Multipath TCP
Internet-Draft
Intended status: Standards Track
Expires: August 11, 2012

G. Hampel
T. Klein
Alcatel-Lucent

February 8, 2012

MPTCP Proxies and Anchors
draft-hampel-mptcp-proxies-anchors-00

Abstract

MPTCP proxies and anchors are network-based functions, which support MPTCP connections. The MPTCP proxy provides multipath support for MPTCP-capable hosts on behalf of their MPTCP-unaware peers. This facilitates incremental deployment of MPTCP. The MPTCP anchor permits subflow establishment for MPTCP connections when direct interaction between end hosts fails. This permits tolerance to local IP protocol restrictions and it provides robustness in case of break-before-make mobility events. MPTCP proxies and anchors are especially suited for wireless access environments.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). 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 August 11, 2012.

Table of Contents

1. Introduction

Currently, a host can enjoy the advantages of MPTCP only if its peer supports MPTCP as well [1]. This requirement creates an impediment to incremental deployment since the incentive for a host to upgrade to MPTCP is small as long as its potential peers have not upgraded too.

The incremental deployment problem especially applies to wireless environments, where traffic is dominated by interactions between mobile clients and network-side servers. While MPTCP can be rolled out rather quickly on mobile devices due to their short life cycle and frequent kernel upgrades, changes on application servers are usually harder to conduct. Further, the benefit of MPTCP may be more obvious to mobile users than to application service providers.

The incremental deployment problem can be overcome through the introduction of the MPTCP proxy, which resides in the network and provides MPTCP support for MPTCP-capable hosts (e.g. mobile devices) on behalf of their MPTCP-unaware peers (e.g. application services).

Since MPTCP proxies will most likely be run by network operators rather than application service providers they can support a multitude of application services, which makes incremental deployment of MPTCP rather efficient. Further, network operators may see a benefit in MPTCP deployment since it adds value to the network services they provide and since they mostly support a billing mechanism to reimburse themselves from MPTCP operation.

The MPTCP anchor is another MPTCP network function whose main purpose is to support end-to-end multipath connections. It operates as a subflow relay to facilitate subflow establishment between end points that do not enjoy direct reachability. This may happen, for instance, if the end points pertain to different IP protocols or if the hosts have lost end-to-end connectivity after a break-before-make mobility event.

The anchor function is most beneficial for peer-to-peer applications such as voice/video communciations, which are run on MPTCP-enabled mobile or multi-homed devices. Flexibility in IP protocol support is important for this use case during the rollout of IPv6. The anchor function further allows the network operator to provide differentiated services for over-the-top applications.

This document discusses relevant features and signaling enhancements needed for the support of MPTCP proxies and MPTCP anchors.

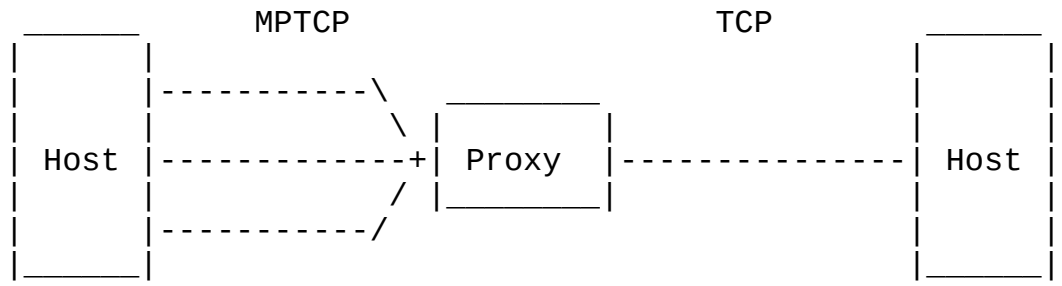
2. MPTCP Network Functions

All network-based functions that interact with MPTCP connections through MPTCP signaling are referred to as "MPTCP network functions". MPTCP network functions are assumed to reside on "MPTCP network nodes". We consider two types of MPTCP network functions namely the MPTCP proxy and the MPTCP anchor. Anchor- and proxy functions can be collocated on one MPTCP network node.

2.1. MPTCP Proxy

The MPTCP proxy supports MPTCP on behalf of an MPTCP-unaware host. It splits the connection between multipath-capable and multipath-unaware host into a MPTCP section and a TCP section, respectively (Figure 1). All subflows established by the multipath-capable host terminate at the proxy.

Proxy operation is discussed in Section 4.



Split connection with MPTCP Proxy

Figure 1

2.2. MPTCP Anchor

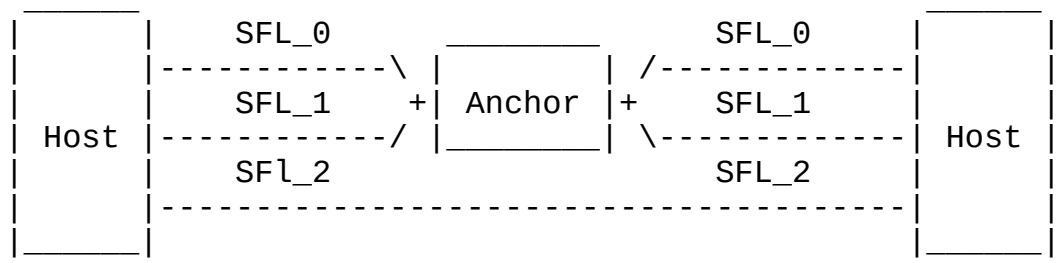
The MPTCP anchor provides a network-based access point (i.e. IP address), which a MPTCP host can use to create additional subflows to the peer. The anchor relays all packets arriving from this host to the peer and vice versa. This creates a split subflow consistent of one section between host and anchor and the other between anchor and peer (Figure 2). The anchor’s operation involves address- and eventually also port translation. Anchors can also insert or modify MPTCP options of passing or relayed packets.

Anchor addresses can be introduced during connection establishment or at any later point in time. Anchor functions can be invoked or released during the entire lifetime of the connection.

An anchor function can interconnect end points using different IP

protocol versions with a subflow. In this case the anchor operates as an IP protocol translator (Section 5.5). The anchor also serves as a "meeting point" for the establishment of a new subflows when all other subflows have failed and direct end-to-end subflow establishment is not possible. This applies to scenarios where both end hosts have simultaneously moved or when one host moves while the other resides behind a firewall (Section 5.6).

Anchor operation is discussed in Section 5.



MPTCP connection with Anchor

Figure 2

2.3. Implicit vs. Explicit Proxies

An implicit proxy resides on the direct routing path between two hosts engaging into a connection. This allows the hosts to establish the connection directly with each other, while the proxy can derive all information via packet inspection, insert and modify packets as necessary and thereby create the MPTCP-TCP split connection. This proxy is referred to as "implicit" since not explicit signaling is necessary.

When the proxy does *not* reside on the direct routing path between both hosts, explicit signaling is needed to introduce the proxy to the connection. The same applies to a proxy that does not reside on the path used for connection initiation. Such a proxy is referred to as "explicit" proxy.

An implicit proxy typically resides on a central router in the access network used by one of the hosts during connection establishment. An explicit proxy can reside in any network.

2.4. Implicit vs. Explicit Anchors

The terms "implicit" and "explicit" can also be defined for anchors.

An implicit anchor resides on the routing path used by a subflow of a

MPTCP connection. This allows the anchor to derive all necessary connection-related information via packet inspection during the establishment of this subflow. Then, it can insert and modify packets as necessary and thereby offer anchor services to the end hosts.

When an implicit anchor resides on the initial subflow, it can offer services to *both* end hosts. Otherwise, it can offer services only to the subflow-initiating end host (see Section 5).

When the anchor does not reside on a direct routing path between both connection end points, explicit signaling is needed to introduce the anchor to the connection. Such an anchor is referred to as "explicit" anchor.

Anchors can support connections between two hosts as well as between a host and a MPTCP proxy. Usually, anchors are more beneficial in the former of the two scenarios.

2.5. End-Host Authentication

MPTCP proxies and anchors should support an explicit or implicit mechanism to authenticate one of the connection’s end hosts. This allows the proxy- or anchor operator to charge for operation of the respective MPTCP network function. There are also security reasons that require end-host authentication as outlined in Section 8.

3. Deployment Scenarios

The predominant use case for MPTCP proxies and MPTCP anchors is seen in wireless access networks. This is motivated by the increasing number of wireless devices that support multiple access technologies as well as multi-homing.

In one deployment scenario, the MPTCP network function resides on a central router of a wireless access network, e.g. a 3G/4G mobile network. Especially 3G and 4G mobile network operators may see an incentive for MPTCP proxy support since it allows them to dynamically offload traffic from licensed to unlicensed spectrum. Further, 3G- and 4G mobile networks already provide a centralized architecture, security support and charging functions, which can be used for MPTCP proxy or anchor operation.

There are also technical reasons to place MPTCP proxies inside cellular networks which are related to the wide-area coverage these networks typically provide. Therefore, the connection can be established via the cellular interface and subsequently migrated to other paths and networks. This substantially simplifies signaling since an implicit proxy/anchor can be used. Further, the cellular network can be used for reachability.

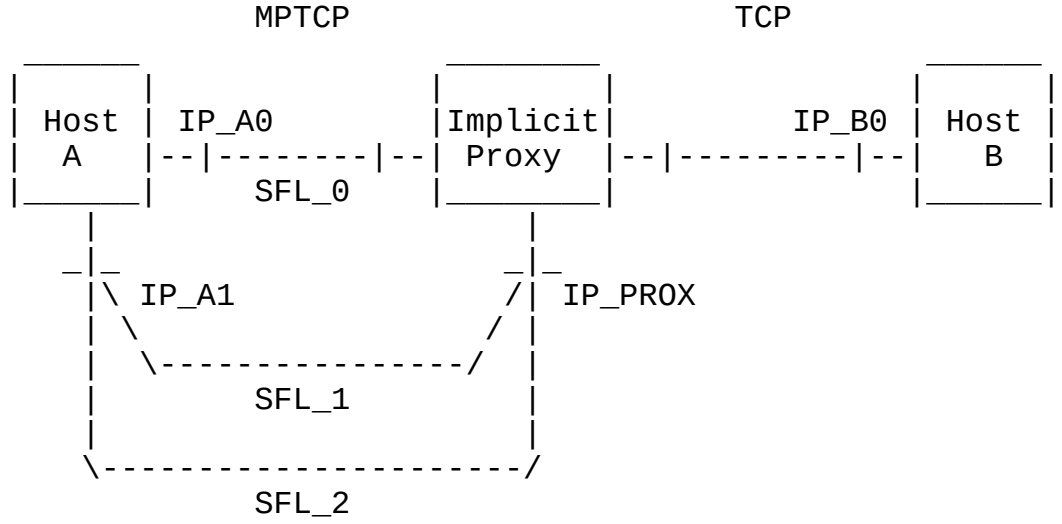
It is expected that anchor- and proxy functions are collocated.

For any deployment scenario, MPTCP-capable hosts need to be configured appropriately so that they can take advantage of implicit and explicit MPTCP network functions. Some aspects of host configuration are discussed in Section 6.

4. Operation with MPTCP Proxies

Proxies must be introduced to the connection during connection establishment and stay engaged during the entire lifetime of the connection.

4.1. Introduction of Implicit Proxy



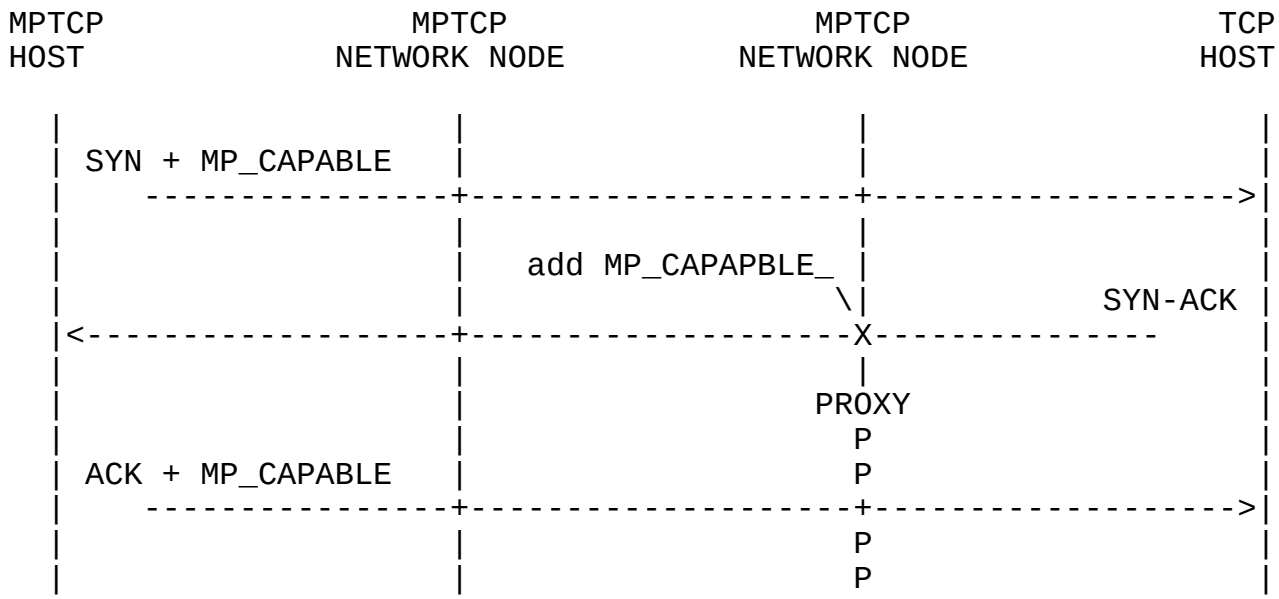
MPTCP-TCP split connection with implicit MPTCP proxy

Figure 3

The MPTCP-capable host starts a MPTCP connection by sending a TCP SYN packet with MP_CAPABLE option to its peer. The proxy inspects the packet and caches the end point locators consistent of IP addresses and port numbers as well as the key enclosed in the MP_CAPABLE option. Based on these locators, the proxy identifies and intercepts the peer's SYN-ACK response packet. The implicit proxy does not change the locators contained on the packet.

In case the SYN-ACK response does not hold the MP_CAPABLE option, the proxy initiates multipath support. It creates a key on behalf of the peer, inserts a MP_CAPABLE option with this key into the SYN-ACK packet, and then forwards the packet to the connection-initiating host.

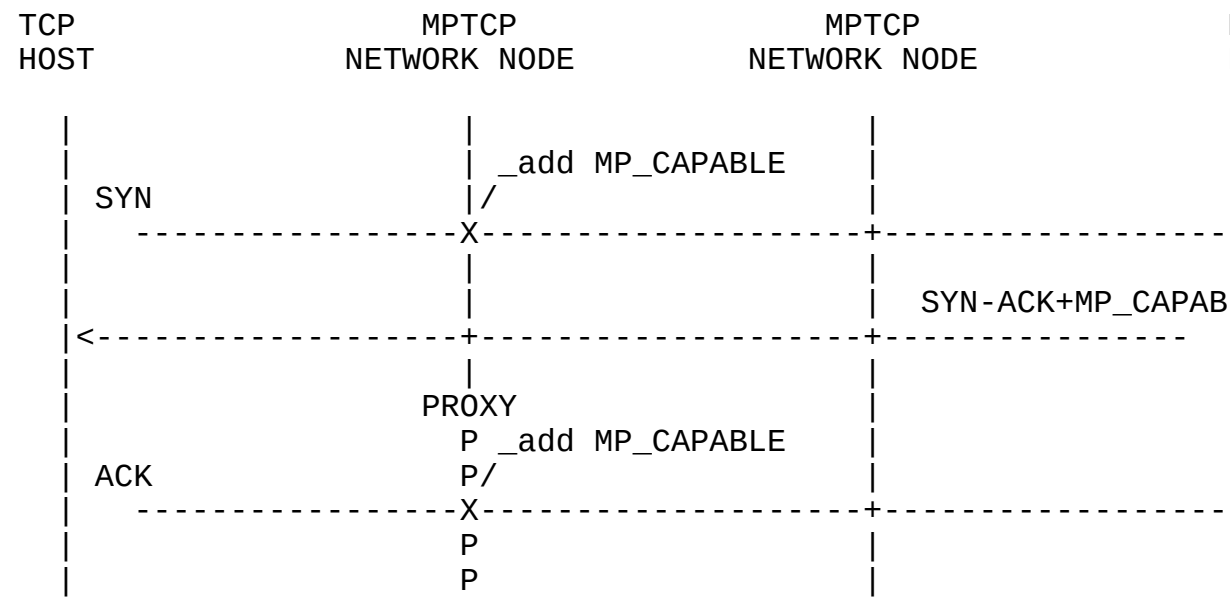
If the SYN-ACK response *does* contain an MP_CAPABLE option, the proxy is not needed. In this case, the network node can provide anchor functionality (see Section 5).



Connection initiation by MPTCP-capable host with implicit proxies on initial path

Figure 4

If multiple implicit proxies reside on the initial path, the proxy closest to the peer should become the MPTCP end point. Since this proxy is the first to receive the peer's SYN-ACK packet, it automatically assumes multipath support by inserting the MP_CAPABLE option.



Connection initiation by MPTCP-unaware host with implicit proxies on initial path

Figure 5

The implicit proxy can also support scenarios, where the peer rather than the connection-initiating host is MPTCP-capable. In this case, the MPTCP proxy adds the MP_CAPABLE option with its own key to the initial SYN packet. If the SYN-ACK response by the peer carries the MP_CAPABLE header, the proxy assumes multipath support.

If multiple proxies reside on the initial path in this latter case, the proxy closest to the session-initiating host should become the MPTCP end point. Since this proxy is the first to receive the peer's SYN packet, it automatically assumes multipath support by inserting the MP_CAPABLE option into this SYN packet.

These signaling procedures work fine as long as at least one of the end hosts supports MPTCP. A problem occurs, when multiple proxies reside on the initial path but *neither* of the end hosts supports MPTCP. In this case, one proxy may add MP_CAPABLE to the SYN packet and the other to the SYN-ACK response packet. In this manner, both proxies end up creating a TCP-MPTCP-TCP split connection with multipath support between each other. Such a situation is likely to occur when each of the hosts' access networks supports a proxy.

To avoid such a situation, the proxy inserting the MP_CAPABLE option into the SYN packet has to reveal its true nature by adding a PROXY flag to this option. When another proxy inspects the SYN packet and finds the MP_CAPABLE option with PROXY flag set, it should not insert

MP_CAPABLE to the SYN-ACK response.

For implicit proxies, end-host authentication is implicitly provided by the host's access authentication as long as the proxy resides in the access network of one of the end hosts. This makes additional signaling for end-host authentication unnecessary.

While this solution restricts operation of implicit proxies to access providers and their affiliates (e.g. roaming partners), it covers the most relevant deployment scenarios.

4.2. Subflow Management with Implicit Proxy

Since the proxy splits the connection into a MPTCP section and a TCP section, it becomes the end point for all further subflows. These subflows may be initiated by the MPTCP-capable host or by the proxy itself.

When the proxy is implicit, it must inform the multipath-capable host about its existence as well as its IP address. Otherwise, the multipath-capable host may try to establish subflows with the multipath-unaware peer. For this purpose, implicit proxies should set the PROXY flag on those MP_CAPABLE options they insert into SYN or SYN-ACK packets. This flag informs the multipath-capable host that the remote end point is represented by a proxy.

After connection establishment, the proxy should advertise its address via ADD_ADDR to the multipath-capable host. This step is necessary since the host does not know the proxy's address.

Currently, the ADD_ADDR option also conveys the request for immediate subflow establishment to the enclosed address. This request has the purpose to enable subflow creation in reverse direction, i.e. when the peer resides behind a firewall.

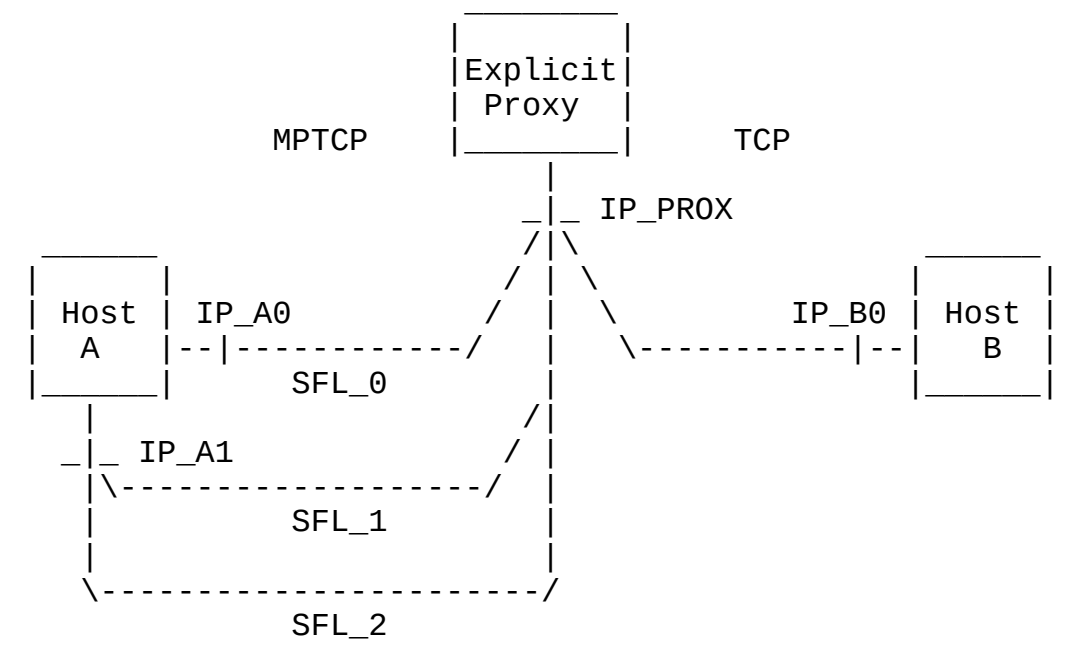
Obviously, immediate subflow creation is not desirable when a proxy announces its IP address as an alterative end point. Therefore, the ADD_ADDR option should be furnished with a JOIN flag, which allows differentiating between the two purposes of ADD_ADDR. Hence subflow creation is only requested when the JOIN flag is set.

Since MPTCP options are not delivered reliably, the ADD_ADDR option may get lost. In this case, the host has no means to find out about the proxy's IP address. For that reason, an additional SEEK_ADDR option should be supported which allows the host to solicit address advertisements by MPTCP network nodes and the peer.

SEEK ADDR should hold a field for the IP version requested. If this

field is set to zero, addresses pertaining to any IP version can be advertised.

4.3. Introduction of Explicit Proxy



MPTCP-TCP split connection with explicit MPTCP proxy

Figure 6

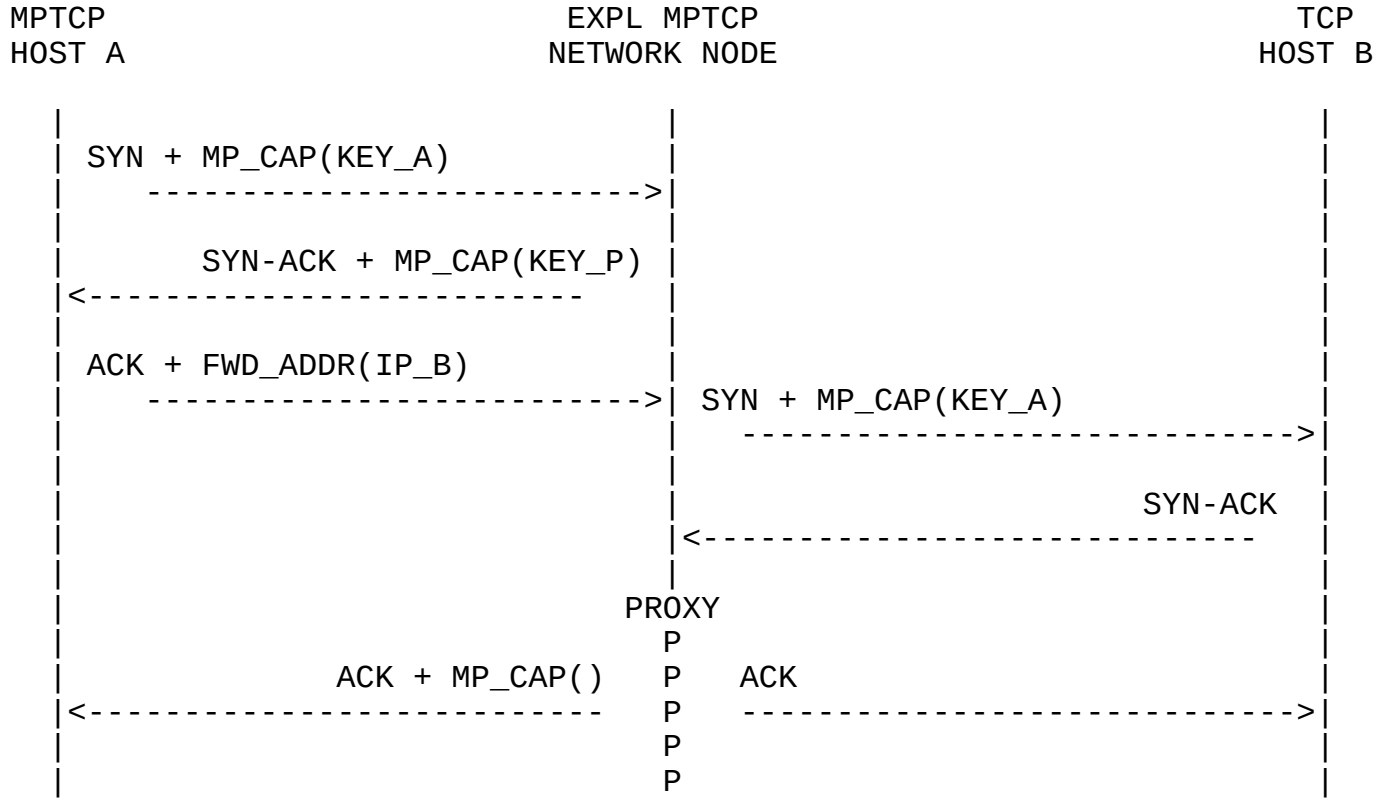
If the proxy does not reside on the direct routing path of the intended connection the connection initiator must provide the proxy with explicit information on the peer's network locator, i.e. IP address and port number. Since the explicit proxy may reside in a different network, additional signaling for host authentication has to be supported as well.

In case connection establishment reveals that both end hosts support MPTCP (or if the peer is supported by an implicit proxy), the explicit proxy function is not needed. In this case, the MPTCP network node automatically assumes explicit anchor function since it splits the initial subflow.

For connection establishment, the following signaling approaches are considered:

- In-band MPTCP signaling: The peer's network locator (i.e. IP address and port number) and the host's authentication information are sent in-band on MPTCP options. Since the amount of

information is too large to fit into the TCP header of the initial SYN packet additional packets need to be exchanged for signaling purposes. A simple handhake can be realized where the MPTCP keys are used as authenticators (Figure 7):



Connection establishment with explicit proxy and in-band MPTCP signaling

Figure 7

* The connection-initiating host (host A) sends the SYN packet with MP_CAPABLE containing key_A as authenticator to the explicit MPTCP network node which caches key_A and host A's locator.

* The MPTCP network node answers with SYN_ACK enclosing MP_CAPABLE with key_P as its own authenticator. It should *not* set the PROXY flag, since it doesn't know at this point if proxy function is required.

* Host A sends an ACK enclosing FWD_ADDR, which holds the peer's (i.e. host B's) IP address. FWD_ADDR may also hold a port number if it is different from the port number used to address

the MPTCP network node.

* The MPTCP network node sends SYN with MP_CAPABLE holding key A to host B using its own IPaddress. It also sets the ANCHOR flag in MP_CAPABLE as discussed in Section 5.

* If host B is not MPTCP-capable, it responds with a simple SYN-ACK packet. Otherwise, it inserts MP_CAPABLE with key B into the SYN-ACK packet. If MP_CAPABLE is absent, the MPTCP network node assumes proxy function. Otherwise, it assumes anchor function.

* The proxy function sends an ACK to host A and encloses the MP_CAPABLE header with the PROXY flag set. This informs host A that host B does not support MPTCP and that the MPTCP network node has assumed proxy function. The MP_CAPABLE option does not have to hold any key at this point since all keying information has already been exchanged.

* The proxy function also sends a simple ACK to host B.

o Out-of-band MPTCP signaling: MPTCP introduces a separate signaling connection to exchange the necessary signaling information prior to establishment of the traffic connection. Since such an out-of-band solution substantially extends the present scope of MPTCP it is not further considered.

o Independent signaling: The host and the explict MPTCP network node use an independent signaling protocol, in which the host authenticates itself and provides the peer’s locator. This protocol can be supported on session or application layer such as SIP [2], for instance. In this protocol, host and MPTCP network node establish the 64-bit key, which is cached by the proxy together with the peer’s locator and inserted by the host into MP_CAPABLE when initiating the MPTCP connection. This allows the network node to find the peer’s locator and to forward the SYN packet to the peer using its own IP address. The network node should set the ANCHOR flag when relaying the MP_CAPABLE packet to the peer. In case the SYN-ACK return packet arriving from the peer does *not* contain an MP_CAPABLE option, the network node assume proxy function. In this case, the proxy inserts MP_CAPABLE into the SYN-ACK packet, sets the PROXY flag and sends the packet to the connection-initiating host using its own IP address and port number as the packet’s source. The host responds with an ACK holding its own key as well as the key contained in the SYN-ACK packet.

Security issues related to such explicit proxy solutions are

discussed in Section 8.

It makes little sense to consider explicit-proxy scenarios where the connection-initiating host is not MPTCP-capable.

4.4. Subflow Management with Explicit Proxy

Subflow establishment with an explicit proxy follows along the same lines as for an implicit proxy. The explicit proxy, however, does not have to send an ADD_ADDR option since the host already knows the proxy's address.

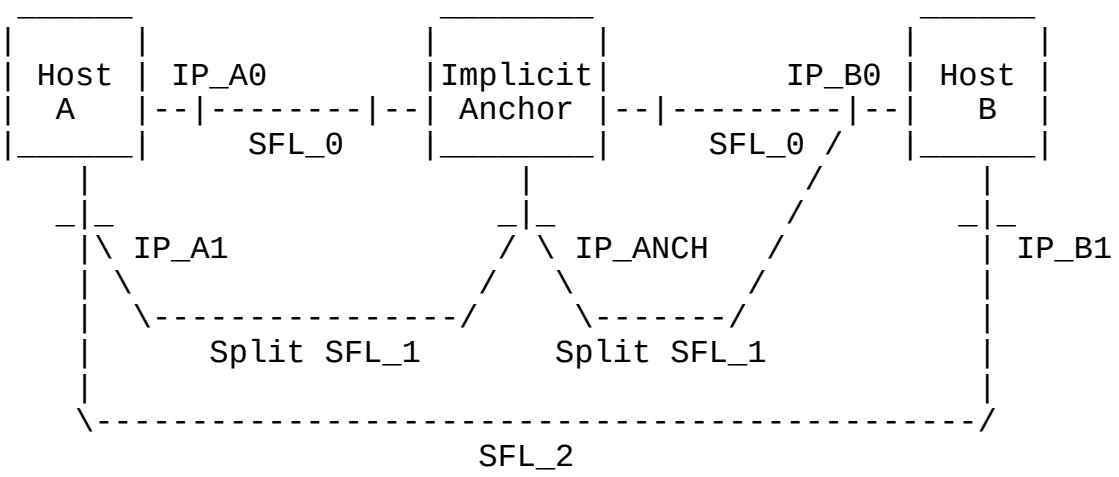
5. Operation with MPTCP Anchors

The anchor function splits subflows into two subflow sections, where each section interconnects an end host with one of the anchor's IP addresses (Figure 8). The anchor relays all packets arriving on one subflow section to the other by rewriting the IP addresses of the packet headers. The anchor may also translate port numbers. Anchors can also insert or modify MPTCP options of passing packets.

To keep end-to-end semantics in tact, the end nodes must have full awareness of the anchor's presence and its operation, i.e. if subflows are split and if an IP address belongs to an anchor or to the peer. Further, each host must know about the address-id its peer uses on the remote section of a split subflow. This ensures proper subflow tear-down in case the peer announces address removal via REMOVE_ADDR option.

Anchors can be introduced during connection establishment or at any later point in time. Anchor services can be invoked or released during the entire lifetime of the connection.

5.1. Introduction of Implicit Anchor



MPTCP connection with implicit MPTCP anchor

Figure 8

When an implicit anchor resides on the initial path, it caches the locators (i.e. IP addresses and port numbers) of the initial subflow as well as the keys exchanged during connection establishment. This allows the anchor to derive the corresponding tokens and cache them together with the end hosts' locators of this subflow.

Then, the anchor advertises its IP address to the end hosts by sending an ADD_ADDR option to one or to both end hosts. The ADD_ADDR option can be inserted into a packet that is passing on the initial subflow. The anchor may also insert a port number into the ADD_ADDR option.

The anchor has to mark the ADD_ADDR option in a manner that allows the host receiving the option to distinguish it from an ADD_ADDR option sent by the peer. For this purpose, the anchor should set the address-id in the ADD_ADDR option to an anchor-reserved value (e.g. 255). This does not lead to any conflict in case multiple anchors advertise their addresses with the same address-id value, since anchor addresses are considered invariants that need not be removed. Obviously, neither end hosts nor proxies should use this anchor-reserved address-id value.

When an implicit anchor resides on the path used by a later subflow, it caches the subflows locators as well as the token used during subflow establishment. Obviously, anchor support can only be provided for the host that initiated this subflow (host A) but not for its peer (host B) since the anchor only knows host B’s token. Therefore, the anchor advertises its IP address (and port number) only to host A.

The host receiving an ADD_ADDR options from an anchor caches the anchor’s address and port number contained in this option. When the ADD_ADDR option does not carry a port number, the remote port number of the subflow, where the option arrived, is cached instead.

Since the delivery of ADD_ADDR is not reliable, an end host may proactively seek anchor addresses via the SEEK_ADDR option introduced above. Both anchor and peer should respond with an ADD_ADDR option. The host can differentiate the originators of these replies by the enclosed address-id value.

5.2. Subflow Management with Implicit Anchor

When a host wishes to establish a subflow via anchor, it initiates a subflow to the address and port number cached for the anchor. Based on the destination port number of the SYN packet and the token contained in MP_JOIN, the anchor identifies the peer’s locator and forwards the packet to the peer using one of its own addresses and port numbers as the packet’s source. The peer’s SYN-ACK return packet and all following packets are relayed by the anchor in the same manner. Since the anchor does not change the address-ids contained in the MP_JOIN options of the initial handshake, each host learns the peer’s address-id used for this split-subflow.

While the host initiating the subflow (host A) is aware of the anchor's presence, its peer (host B) may not know that this subflow is split because the anchor has not introduced itself to the peer or because the corresponding ADD_ADDR option got lost. In such a case, host B may falsely assume that the anchor's IP address belongs to host A and map it to the address-id contained in MP_JOIN. This may lead to a conflict, in case host A has announced (or will announce) this address-id for another address. Further, host B may be tempted to use the anchor's IP address for further subflows without knowing that this may invoke triangular routing.

To avoid such misunderstanding, the MP_JOIN option on the SYN packet has to be marked with an ANCHOR flag. This flag tells host B, that the source address on the packet header belongs to an anchor and that it is not associated with the address-id carried in the MP_JOIN option. The ANCHOR flag should be set by the anchor when relaying the SYN packet.

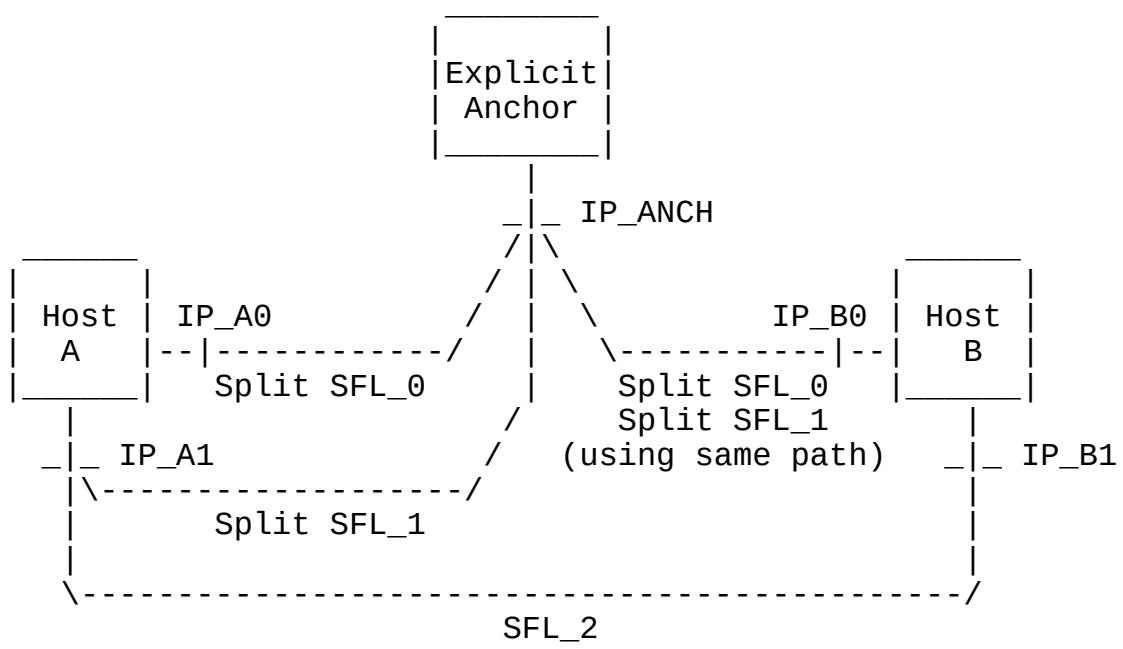
While host B may implicitly learn the anchor's IP address in this manner, it is not advised to use this anchor for new subflows unless the anchor has explicitly advertised its IP address. Host B can solicit such IP address advertisement via SEEK_ADDR sent on the split subflow.

Each host should cache the peer's address-id together with the state information it holds for the corresponding split subflow. In case the host receives an REMOVE_ADDR option, it can identify and tear down all split-subflows pertaining to the address-id held in this option.

The establishment of split subflows via anchor may introduce address-ids without the corresponding IP addresses. This is a similar situation as when direct end-to-end subflows pass network address translators, and it does not pose any principle problem.

The anchor caches the host's locators and address-ids of the split subflow together with all information it holds for this connection. The anchor further keeps subflow-related state information for a short time frame after the subflow has been closed. The tokens and address ids are held for a short time after the last subflow known by this anchor has been closed. The tear-down delay permits the anchor to support break-before-make mobility scenarios discussed below.

5.3. Introduction of Explicit Anchor

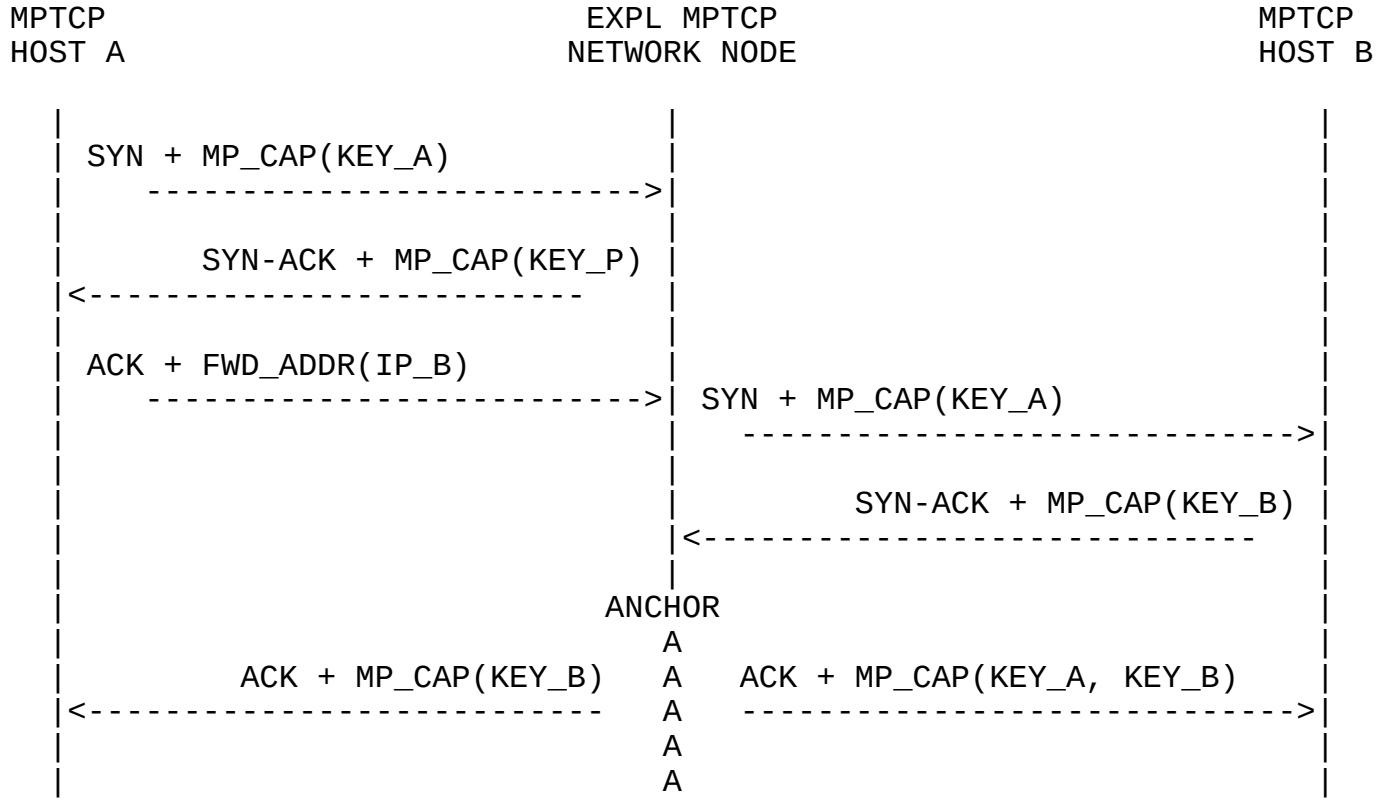


MPTCP-TCP split connection with explicit MPTCP anchor

Figure 9

If the anchor does not reside on a direct routing path it has to be introduced via explicit signaling by one of the hosts. The signaling has to include authentication information and the peer's locator. Since these are the same conditions as for explicit proxies the same solution scenarios can be applied as discussed in Section 4.3. For the reasons mentioned above, only scenarios with in-band MPTCP signaling and independent signaling are considered.

- In-band MPTCP signaling: The first four steps of the connection establishment are identical to those discussed for the explicit proxy (see Figure 7 and Figure 10):



Connection establishment with explicit anchor and in-band MPTCP signaling

Figure 10

* Steps 1-4 of connection establishment with explicit proxy.

* In case host B is MPTCP-capable, it inserts MP_CAPABLE with key B into the SYN-ACK response packet. Upon reception of this packet, the MPTCP network node assumes anchor function instead of proxy function.

* The anchor function sends an ACK to host A and encloses the MP_CAPABLE header with key_B and it sets the ANCHOR flag. This informs host A that host B does support MPTCP and that the MPTCP network node has assumed anchor function. At this point, host A overwrites key_P with key_B.

* The anchor function also sends an ACK to host B, where it inserts MP_CAPABLE with key_A and key_B and sets the ANCHOR flag. This tells host B that an anchor resides on the initial path.

- Independent signaling: The explicit MPTCP network node relays the host's SYN packet holding the MP_CAPABLE option to the peer. If the SYN-ACK return packet holds the MP_CAPABLE option, the MPTCP network node assumes anchor function and the initial subflow becomes a split subflow. When relaying the SYN-ACK packet to the connection-initiating host, the anchor should set the ANCHOR flag. The host responds with an ACK holding MP_CAPABLE with both keys.

In case host B is not multipath-aware it may be supported by an implicit proxy residing on the path between host B and the explicit anchor. This proxy may reside in host B's access network for instance. The implicit proxy sets the PROXY flag in the MP_CAPABLE option of the SYN-ACK return packet as described in section 4.1. Since the explicit anchor sets the ANCHOR flag at the same time, host A can infer that the PROXY flag was set by an implicit proxy.

A host can also introduce an explicit anchor after connection establishment. This has only limited benefit since the peer won't be able to proactively use this anchor. Further, it is rather complicated to embed such an anchor introduction into the MP_JOIN handshake. For that reason, only methods involving independent signaling protocols are considered here. Such a protocol has to provide authentication information, the remote end point locator and the remote tokens used on this connection.

5.4. Subflow Management with Explicit Anchor

After introduction of the explicit anchor, establishment of further split subflows follows the same procedure as discussed for implicit anchors in Section 5.2.

5.5. Protocol Translation with Anchor

The anchor can be used for IP protocol translation on a split subflow in case host A wishes to support IPv6 on a new interface while host B only supports IPv4. Protocol translation further becomes necessary when one host moves from an IPv4 network to an IPv6 network while the peer's network only supports IPv4 (and vice versa).

In such scenarios, host A sends SEEK_ADDR on all subflows with the IPVer field set to IPv6. In response, anchors will send their respective IPv6 addresses. Then, host A initiates a new subflow to one anchors' IPv6 address. Since the anchor has cached at least one of host B's IPv4 addresses, it can create an IPv6/IPv4 split-subflow using an IPv6 and an IPv4 address.

5.6. Connection Robustness with Anchor

The anchor can provide enhanced connection robustness in scenarios where the only remaining subflow breaks and direct end-to-end subflow establishment is not possible. This may happen, for instance, when both hosts simultaneously move to a new address. Direct subflow establishment is not possible in this case since neither host knows the peer's new IP address.

In another scenario, a host moves to a new IP address while the peer resides behind a firewall. The host cannot reach the peer since the firewall blocks packets arriving from a new address. The peer cannot reach the host either since it does not know the host's new IP address.

In these scenarios, each host will try to establish a direct subflow first. If this fails each host tries subflow establishment via an anchor. Since the anchor recognizes the connection based on token and port number contained in each host's SYN-packet, it can cache the host's new address contained on the packet and use it as the destination for SYN-packets sent by the peer. In this manner, a new subflow can be established via the anchor.

For this purpose, the anchor should keep connection-related state information for some time after the subflow it is residing on has been torn down.

The procedure further requires that the anchor holds both end hosts' tokens. This applies to anchors that reside on the initial path during connection establishment.

6. Host Configuration

MPTCP-capable hosts should be appropriately configured to take advantage of MPTCP network functions. In a deployment scenario, where proxies and anchors are integrated with a central router of a 3G/4G cellular network, the host should initiate connections that deserve MPTCP support via the cellular interface if possible. After connection establishment, additional paths can be established and utilized for traffic exchange.

In case explicit MPTCP network functions are provided, the host must be configured to support the proprietary protocol that introduces these nodes to the MPTCP connection. It must further be configured with the IP addresses for explicit proxies.

The details on host configuration and the criteria on path selection are beyond the scope of this document.

7. New Signaling

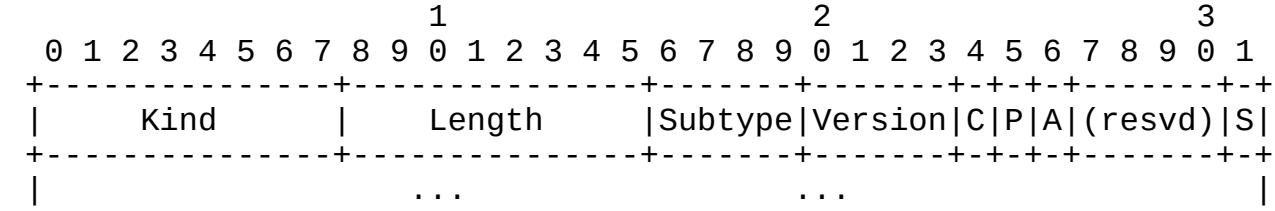
The following subsections discuss signaling changes necessary to support MPTCP network functions.

7.1. PROXY Flag

The PROXY flag needs to be added to the MP_CAPABLE option. The PROXY flag is set by MPTCP network nodes to announce that they assume proxy function.

The PROXY flag serves two purposes. It avoids that implicit proxies residing on the initial path between MPTCP-unaware hosts sustain a MPTCP connection with each other. It also informs a MPTCP-capable host that a proxy provides MPTCP on behalf of an MPTCP-unaware peer. This avoids unnecessary attempts by this host to establish subflows directly with the MPTCP-unaware peer.

The PROXY flag can be added into the header of the MP_CAPAPBLE option (shown as "P" in Figure 11).



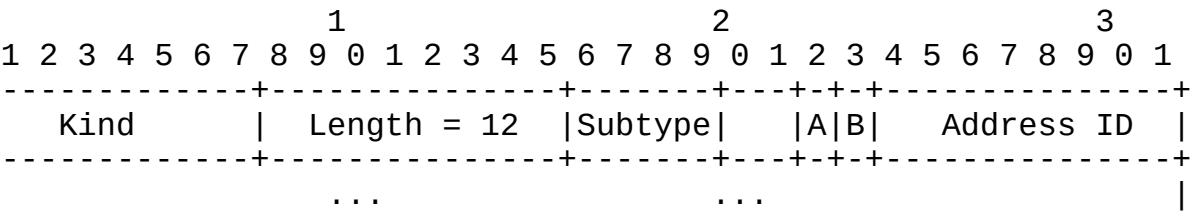
MP_CAPABLE header with PROXY (P) and ANCHOR (A) flags

Figure 11

7.2. ANCHOR Flag

The ANCHOR flag needs to be added to the MP_CAPABLE option and to the MP_JOIN option. The flag informs the receiving host (or proxy) that an anchor has relayed this packet. This avoids misunderstandings about the source IP address of the packet and the address-id it carries.

The ANCHOR flag can be added to the headers of MP_CAPAPBLE and MP_JOIN (shown as "A" in Figure 11 and Figure 12).



MP_JOIN header for SYN and SYN-ACK with ANCHOR flag

Figure 12

7.3. JOIN Flag

The ADD_ADDR option is currently overloaded with two requests: 1) Cache this address and 2) initiate a subflow to this address right away. While this bundling of requests makes sense for end-to-end interactions, it becomes problematic for proxies and anchors, which only want to inform the peers about their respective addresses.

The issue can be resolved by adding a JOIN flag to the ADD_ADDR option. This, however, creates some issues since the option has no room left for additional information. The option is further rather long, especially if IPv6 addresses and port numbers have to be carried.

The following approaches can be considered:

- The IPVer field is reduced from 4 to 3 bits as proposed by Olivier Bonaventure. This still leaves room for 5 future IP versions apart from IPv4 and IPv6. (Note that IP version = 0 is used by SEEK_ADDR to refer to "all IP versions"). The released bit is available for the JOIN flag.

- The ADDRESS ID field is reduced by 1 bit to allocate room for JOIN as proposed by Costin Raiciu. This reduces the number of simultaneously supported addresses from 256 to 128 (or 255 and 127 if the anchor-reserved address-id is included as well).

- The ADD_ADDR option only provides addresses and address-ids while a new option conveys the request to create a subflow with respect to a specific address id. A similar proposal was also made by Yoshifumi Nishida.

- An octet is added to the ADD_ADDR option.

7.4. Anchor-Reserved Address-Id Value

The anchor-reserved address-id value is used when anchors advertise their IP address via ADD_ADDR. It informs the receiving host that the address belongs to an anchor and not to the peer.

The anchor reserved address-id value could be set for 255, for instance.

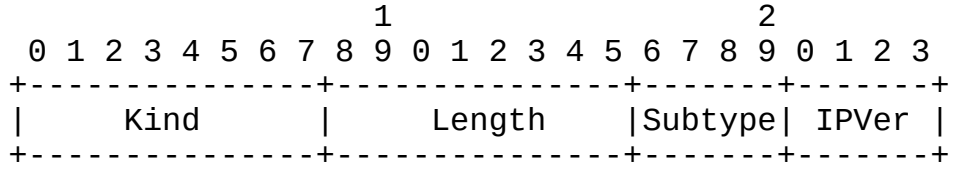
7.5. SEEK_ADDR Option

The SEEK_ADDR option is sent by a host to solicit its peer as well as proxies and anchors to advertise their addresses. This option is necessary for operation with proxies and anchors, which rely on reliable address advertising.

The SEEK_ADDR option holds the IP version field. If the value of this field is set to zero, addresses to all IP versions are sought.

SEEK_ADDR also permits peers and MPTCP network nodes to reduce address advertising. It is not necessary, for instance, to preemptively advertise IPv6 addresses on connections that only use IPv4 and vice versa.

The SEEK_ADDR option only holds the IP version field which leads to length of 3 octets (Figure 13).



SEEK_ADDR option

Figure 13

7.6. FWD_ADDR Option

The FWD_ADDR option is used by a host to forward its peer’s IP address and port number to an explicit MPTCP network node.

The fields of the FWD_ADDR are identical to that of the ADD_ADDR option. Since both options have different semantic meanings they should also carry different subtypes.

8. Security

Mobility and multi-homing protocols are vulnerable to session redirection attacks such as session hijacking and distributed DoS (DDoS)[3]. For MPTCP, these matters have been discussed in [4]. The introduction of implicit proxies and anchors does not add new principal vulnerabilities.

One potential weakness is seen in connections via explicit proxy (or anchor), since the proxy can be used by the adversary to disguise its true location. In a DDoS attack, the adversary establishes multiple connections with the victim host and then floods the victim with a high volume of traffic on each connection. The severity of such an attack does not change when these connections are conducted via explicit proxy. Since the proxy uses its own IP address to forward the attacker's packets to the victim, the attacker's IP address remains hidden to the victim. This makes it impossible for the victim to identify an adversary prior to accepting a connection and to trace back the traffic flood to the attacker's location.

One could argue that this situation could be improved by specifying a strong authentication method to be exercised between host and proxy. This, however, is not necessarily the case since a strong authentication protocol by itself does not enforce the use of strong authenticators.

Note that this situation is different for mobility protocols like Mobile IPv6. In Mobile IPv6, the home agent uses the mobile host's unique home address as the source for traffic originated by the mobile host. The home address is therefore an authenticator of the traffic originator.

To support the same level of security, the explicit proxy could use a unique IP address for each host. While such an approach is feasible in IPv6 it may have limited applicability in IPv4 due to IP address exhaustion.

9. Acknowledgements

The authors wish to acknowledge suggestions and contributions on proxies and anchors by Olivier Bonaventure, Philip Eardley, Alan Ford, Mark Handley, Yoshifumi Nishida and Costin Raiciu.

10. References

[1] Ford, A., Raiciu, C., Greenhalgh, A., and M. Handley, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 6182, June 2002.

[3] Nikander, P., Arkko, J., Auro, T., Montenegro, G., and E. Nordmark, "Mobile IP Version 6 Route Optimization Security Design Background", RFC 4225, December 2005.

[4] Bangulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, March 2011.

Authors' Addresses

Georg Hampel
Alcatel-Lucent
600 Mountain Ave
Murray Hill, NJ 07974
US

Phone: +1 908 582 2377
Fax: +1 908 582 8222
Email: georg.hampel@alcatel-lucent.com

Thierry Klein
Alcatel-Lucent
600 Mountain Ave
Murray Hill, NJ 07974
US

Phone: +1 908 582 3585
Fax: +1 908 582 8222
Email: thierry.klein@alcatel-lucent.com