCP sequence number the receiver will see).
- o Intrusion Detection Systems: IDSs may look for traffic patterns to protect a network, and may have false positives with MPTCP and drop the connections during normal operation. Future MPTCP-aware middleboxes will require the ability to correlate the various paths in use.
- o Content-aware Firewalls: Some middleboxes may actively change data in packets, such as re-writing URIs in HTTP traffic.

In addition, all classes of middleboxes may affect TCP traffic in the following ways:



- o TCP Options: some middleboxes may drop packets with unknown TCP options, or strip those options from the packets.
- o Segmentation and Coalescing: middleboxes (or even something as close to the end host as TCP Segmentation Offloading (TSO) on a Network Interface Card (NIC)) may change the packet boundaries from those which the sender intended. It may do this by splitting packets, or coalescing them together. This leads to two major impacts: we cannot guarantee where a packet boundary will be, and we cannot say for sure what a middlebox will do with TCP options in these cases (they may be repeated, dropped, or sent only once).

## 8. Contributors

The authors would like to acknowledge the contributions of Andrew McDonald and Bryan Ford to this document.

The authors would also like to thank the following people for detailed reviews: Olivier Bonaventure, Gorrry Fairhurst, Iljitsch van Beijnum, Philip Eardley, Michael Scharf, Lars Eggert, Cullen Jennings, Joel Halpern, Juergen Quittek, Alexey Melnikov, David Harrington, Jari Arkko and Stewart Bryant.

## 9. Acknowledgements

Alan Ford, Costin Raiciu, Mark Handley, and Sebastien Barre are supported by Trilogy (<http://www.trilogy-project.org>), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Program. The views expressed here are those of the author(s) only. The European Commission is not liable for any use that may be made of the information in this document.

## 10. IANA Considerations

None.

## 11. Security Considerations

This informational document provides an architectural overview for Multipath TCP and so does not, in itself, raise any security issues. A separate threat analysis [12] lists threats that can exist with a Multipath TCP. However, a protocol based on the architecture in this document will have a number of security requirements. The high level goals for such a protocol are identified in Section 2.3, whilst

Section 5.8 provides more detailed discussion of security requirements and design decisions which are applied in the MPTCP protocol design [5].

## 12. References

### 12.1. Normative References

- [1] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 12.2. Informative References

- [3] Wischik, D., Handley, M., and M. Bagnulo Braun, "The Resource Pooling Principle", ACM SIGCOMM CCR vol. 38 num. 5, pp. 47-52, October 2008, <<http://ccr.sigcomm.org/online/files/p47-handleyA4.pdf>>.
- [4] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, November 2000.
- [5] Ford, A., Raiciu, C., and M. Handley, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-02 (work in progress), October 2010.
- [6] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [7] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", draft-ietf-mptcp-congestion-01 (work in progress), January 2011.
- [8] Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-ietf-mptcp-api-00 (work in progress), November 2010.
- [9] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.
- [10] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

- [11] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.
- [12] Bagnulo, M., "Threat Analysis for TCP Extensions for Multi-path Operation with Multiple Addresses", draft-ietf-mptcp-threat-07 (work in progress), January 2011.
- [13] Becke, M., Dreibholz, T., Iyengar, J., Natarajan, P., and M. Tuexen, "Load Sharing for the Stream Control Transmission Protocol (SCTP)", draft-tuexen-tsvwg-sctp-multipath-01 (work in progress), December 2010.
- [14] Ford, B. and J. Iyengar, "Breaking Up the Transport Logjam", ACM HotNets, October 2008.
- [15] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [16] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, October 2000.
- [17] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, June 2001.
- [18] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, August 2010.
- [19] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.
- [20] Gont, F. and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", RFC 6093, January 2011.
- [21] Raghunarayan, R., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, March 2005.
- [22] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.
- [23] Handley, M., Paxson, V., and C. Kreibich, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", Usenix Security 2001, 2001, <[http://www.usenix.org/events/sec01/full\\_papers/handley/handley.pdf](http://www.usenix.org/events/sec01/full_papers/handley/handley.pdf)>.

## Appendix A. Changelog

(For removal by the RFC Editor)

## A.1. Changes since draft-ietf-mptcp-architecture-04

- o Responded to IETF Last Call and IESG review comments.

## A.2. Changes since draft-ietf-mptcp-architecture-03

- o Responded to AD review comments.

## A.3. Changes since draft-ietf-mptcp-architecture-02

- o Responded to WG last call review comments. Included editorial fixes, adding Section 2.4, and improving Section 5.4 and Section 7.

## A.4. Changes since draft-ietf-mptcp-architecture-01

- o Responded to review comments.
- o Added security sections.

## A.5. Changes since draft-ietf-mptcp-architecture-00

- o Added middlebox compatibility discussion (Section 7).
- o Clarified path identification (TCP 4-tuple) in Section 5.5.
- o Added brief scenario and diagram to Section 1.3.

## Authors' Addresses

Alan Ford  
Roke Manor Research  
Old Salisbury Lane  
Romsey, Hampshire SO51 0ZN  
UK

Phone: +44 1794 833 465  
Email: alan.ford@roke.co.uk

Costin Raiciu  
University College London  
Gower Street  
London WC1E 6BT  
UK

Email: c.raiciu@cs.ucl.ac.uk

Mark Handley  
University College London  
Gower Street  
London WC1E 6BT  
UK

Email: m.handley@cs.ucl.ac.uk

Sebastien Barre  
Universite catholique de Louvain  
Pl. Ste Barbe, 2  
Louvain-la-Neuve 1348  
Belgium

Phone: +32 10 47 91 03  
Email: sebastien.barre@uclouvain.be

Janardhan Iyengar  
Franklin and Marshall College  
Mathematics and Computer Science  
PO Box 3003  
Lancaster, PA 17604-3003  
USA

Phone: 717-358-4774  
Email: jiyengar@fandm.edu



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: July 30, 2011

M. Bagnulo  
UC3M  
January 26, 2011

Threat Analysis for TCP Extensions for Multi-path Operation with  
Multiple Addresses  
draft-ietf-mptcp-threat-08

Abstract

Multi-path TCP (MPTCP for short) describes the extensions proposed for TCP so that endpoints of a given TCP connection can use multiple paths to exchange data. Such extensions enable the exchange of segments using different source-destination address pairs, resulting in the capability of using multiple paths in a significant number of scenarios. Some level of multihoming and mobility support can be achieved through these extensions. However, the support for multiple IP addresses per endpoint may have implications on the security of the resulting MPTCP protocol. This note includes a threat analysis for MPTCP.

Status of this Memo

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

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

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

This Internet-Draft will expire on July 30, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Scope . . . . .	3
3. Related work . . . . .	4
4. Basic MPTCP. . . . .	6
5. Flooding attacks . . . . .	7
6. Hijacking attacks . . . . .	10
6.1. Hijacking attacks to the Basic MPTCP protocol . . . . .	10
6.2. Time-shifted hijacking attacks . . . . .	12
6.3. NAT considerations . . . . .	14
7. Recommendation . . . . .	15
8. Security Considerations . . . . .	15
9. IANA Considerations . . . . .	16
10. Contributors . . . . .	16
11. Acknowledgments . . . . .	16
12. References . . . . .	16
12.1. Normative References . . . . .	16
12.2. Informative References . . . . .	16
Author's Address . . . . .	17



## 1. Introduction

Multi-path TCP (MPTCP for short) describes the extensions proposed for TCP [RFC0793] so that endpoints of a given TCP connection can use multiple paths to exchange data. Such extensions enable the exchange of segments using different source-destination address pairs, resulting in the capability of using multiple paths in a significant number of scenarios. Some level of multihoming and mobility support can be achieved through these extensions. However, the support for multiple IP addresses per endpoint may have implications on the security of the resulting MPTCP protocol. This note includes a threat analysis for MPTCP. There are there may other ways to provide multiple paths for a TCP connection other than the usage of multiple addresses. The threat analysis performed in this document is limited to the specific case of using multiple addresses per endpoint.

## 2. Scope

There are multiple ways to achieve Multi-path TCP. Essentially what is needed is for different segments of the communication to be forwarded through different paths by enabling the sender to specify some form of path selector. There are multiple options for such a path selector, including the usage of different next hops, using tunnels to different egress points and so on. In this note, we will focus on a particular approach, namely MPTCP, that rely on the usage of multiple IP address per endpoint and that uses different source-destination address pairs as a mean to express different paths. So, in the rest of this note, the MPTCP expression will refer to this Multi-addressed flavour of Multi-path TCP [I-D.ietf-mptcp-multiaddressed].

In this note we perform a threat analysis for MPTCP. Introducing the support of multiple addresses per endpoint in a single TCP connection may result in additional vulnerabilities compared to single-path TCP. The scope of this note is to identify and characterize these new vulnerabilities. So, the scope of the analysis is limited to the additional vulnerabilities resulting from the multi-address support compared to the current TCP protocol (where each endpoint only has one address available for use per connection). A full analysis of the complete set of threats is explicitly out of the scope. The goal of this analysis is to help the MPTCP protocol designers create an MPTCP specification that is as secure as the current TCP. It is a non-goal of this analysis to help in the design of MPTCP that is more secure than regular TCP.

We will focus on attackers that are not along the path, at least not during the whole duration of the connection. In the current single

path TCP, an on-path attacker can launch a significant number of attacks, including eavesdropping, connection hijacking Man-in-the-Middle attacks and so on. However, it is not possible for the off-path attackers to launch such attacks. There is a middle ground in case the attacker is located along the path for a short period of time to launch the attack and then moves away, but the attack effects still apply. These are the so-called time-shifted attacks. Since these are not possible in today's TCP, we will also consider them as part of the analysis. So, summarizing, we will consider both attacks launched by off-path attackers and time-shifted attacks. Attacks launched by on-path attackers are out of scope, since they also apply to current single-path TCP.

However, that some current on-path attacks may become more difficult with multi-path TCP, since an attacker (on a single path) will not have visibility of the complete data stream.

### 3. Related work

There is a significant amount of previous work in terms of analysis of protocols that support address agility. In this section we present the most relevant ones and we relate them to the current MPTCP effort.

Most of the problems related to address agility have been deeply analyzed and understood in the context of Route Optimization support in Mobile IPv6 (MIPv6 RO) [RFC3775]. [RFC4225] includes the rationale for the design of the security of MIPv6 RO. All the attacks described in the aforementioned analysis apply here and are an excellent basis for our own analysis. The main differences are:

- o In MIPv6 RO, the address binding affects all the communications involving an address, while in the MPTCP case, a single connection is at stake. If a binding between two addresses is created at the IP layer, this binding can and will affect all the connections that involve those addresses. However, in MPTCP, if an additional address is added to an ongoing TCP connection, the additional address will/can only affect the connection at hand and not other connections even if the same address is being used for those other connections. The result is that in MPTCP there is much less at stake and the resulting vulnerabilities are less. On the other hand, it is very important to keep the assumption valid that the address bindings for a given connection do not affect other connections. If reusing of binding or security information is to be considered, this assumption could be no longer valid and the full impact of the vulnerabilities must be assessed.

- o In MIPv6 there is a trusted third party, called the Home Agent that can help with some security problems, as expanded in the next bullet.
- o In MIPv6 RO, there is the assumption that the original address (Home Address) through which the connection has been established is always available and in case it is not, the communication will be lost. This is achieved by leveraging in the on the trusted party (the Home Agent) to rely the packets to the current location of the Mobile Node. In MPTCP, it is an explicit goal to provide communication resilience when one of the address pairs is no longer usable, so it is not possible to leverage on the original address pair to be always working.
- o MIPv6 RO is of course designed for IPv6 and it is an explicit goal of MPTCP to support both IPv6 and IPv4. Some MIPv6 RO security solutions rely on the usage of some characteristics of IPv6 (such as the usage of CGAs [RFC3972]), which will not be usable in the context of MPTCP.
- o As opposed to MPTCP, MIPv6 RO does not have a connection state information, including sequence numbers, port numbers that could be leveraged to provide security in some form.

In the Shim6 [RFC5533] design, similar issues related to address agility were considered and a threat analysis was also performed [RFC4218]. The analysis performed for Shim6 also largely applies to the MPTCP context, the main difference being:

- o The Shim6 protocol is a layer 3 protocol so all the communications involving the target address are at stake; in MPTCP, the impact can be limited to a single TCP connection.
- o Similarly to MIPv6 RO, Shim6 only uses IPv6 addresses as identifiers and leverages on some of their properties to provide the security, such as relying on CGAs or HBAs [RFC5535], which is not possible in the MPTCP case where IPv4 addresses must be supported.
- o Similarly to MIPv6 RO, Shim6 does not have a connection state information, including sequence numbers, port that could be leveraged to provide security in some form.

SCTP [RFC4960] is a transport protocol that supports multiple addresses per endpoint and the security implications are very close to the ones of MPTCP. A security analysis, identifying a set of attacks and proposed solutions was performed in [RFC5062]. The results of this analysis apply directly to the case of MPTCP. However, the analysis was performed after the base SCTP protocol was designed and the goal of the document was essentially to improve the security of SCTP. As such, the document is very specific to the actual SCTP specification and relies on the SCTP messages and behaviour to characterize the issues. While some them can be translated to the MPTCP case, some may be caused by specific

behaviour of SCTP.

So, the conclusion is that while we do have a significant amount of previous work that is closely related and we can and will use it as a basis for this analysis, there is a set of characteristics that are specific to MPTCP that grant the need for a specific analysis for MPTCP. The goal of this analysis is to help MPTCP protocol designers to include a set of security mechanisms that prevent the introduction of new vulnerabilities to the Internet due to the adoption of MPTCP.

#### 4. Basic MPTCP.

The goal of this document is to serve as input for MPTCP protocol designers to properly take into account the security issues. As such, the analysis cannot be performed for a specific MPTCP specification, but must be a general analysis that applies to the widest possible set of MPTCP designs. We will characterize what are the fundamental features that any MPTCP protocol must provide and attempt to perform the security implications only assuming those. In some cases, we will have a design choice that will significantly influence the security aspects of the resulting protocol. In that case we will consider both options and try to characterize both designs.

We assume that any MPTCP will behave in the case of a single address per endpoint as TCP. This means that a MPTCP connection will be established by using the TCP 3-way handshake and will use a single address pair.

The addresses used for the establishment of the connection do have a special role in the sense that this is the address used as identifier by the upper layers. The address used as destination address in the SYN packet is the address that the application is using to identify the peer and has been obtained either through the DNS (with or without DNSSEC validation) or passed by a referral or manually introduced by the user. As such, the initiator does have a certain amount of trust in the fact that it is establishing a communication with that particular address. If due to MPTCP, packets end up being delivered to an alternative address, the trust that the initiator has placed on that address would be deceived. In any case, the adoption of MPTCP necessitates a slight evolution of the traditional TCP trust model, in that the initiator is additionally trusting the peer to provide additional addresses which it will trust to the same degree as the original pair. An application or implementation that cannot trust the peer in this way should not make use of multiple paths.

During the 3-way handshake, the sequence number will be synchronized

for both ends, as in regular TCP. We assume that a MPTCP connection will use a single sequence number for the data, even if the data is exchanged through different paths, as MPTCP provides an in-order delivery service of bytes

Once the connection is established, the MPTCP extensions can be used to add addresses for each of the endpoints. This is achieved by each end sending a control message containing the additional address(es). In order to associate the additional address to an ongoing connection, the connection needs to be identified. We assume that the connection can be identified by the 4-tuple of source address, source port, destination address, destination port used for the establishment of the connection. So, at least, the control message that will convey the additional address information can also contain the 4-tuple in order to inform about what connection the address belong to (if no other connection identifier is defined). There are two different ways to convey address information:

- o Explicit mode: the control message contain a list of addresses.
- o Implicit mode: the address added is the one included in the source address field of the IP header

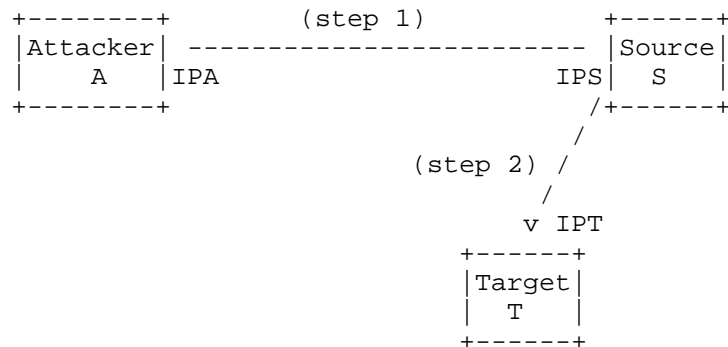
These two modes have different security properties for some type of attacks. The explicit mode seems to be the more vulnerable to abuse. The implicit mode may benefit from forms of ingress filtering security, which would reduce the possibility of an attacker to add any arbitrary address to an ongoing connection. However, ingress filtering deployment is far from universal, and it is unwise to rely on it as a basis for the protection of MPTCP.

Further consideration about the interaction between ingress filtering and implicit mode signaling is needed in the case that we need to remove an address that is no longer available from the MPTCP connection. A host attached to a network that performs ingress filtering and using implicit signaling would not be able to remove an address that is no longer available (either because of a failure or due to a mobility event) from an ongoing MPTCP connection.

We will assume that MPTCP will use all the address pairs that it has available for sending packets and that it will distribute the load based on congestion among the different paths.

## 5. Flooding attacks

The first type of attacks that are introduced by address agility are the flooding (or bombing) attacks. The setup for this attack is depicted in the following figure:



The scenario consists of an attacker A who has an IP address IPA. A server that can generate a significant amount of traffic (such as a streaming server), called source S and that has IP address IPS. Target T has an IP address IPT.

In step 1 of this attack, the attacker A establishes a MPTCP connection with the source of the traffic server S and starts downloading a significant amount of traffic. The initial connection only involves one IP address per endpoint, IPA and IPS. Once that the download is on course, in the step 2 of the attack is that the attacker A adds IPT as one of the available addresses for the communication. How the additional address is added depends on the MPTCP address management mode. In explicit address management, the attacker A only needs to send a signaling packet conveying address IPT. In implicit mode, the attacker A would need to send a packet with IPT as the source address. Depending on whether ingress filtering is deployed and the location of the attacker, it may be possible or not for the attacker to send such a packet. At this stage, the MPTCP connection still has a single address for the Source S i.e. IPS but has two addresses for the Attacker A, IPA and IPT. The attacker now attempts to get the Source S to send the traffic of the ongoing download to the Target T IP address i.e. IPT. The attacker can do that by pretending that the path between IPA and IPT is congested but that the path between IPS and IPT is not. In order to do that, it needs to send ACKs for the data that flows through the path between IPS and IPT and do not send ACKs for the data that is sent to IPA. The details of this will depend on how the data sent through the different paths is ACKed. One possibility is that ACKs for the data sent using a given address pair should come in packets containing the same address pair. If so, the attacker would need to send ACKs using packets containing IPT as the source address to keep the attack flowing. This may be possible or not depending on the deployment of ingress filtering and the location of the attacker. The attacker would also need to guess the sequence number of the data

being sent to the Target. Once the attacker manages to perform these actions the attack is on place and the download will hit the target. In this type of attacks, the Source S still thinks it is sending packets to the Attacker A while in reality it is sending the packet to Target T.

Once that the traffic from the Source S start hitting the Target T, the target will react. Since the packets are likely to belong to a non existent TCP connection, the Target T will issue RST packets. It is relevant then to understand how MPTCP reacts to incoming RST packets. It seems that the at least the MPTCP that receives a RST packet should terminate the packet exchange corresponding to the particular address pair (maybe not the complete MPTCP connection, but at least it should not send more packets with the address pair involved in the RST packet). However, if the attacker, before redirecting the traffic has managed to increase the window size considerably, the flight size could be enough to impose a significant amount of traffic to the Target node. There is a subtle operation that the attacker needs to achieve in order to launch a significant attack. On the one hand it needs to grow the window enough so that the flight size is big enough to cause enough effect and on the other hand the attacker needs to be able to simulate congestion on the IPA-IPS path so that traffic is actually redirected to the alternative path without significantly reducing the window. This will heavily depend on how the coupling of the windows between the different paths works, in particular how the windows are increased. Some designs of the congestion control window coupling could render this attack ineffective. If the MPTCP protocol requires performing slow start per subflow, then the flooding will be limited by the slow-start initial window size.

Previous protocols, such as MIPv6 RO and SCTP, that have to deal with this type of attacks have done so by adding a reachability check before actually sending data to a new address. The solution used in other protocols, would include the Source S to explicitly asking the host sitting in the new address (the Target T sitting in IPT) whether it is willing to accept packets from the MPTCP connection identified by the 4-tuple IPA, port A, IPS, port S. Since this is not part of the established connection that Target T has, T would not accept the request and Source S would not use IPT to send packets for this MPTCP connection. Usually, the request also includes a nonce that cannot be guessed by the attacker A so that it cannot fake the reply to the request easily. In the case of SCTP, it sends a message with a 64-bit nonce (in a HEARTBEAT).

One possible approach to do this reachability test would be to perform a 3-way handshake for each new address pair that is going to be used in a MPTCP connection. While there are other reasons for

doing this (such as NAT traversal), such approach would also act as a reachability test and would prevent the flooding attacks described in this section.

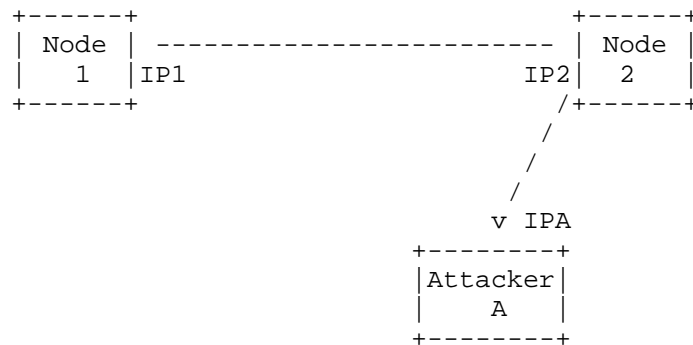
Another type of flooding attack that could potentially be performed with MPTCP is one where the attacker initiates a communication with a peer and includes a long list of alternative addresses in explicit mode. If the peer decides to establish subflows with all the available addresses, the attacker have managed to achieve an amplified attack, since by sending a single packet containing all the alternative addresses it triggers the peer to generate packets to all the destinations.

## 6. Hijacking attacks

### 6.1. Hijacking attacks to the Basic MPTCP protocol

The hijacking attacks essentially use the MPTCP address agility to allow an attacker to hijack a connection. This means that the victim of a connection thinks that it is talking to a peer, while it is actually exchanging packets with the attacker. In some sense it is the dual of the flooding attacks (where the victim thinks it is exchanging packets with the attacker but in reality is sending the packets to the target).

The scenario for a hijacking attack is described in the next figure.



We have a MPTCP connection established between Node 1 and Node 2. The connection is using only one address per endpoint, IP1 and IP2. The attacker then launches the hijacking attack by adding IPA as an additional address for Node 1. There is not much difference between

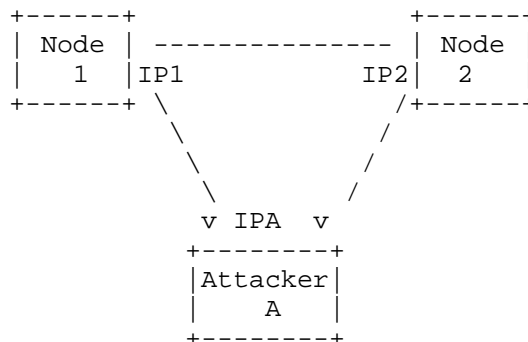


explicit or implicit address management, since in both cases the Attacker A could easily send a control packet adding the address IPA, either as control data or as the source address of the control packet. In order to be able to hijack the connection, the attacker needs to know the 4-tuple that identifies the connection, including the pair of addresses and the pair of ports. It seems reasonable to assume that knowing the source and destination IP addresses and the port of the server side is fairly easy for the attacker. Learning the port of the client (i.e. of the initiator of the connection) may prove to be more challenging. The attacker would need to guess what the port is or to learn it by intercepting the packets. Assuming that the attacker can gather the 4-tuple and issue the message adding IPA to the addresses available for the MPTCP connection, then the attacker A has been able to participate in the communication. In particular:

- o Segments flowing from the Node 2: Depending how the usage of addresses is defined, Node 2 will start using IPA to send data to. In general, since the main goal is to achieve multi-path capabilities, we can assume that unless there are already many IP address pairs in use in the MPTCP connection, Node 2 will start sending data to IPA. This means that part of the data of the communication will reach the Attacker but probably not all of it. This already has negative effects, since Node 1 will not receive all the data from Node 2. Moreover, from the application perspective, this would result in DoS attack, since the byte flow will stop waiting for the missing data. However, it is not enough to achieve full hijacking of the connection, since part of data will be still delivered to IP1, so it would reach Node 1 and not the Attacker. In order for the attacker to receive all the data of the MPTCP connection, the Attacker must somehow remove IP1 of the set of available addresses for the connection. In the case of implicit address management, this operation is likely to imply sending a termination packet with IP1 as source address, which may or not be possible for the attacker depending on whether ingress filtering is in place and the location of the attacker. If explicit address management is used, then the attacker will send a remove address control packet containing IP1. Once IP1 is removed, all the data sent by Node 2 will reach the Attacker and the incoming traffic has been hijacked.
- o Segments flowing to the Node 2: As soon as IPA is accepted by Node 2 as part of the address set for the MPTCP connection, the Attacker can send packets using IPA and those packets will be considered by Node 2 as part of MPTCP connection. This means that the attacker will be able to inject data into the MPTCP connection, so from this perspective, the attacker has hijacked part of the outgoing traffic. However, Node 1 would still be able to send traffic that will be received by Node 2 as part of the MPTCP connection. This means that there will be two sources of

data i.e. Node 1 and the attacker, potentially preventing the full hijacking of the outgoing traffic by the attacker. In order to achieve a full hijacking, the attacker would need to remove IP1 from the set of available addresses. This can be done using the same techniques described in the previous paragraph.

A related attack that can be achieved using similar techniques would be a Man-in-the-Middle (MitM) attack. The scenario for the attack is depicted in the figure below.



There is an established connection between Node 1 and Node 2. The Attacker A will use the MPTCP address agility capabilities to place itself as a MitM. In order to do so, it will add IP address IPA as an additional address for the MPTCP connection on both Node 1 and Node 2. This is essentially the same technique described earlier in this section, only that it is used against both nodes involved in the communication. The main difference is that in this case, the attacker can simply sniff the content of the communication that is forwarded through it and in turn forward the data to the peer of the communication. The result is that the attacker can place himself in the middle of the communication and sniff part of the traffic unnoticed. Similar considerations about how the attacker can manage to get to see all the traffic by removing the genuine address of the peer apply.

## 6.2. Time-shifted hijacking attacks

A simple way to prevent off-path attackers to launch hijacking attacks is to provide security of the control messages that add and remove addresses by the usage of a cookie. In this type of approaches, the peers involved in the MPTCP connection agree on a cookie, that is exchanged in plain text during the establishment of

the connection and that needs to be presented in every control packet that adds or removes an address for any of the peers. The result is that the attacker needs to know the cookie in order to launch any of the hijacking attacks described earlier. This implies that off path attackers can no longer perform the hijacking attacks and that only on-path attackers can do so, so one may consider that a cookie based approach to secure MPTCP connection results in similar security than current TCP. While it is close, it is not entirely true.

The main difference between the security of a MPTCP protocol secured through cookies and the current TCP protocol are the time shifted attacks. As we described earlier, a time shifted attack is one where the attacker is along the path during a period of time, and then moves away but the effects of the attack still remains, after the attacker is long gone. In the case of a MPTCP protocol secured through the usage of cookies, the attacker needs to be along the path until the cookie is exchanged. After the attacker has learnt the cookie, it can move away from the path and can still launch the hijacking attacks described in the previous section.

There are several types of approaches that provide some protection against hijacking attacks and that are vulnerable to some forms of time-shifted attacks. We will next present some general taxonomy of solutions and we describe the residual threats:

- o Cookie-based solution: As we described earlier, one possible approach is to use a cookie, that is sent in clear text in every MPTCP control message that adds a new address to the existing connection. The residual threat in this type of solution is that any attacker that can sniff any of these control messages will learn the cookie and will be able to add new addresses at any given point in the lifetime of the connection. Moreover, the endpoints will not detect the attack since the original cookie is being used by the attacker. Summarizing, the vulnerability window of this type of attacks includes all the flow establishment exchanges and it is undetectable by the endpoints.
- o Shared secret exchanged in plain text: An alternative option that is more secure than the cookie based approach is to exchange a key in clear text during the establishment of the first subflow and then validate the following subflows by using a keyed HMAC signature using the shared key. This solution would be vulnerable to attackers sniffing the message exchange for the establishment of the first subflow, but after that, the shared key is not transmitted any more, so the attacker cannot learn it through sniffing any other message. Unfortunately, in order to be compatible with NATs (see analysis below) even though this approach includes a keyed HMAC signature, this signature cannot cover the IP address that is being added. This means that this type of approaches are also vulnerable to integrity attacks of the

exchanged messages. This means that even though the attacker cannot learn the shared key by sniffing the subsequent subflow establishment, the attacker can modify the subflow establishment message and change the address that is being added. So, the vulnerability window for confidentiality to the shared key is limited to the establishment of the first subflow, but the vulnerability window for integrity attacks still includes all the subflow establishment exchanges. These attacks are still undetectable by the endpoints. The SCTP security falls in this category.

- o Strong crypto anchor exchange. Another approach that could be used would be to exchange some strong crypto anchor while the establishment of the first subflow, such as a public key or a hash chain anchor. Subsequent subflows could be protected by using the crypto material associated to that anchor. An attacker in this case would need to change the crypto material exchanged in the connection establishment phase. As a result the vulnerability window for forging the crypto anchor is limited to the initial connection establishment exchange. Similarly to the previous case, due to NAT traversal considerations, the vulnerability window for integrity attacks include all the subflow establishment exchanges. Because the attacker needs to change the crypto anchor, this approach are detectable by the endpoints, if they communicate directly.

### 6.3. NAT considerations

In order to be widely adopted MPTCP must work through NATs. NATs are an interesting device from a security perspective. In terms of MPTCP they essentially behave as a Man-in-the-Middle attacker. MPTCP security goal is to prevent from any attacker to insert their addresses as valid addresses for a given MPTCP connection. But that is exactly what a NAT does, they modify the addresses. So, if MPTCP is to work through NATs, MPTCP must accept address rewritten by NATs as valid addresses for a given session. The most direct corollary is that the MPTCP messages that add addresses in the implicit mode (i.e. the SYN of new subflows) cannot be protected against integrity attacks, since they must allow for NATs to change their addresses. This rules out any solution that would rely on providing integrity protection to prevent an attacker from changing the address used in a subflow establishment exchange. This implies that alternative creative mechanisms are needed to protect from integrity attacks to the MPTCP signaling that adds new addresses to a connection. It is far from obvious how one such creative approach could look like at this point.

In the case of explicit mode, you could protect the address included in the MPTCP option. Now the question is what address to include in

the MPTCP option that conveys address information. If the address included is the address configured in the host interface and that interface is behind a NAT, the address information is useless, as the address is not actually reachable from the other end so there is no point in conveying it and even less in securing it. It would be possible to envision the usage of NAT traversal techniques such as STUN to learn the address and port that the NAT has assigned and convey that information in a secure manner. While this is possible, it relies on using NAT traversal techniques and also tools to convey the address and the port in a secure manner.

## 7. Recommendation

The presented analysis shows that there is a tradeoff between the complexity of the security solution and the residual threats. In order to define a proper security solution, we need to assess the tradeoff and make a recommendation. After evaluating the different aspects in the MPTCP WG, our conclusion are as follows:

MPTCP should implement some form of reachability check using a random nonce (e.g. TCP 3-way handshake) before adding a new address to an ongoing communication in order to prevent flooding attacks.

The default security mechanisms for MPTCP should be to exchange a key in clear text in the establishment of the first subflow and then secure following address additions by using a keyed HMAC using the exchanged key.

MPTCP security mechanism should support using a pre-shared key to be used in the keyed HMAC, providing a higher level of protection than the previous one.

A mechanism to prevent replay attacks using these messages should be provided e.g. a sequence number protected by the HMAC.

The MPTCP protocol should be extensible and it should be able to accommodate multiple security solutions, in order to enable the usage of more secure mechanisms if needed.

## 8. Security Considerations

This note contains a security analysis for MPTCP, so no further security considerations need to be described in this section.

## 9. IANA Considerations

This document does not require any action from IANA.

## 10. Contributors

Alan Ford - Roke Manor Research Ltd.

## 11. Acknowledgments

Rolf Winter, Randall Stewart, Andrew McDonald, Michael Tuexen, Michael Scharf, Tim Shepard, Yoshifumi Nishida, Lars Eggert, Phil Eardley, Jari Arkko, David Harrington, Dan Romascanu, Alexey Melnikov reviewed an earlier version of this document and provided comments to improve it.

Mark Handley pointed out the problem with NATs and integrity protection of MPTCP signaling.

Marcelo Bagnulo is partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program.

## 12. References

### 12.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

### 12.2. Informative References

- [RFC4225] Nikander, P., Arkko, J., Aura, T., Montenegro, G., and E. Nordmark, "Mobile IP Version 6 Route Optimization Security Design Background", RFC 4225, December 2005.
- [RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, October 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC5062] Stewart, R., Tuexen, M., and G. Camarillo, "Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures", RFC 5062,

September 2007.

- [RFC5535] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, June 2009.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [I-D.ietf-mptcp-multiaddressed]  
Ford, A., Raiciu, C., and M. Handley, "TCP Extensions for Multipath Operation with Multiple Addresses",  
draft-ietf-mptcp-multiaddressed-02 (work in progress),  
October 2010.

#### Author's Address

Marcelo Bagnulo  
Universidad Carlos III de Madrid  
Av. Universidad 30  
Leganes, Madrid 28911  
SPAIN

Phone: 34 91 6248814  
Email: marcelo@it.uc3m.es  
URI: <http://www.it.uc3m.es>





This Internet-Draft, draft-raiciu-mptcp-congestion-01.txt, has been replaced by another document, draft-ietf-mptcp-congestion-00.txt, and has been deleted from the Internet-Drafts directory.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the author(s) or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the author(s) directly.

Draft Author(s):

M. Handley <m.handley@cs.ucl.ac.uk>  
C. Raiciu <c.raiciu@cs.ucl.ac.uk>  
D. Wischik <d.wischik@cs.ucl.ac.uk>