Internet Area Working Group V. Olteanu
Internet-Draft D. Niculescu
Intended status: Experimental University Politehnica of Bucharest
Expires: May 3, 2018 October 30, 2017

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-01

Abstract

The SOCKS protocol is used primarily to proxy TCP connections to arbitrary destinations via the use of a proxy server. Under the latest version of the protocol (version 5), it takes 2 RTTs (or 3, if authentication is used) before data can flow between the client and the server.

This memo proposes SOCKS version 6, which reduces the number of RTTs used, takes full advantage of TCP Fast Open, and adds support for 0-RTT authentication.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

Table of Contents

1. Introduction

Versions 4 and 5 [RFC1928] of the SOCKS protocol were developed two decades ago and are in widespread use for circuit level gateways or as circumvention tools, and enjoy wide support and usage from various software, such as web browsers, SSH clients, and proxifiers. However, their design needs an update in order to take advantage of the new features of transport protocols, such as TCP Fast Open [RFC7413], or to better assist newer transport protocols, such as MPTCP [RFC6824].

One of the main issues faced by SOCKS version 5 is that, when taking into account the TCP handshake, method negotiation, authentication, connection request and grant, it may take up to 5 RTTs for a data

exchange to take place at the application layer. This is especially costly in networks with a large delay at the access layer, such as 3G, 4G, or satelite.

The desire to reduce the number of RTTs manifests itself in the design of newer security protocols. TLS version 1.3 [I-D.ietf-tls-tls13] defines a zero round trip (0-RTT) handshake mode for connections if the client and server had previously communicated.

TCP Fast Open [RFC7413] is a TCP option that allows TCP to send data in the SYN and receive a response in the first ACK, and aims at obtaining a data response in one RTT. The SOCKS protocol needs to concern itself with at least two TFO deployment scenarios: First, when TFO is available end-to-end (at the client, at the proxy, and at the server); second, when TFO is active between the client and the proxy, but not at the server.

This document describes the SOCKS protocol version 6. The key improvements over SOCKS version 5 are:

- The client sends as much information upfront as possible, and does not wait for the authentication process to conclude before requesting the creation of a socket.
- The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the payload for the initial SYN that is sent out to the server.
- The protocol can be extended via options without breaking backward-compatibility.
- The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

1.1. Revision log

draft-01

- Added this section.
- Support for idempotent commands.
- Removed version numbers from operation replies.
- Request port number for SOCKS over TLS. Deprecate encryption/ encapsulation within SOCKS.

- Added Version Mismatch Replies.
- Renamed the AUTH command to NOOP.
- Shifted some fields to make requests and operation replies easier to parse.

2. 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 [RFC2119].

3. Mode of operation

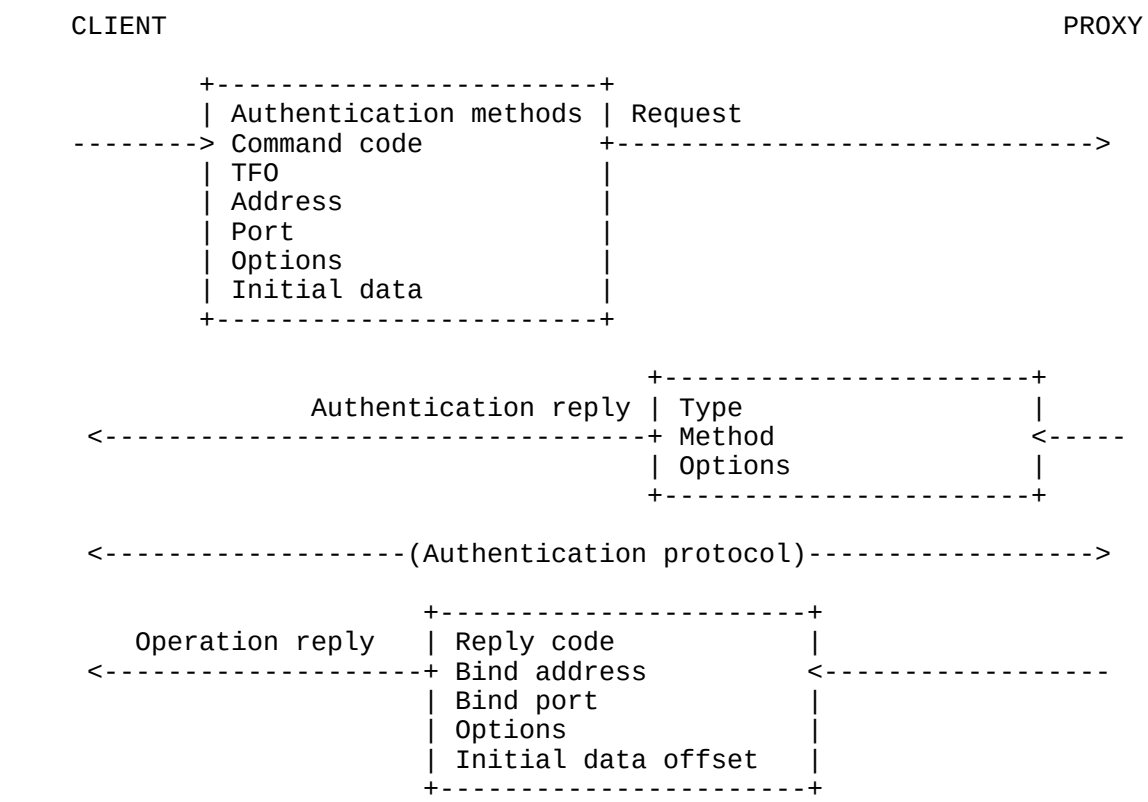

Figure 1: The SOCKS version 6 protocol message exchange

When a TCP-based client wishes to establish a connection to a server, it must open a TCP connection to the appropriate SOCKS port on the SOCKS proxy. The client then enters a negotiation phase, by sending the request in figure Figure 1, that contains, in addition to fields present in SOCKS 5 [RFC1928], fields that facilitate low RTT usage and faster authentication negotiation.

Next, the server sends an authentication reply. If the request did not contain the necessary authentication information, the proxy indicates an authentication method that must proceed. This may trigger a longer authentication sequence that could include tokens for ulterior faster authentications. The part labeled "Authentication protocol" is specific to the authentication method employed and is not expected to be employed for every connection between a client and its proxy server. The authentication protocol typically takes up 1 RTT or more.

If the authentication is successful, an operation reply is generated by the proxy. It indicates whether the proxy was successful in creating the requested socket or not.

In the fast case, when authentication is properly set up, the proxy attempts to create the socket immediately after the receipt of the request, thus achieving an operational conection in one RTT (provided TFO functionality is available at the client, proxy, and server).

4. Connection Requests

The client starts by sending a request to the proxy.

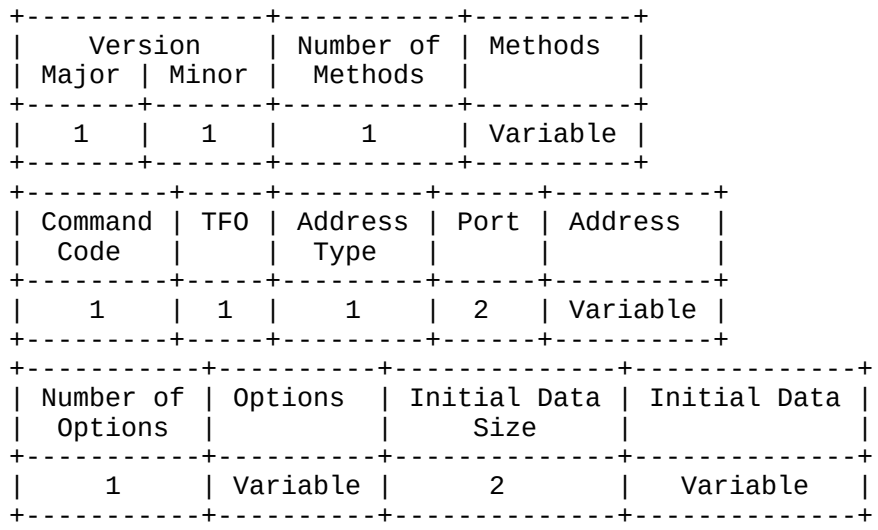

Figure 2: SOCKS 6 Request

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- Number of Methods: The number of supported authentication methods that the client wishes to advertise.
- Methods: One byte per advertised method. Method numbers are assigned by IANA.
- Command Code:
  - 0x00 NOOP: authenticate the client and do nothing.
  - 0x01 CONNECT: requests the establishment of a TCP connection.
  - 0x02 BIND: requests the establishment of a TCP port binding.
  - 0x03 UDP ASSOCIATE: requests a UDP port association.
- TFO:
  - 0x00 indicates that the proxy MUST NOT attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. In case of an AUTH or UDP ASSOCIATE command, this field MUST be set to 0x00.

  * 0x01 indicates that the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command.

- Address Type:
  * 0x01: IPv4
  * 0x03: Domain Name
  * 0x04: IPv6
- Address: this field's format depends on the address type:
  * IPv4: a 4-byte IPv4 address
  * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
  * IPv6: a 16-byte IPv6 address
- Port: the port in network byte order.
- Number of Options: the number of SOCKS options that appear in the Options field.
- Options: see section Section 8.
- Initial Data Size: A two-byte number in network byte order. In case of AUTH, BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, this is the number of bytes of initial data that are supplied in the following field.
- Initial Data: The first octets of the data stream.

Clients MUST support the "No authentication required" method. Clients MAY omit advertising the "No authentication required" option.

Clients SHOULD NOT issue AUTH commands unless they advertise authentication methods with support for 0-RTT authentication.

The server MAY truncate the initial data to an arbitrary size and disregard the rest. This is will be communicated later to the client, should the authentication process be successful (see section Section 7). As such, server implementations do not have to buffer the initial data while waiting for the (potentially malicious) client to authenticate.

5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6.0, the proxy sends the following response:

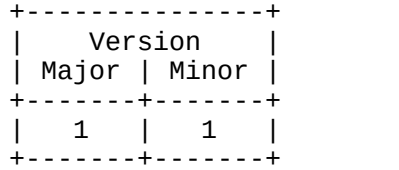

Figure 3: SOCKS 6 Version Mismatch Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

A client MUST close the connection after receiving such a reply.

6. Authentication Replies

Upon receipt of a valid request, the proxy sends an Authentication Reply:

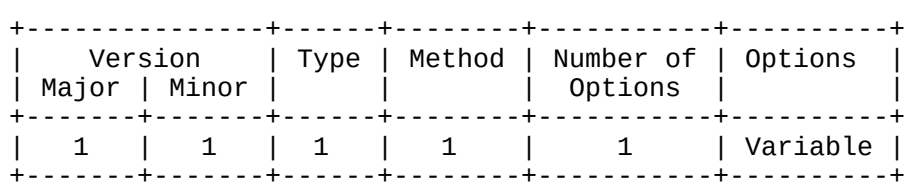

Figure 4: SOCKS 6 Authentication Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- Type:
  - 0x00: authentication successful.
  - 0x01: further authentication needed.
- Method: The chosen authentication method.
- Number of Options: the number of SOCKS options that appear in the Options field.

o Options: see section Section 8.

Multihomed clients SHOULD cache the chosen method on a per-interface basis and SHOULD NOT include authentication options related to any other methods in further requests originating from the same interface.

If the server signals that further authentication is needed and selects "No Acceptable Methods", the client MUST close the connection.

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy MAY supply information that allows the client to authenticate a future request using an authentication option. Descriptions of such negotiations are beyond the scope of this memo.

If the cliend issued an AUTH command, the client MUST close the connection after the negociation is complete.

7. Operation Replies

After the authentication negotiations are complete, the server sends an Operation Reply:

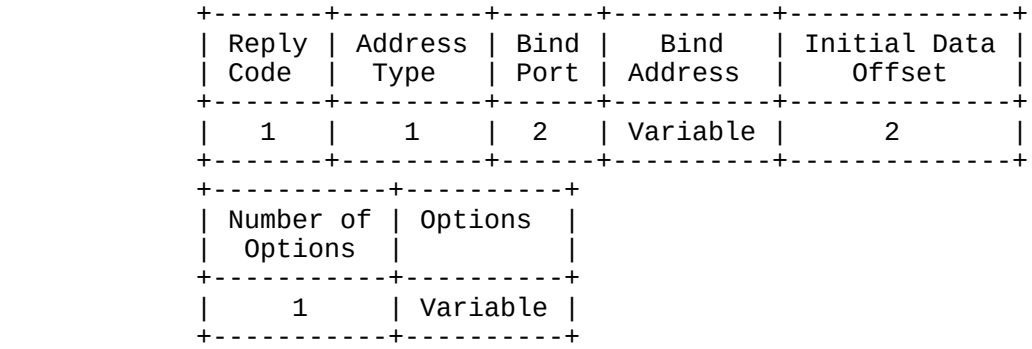

Figure 5: SOCKS 6 Operation Reply

o Reply Code:

* 0x00: Succes

* 0x01: General SOCKS server failure

* 0x02: Connection not allowed by ruleset

    * 0x03: Network unreachable
    * 0x04: Host unreachable
    * 0x05: Connection refused
    * 0x06: TTL expired
    * 0x07: Command not supported
    * 0x08: Address type not supported

- Address Type:
    * 0x01: IPv4
    * 0x03: Domain Name
    * 0x04: IPv6
- Bind Address: the proxy bound address in the following format:
    * IPv4: a 4-byte IPv4 address
    * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
    * IPv6: a 16-byte IPv6 address
- Bind Port: the proxy bound port in network byte order.
- Number of Options: the number of SOCKS options that appear in the Options field.
- Options: see section Section 8.
- Initial Data Offset: A two-byte number in network byte order. In case of BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, it represents the offset in the plain data stream from which the client is expected to continue sending data.

If the proxy returns a reply code other than "Success", the client MUST close the connection.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass. The client MUST resume the data stream at the offset indicated by the Initial Data Offset field.

7.2. Handling BIND

In case the client has issued a BIND request, it must wait for a second Operation reply from the proxy, which signifies that a host has connected to the bound port. The Bind Address and Bind Port fields contain the address and port of the connecting host. Afterwards, application data may pass.

7.3. Handling UDP ASSOCIATE

The relay of UDP packets is handled exactly as in SOCKS 5 [RFC1928].

8. SOCKS Options

SOCKS options have the following format:

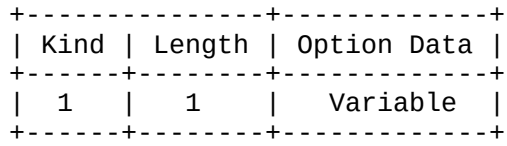

Figure 6: SOCKS 6 Option

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: The length of the option.
- Option Data: The contents are specific to each option kind.

8.1. Authentication options

Authentication options carry method-specific authentication data. They can be part of SOCKS Requests and Authentication Replies.

Authentication options have the following format:

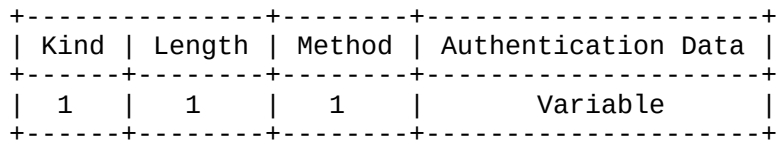

Figure 7: Authentication Option

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: The length of the option.
- Method: The number of the authentication method. These numbers are assigned by IANA.
- Authentication Data: The contents are specific to each method.

All proxy implementations MUST support authentication method options. Clients MAY omit advertising authentication methods for which they have included at least an authentication option.

8.2. Idempotence options

To protect against duplicate SOCKS Requests, authenticated clients can request, and then spend, idempotence tokens. A token can only be spent on a single SOCKS request.

Tokens are 4-byte unsigned integers in a modular 4-byte space. Therefore, if x and y are tokens, x is smaller than y if (y - x) < 2^31 in unsigned 32-bit arithmetic.

Proxies grant contiguous ranges of tokens called token windows. Token windows are defined by their base (the first token in the range) and size. Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Requesting and spending tokens is done via Idempotence options:

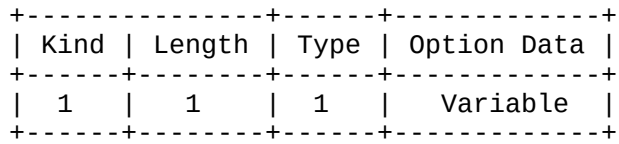

Figure 8: Idempotence Option

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: The length of the option.
- Type:
  - 0x00: Token Request
  - 0x01: Token Window Advertisement
  - 0x02: Token Expenditure
  - 0x03: Token Expenditure Reply
- Option Data: The contents are specific to each type.

8.2.1. Requesting a fresh token window

A client can obtain a fresh window of tokens by sending a Token Request option as part of a SOCKS Request:

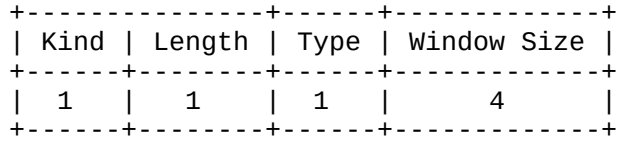

Figure 9: Token Request

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: 7
- Type: 0x00 (Token Request)
- Window Size: The requested window size.

The proxy then includes a Token Window Advertisement option in the corresponding Operation Reply:

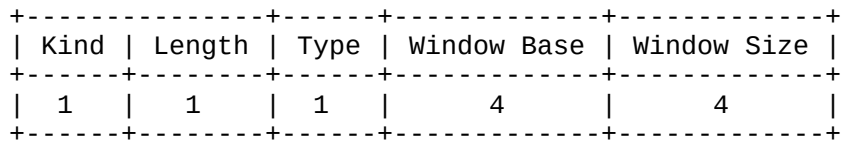

Figure 10: Token Window Advertisement

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: 11
- Type: 0x01 (Token Grant)
- Window Base: The first token in the window.
- Window Size: The window size. This value SHOULD be lower or equal to the requested window size.

8.2.2. Spending a token

The client can attempt to spend a token by including a Token Expenditure option in its SOCKS request:

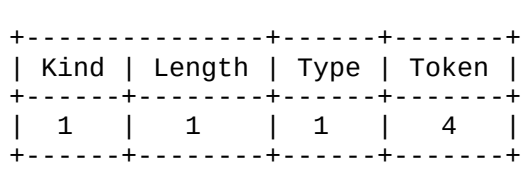

Figure 11: Token Expenditure

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: 7
- Type: 0x02 (Token Expenditure)
- Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The server responds by sending a Token Expenditure Reply option as part of the Operation Reply:

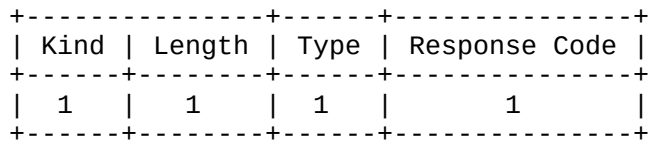

Figure 12: Token Expenditure Response

- Kind: MUST be allocated by IANA. (See section Section 10.)
- Length: 4
- Type: 0x03 (Token Expenditure Response)
- Response Code:
  - 0x00: Success: The token was spent successfully.
  - 0x01: No Window: The proxy does not have a token window associated with the client.
  - 0x02: Out of Window: The token is not within the window.
  - 0x03: Duplicate: The token has already been spent.

If eligible, the token is spent as soon as the client authenticates. If the token is not eligible for spending, the proxy MUST NOT attempt to honor the client's SOCKS Request; further, it MUST indicate a General SOCKS server failure in the Operation Reply.

Proxy implementations SHOULD also send a Token Window Advertisement if:

- the token is out of window, or
- by the proxy's internal logic, successfully spending the token caused the window to shift.

Proxy implementations SHOULD NOT shift the window's base beyond the highest unspent token.

Proxy implementations MAY include a Token Window Advertisement in any Operation Reply.

8.2.3. Handling Token Window Advertisements

Even though the proxy increases the window's base monotonically, there is no mechanism whereby a SOCKS client can receive the Token Window Advertisements in order. As such, clients SHOULD disregard unsollicited Token Window Advertisements with a Window Base less than the previously known value.

9. Security Considerations

9.1. Large requests

Given the format of the request message, a malicious client could craft a request that is in excess of 100 KB and proxies could be prone to DDoS attacks.

To mitigate such attacks, proxy implementations SHOULD be able to incrementally parse the requests. Proxies MAY close the connection to the client if:

- the request is not fully received after a certain timeout, or
- the number of options exceeds an imposed hard cap, or
- the total size of the options exceeds an imposed hard cap, or
- the size of the initial data excedes a hard cap.

Further, the server MAY choose not to buffer any initial data beyond what would be expected to fit in a TFO SYN's payload.

9.2. Replay attacks

In TLS 1.3, early data (which is likely to contain a full SOCKS request) is prone to replay attacks.

While Token Expenditure options can be used to mitigate replay attacks, the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data when sending a Token Request.

10. IANA Considerations

This document requests that IANA allocate option codes for SOCKS 6 options. Further, this document requests option codes for authentication and idempotence options.

This document also requests that IANA allocate a port for SOCKS over TLS.

11. Acknowledgements

The protocol described in this draft builds upon and is a direct continuation of SOCKS 5 [RFC1928].

12. References

12.1. Normative References

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

12.2. Informative References

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-21 (work in progress), July 2017.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <https://www.rfc-editor.org/info/rfc1928>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

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

Authors' Addresses

Vladimir Olteanu
University Politehnica of Bucharest

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest

Email: dragos.niculescu@cs.pub.ro

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 2, 2018

D. Purkayastha
M. Perras
A. Rahman
InterDigital Communications, LLC

October 29, 2017

Considerations for MPTCP operation in 5G
draft-purkayastha-mptcp-considerations-for-nextgen-00

Abstract

MPTCP is deployed in devices which have multiple interfaces to different networks. It takes advantage of these multiple interfaces to improve user experience by bandwidth aggregation, redundancy, and increased reliability by having a fallback option. This document describes scenarios where next generation (5G) mobility management frameworks may impact MPTCP operations thus potentially jeopardizing achievement of some of these gains.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 2, 2018.

Table of Contents

1. Introduction

Utilization of MPTCP in devices with multiple interfaces is very attractive. MPTCP is being deployed and widely adopted in today's smart devices, which typically have multiple network interfaces such as Cellular and Wifi. It provides reliability, bandwidth aggregation capability, and handover efficiency. MPTCP assumes that an interface is associated to only one IP address. This document describes next generation mobility management frameworks that may assign more than one IP address (to a given user device) over a single interface. In this scenario, MPTCP may not produce desired results.

This document describes the anticipated difficulties for MPTCP when working on devices which are being managed by modern mobility management protocols. The goal of this document is to make the WG aware of the upcoming situation and gauge interest in working to preserve MPTCP efficiency.

2. MPTCP background

Multipath TCP is an extension to TCP allowing hosts to use multiple paths to send and receive the data belonging to one connection. The simultaneous use of these multiple paths (sub-flows) for a TCP/IP session improves resource usage within the network and, thus, improves user experience through higher throughput and improved resilience e.g. to network failure.

A Multipath TCP connection is composed of several TCP connections that are called sub-flows. MPTCP manages the creation, removal, and utilization of these sub-flows to send data.

Once an MPTCP connection has begun, further sub-flows can be added to the connection. Hosts have knowledge of their own addresses, and can become aware of the other host's addresses through signaling exchanges. If extra paths are available, additional TCP sessions ("sub-flows") are created on these paths, and are combined with the existing MPTCP session, which continues to appear as a single connection to the applications at both ends. Standard TCP versus MPTCP protocol stacks are illustrated in Figure 1, from [RFC6824].

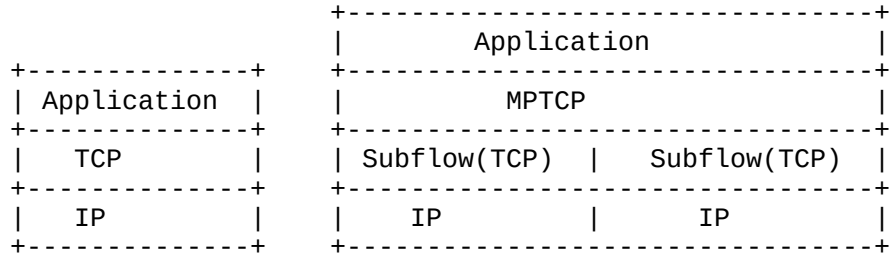

Figure 1: Standard TCP vs MPTCP Protocol Stacks

3. Mobility management protocols

Mobility management is needed because the IP address of a mobile node may change as the node moves. A fair number of network-layer (IP) mobility protocols have been standardized. They all employ a mobility anchor to allow a mobile node to remain reachable after it has moved to a different network. 3GPP 5G has adopted distributed approach and pushed the mobility anchor towards the edge of the network.

3.1. Centralized mobility management (e.g. DSMIP, PMIP)

Existing network-layer mobility management protocols have primarily employed a mobility anchor to ensure connectivity of a mobile node by forwarding packets destined to, or sent from, the mobile node after the node has moved to a different network. The mobility anchor has been centrally deployed in the sense that the traffic of millions of mobile nodes in an operator network is typically managed by the same anchor. Redirecting packets this way can result in long routes.

3.2. Mobility Handling in 5G

3GPP standards organization is currently defining 5G network architecture and systems. This is referred to as multi-homed PDU Session.

In 5G networks, mobility framework under development suggest that IP mobility, network access solutions, and forwarding solutions should enable traffic to avoid traversing a single mobility anchor far from the optimal route. Mobility anchors are distributed in such a way that the mobility anchors are positioned closer to the user; ideally, mobility agents could be collocated with the first-hop router.

The current definition specifies that a PDU (packet data unit) Session may be associated with multiple IPv6 prefixes. The PDU Session provides access to the Data Network via more than one PDU (IPv6) anchor. The different user plane paths leading to the IP anchors branch out at a "common" UPF (user plane function) referred to as a UPF supporting "Branching Point" functionality. The Branching Point (BP) provides forwarding of uplink traffic towards the different IP anchors and merge of downlink traffic to the mobile device i.e. merging the traffic from the different IPv6 anchors on the link towards the device. The BP and UPFs (IP anchors) are illustrated on Figure 2, as described in [_3GPP.23.501].

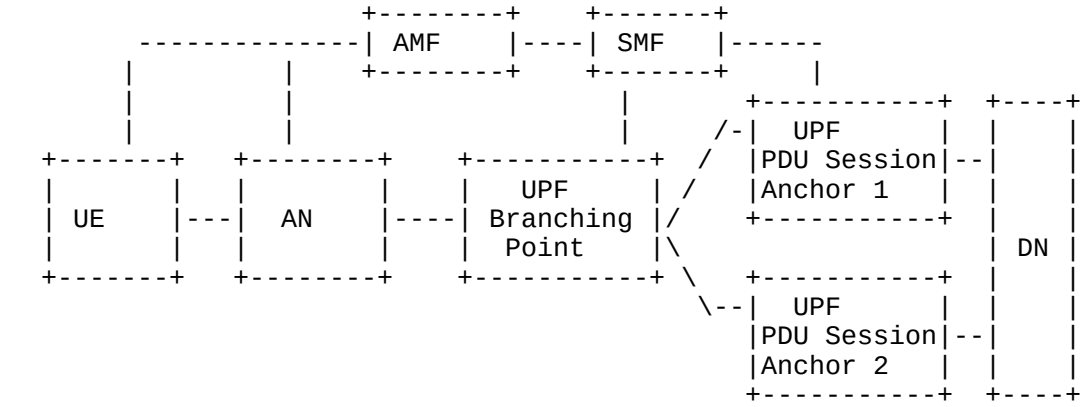

Figure 2: Multi-Homed PDU Session: Service Continuity Case

4. Considerations for MPTCP

Using MPTCP (as currently defined) in 5G network, which may assign multiple IP addresses (to a given device) on the same interface, may need special consideration.

As we know, MPTCP currently interprets multiple IP addresses as indicating separate distinct network interfaces. This incorrect interpretation may lead to not realizing fully the benefit of MPTCP applications such as bandwidth aggregation, redundancy, and reliability in 5G systems.

5. MPTCP operation use case

5.1. BW management increase BW as needed and load balancing features

One of the main goal of using MPTCP is to increase the BW associated to a TCP session when needed. The additional BW is obtained by creating sub-flows over interfaces other than the one already in use (e.g. using different IP addresses) and associating these sub-flows to the existing TCP session, which altogether form the MPTCP session. The total BW available for an MPTCP session is the accumulation of all the sub-flows associated to the same MPTCP session.

In situations, when MPTCP sub-flows are created using IP addresses associated to the same interface, MPTCP operation is impacted. Adding up sub-flow using the same interface doesn’t increase the available BW (considering that the bottleneck may be between the device and the PoA). In this case, MPTCP may continue to add more sub-flows but there may not be proportional gain in bandwidth. In this context, MPTCP functionality is inefficient; the expected MPTCP behavior not being met.

Similar situation arises for load balancing. MPTCP may try to add sub flows associated to the same interface and attempt to distribute the load. From operational perspective, we may not see proportional gain in load balancing.

5.2. Backup path management - created in advance to be available when needed

MPTCP may associate sub-flows to an MPTCP session with one or many of these sub-flows being identified as "backup" sub-flows. Backup sub-flows are used only when no regular sub-flows are available (e.g. if a link failure occurs).

As described in the previous section, the backup mechanism is ineffective if the backup path is using the same interface as the

regular path. If the interface goes down, then the backup path becomes unavailable, at the same time as the regular path. In this case, no backup path is ready to handle data traffic. New sub-flows will need to be created to backup the regular sub-flow, which is unavailable. Packets loss and retransmissions will become inevitable resulting in bad user experience.

6. IANA Considerations

This document requests no IANA actions.

7. Security Considerations

TBD

8. Informative References

[_3GPP.23.501] 3GPP, "System Architecture for the 5G System", 3GPP TS 23.501 1.4.0, 9 2017, <http://www.3gpp.org/ftp/Specs/html-info/23501.htm>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

Authors' Addresses

Debashish Purkayastha
InterDigital Communications, LLC
Conshohocken
USA

Email: Debashish.Purkayastha@InterDigital.com

Michelle Perras
InterDigital Communications, LLC
Montreal
Canada

Email: Michelle.Perras@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com

MPTCP Working Group
Internet-Draft
Intended Status: Standards Track
Expires: April 29, 2018

F. Wang
J. Zuo
Z. Cao
K. Zheng
Huawei

October 30, 2017

A Proactive Approach to Avoid Performance Degradation of MPTCP
draft-zuo-mptcp-degradation-00

Abstract

One of the goals for MPTCP is utilizing multiple paths to perform at least as well as the best path in terms of throughput. However, this goal might not be arrived at because of the path asymmetry, which is called as the performance-degradation problem of MPTCP in this draft. Some existing methods focus on this problem, such as penalizing and opportunistic retransmission, which reactively responds to the head-of-line blocking for trying their best to send data across all paths. In order to efficiently utilize the capabilities of the multiple paths, this draft proposes an approach that proactively selects the best path(s) to send data instead of always bonding all paths together.

Table of Contents

1. Introduction

MultiPath TCP (MPTCP) enables a transport connection across multiple paths simultaneously [RFC6824]. According to [RFC6356], one of the MPTCP's goals is improving throughput: a multipath flow should perform at least as well as a single path flow would on the best of the path available to it. However, this goal cannot be always achieved due to the head-of-line blocking caused by the path asymmetry, e.g. WiFi and LTE in smart phones. To be convenient, this phenomenon (that MPTCP performs worse than the best path) is called as the performance-degradation problem in this draft.

The direct solution for this problem is allocating a large enough receive buffer. As in [RFC6182], 'The RECOMMENDED receive buffer is

2*sum(BW_i)*RTT_max, which ‘ensures subflows do not stall when fast retransmit is triggered on any subflow’. However, the buffer size can be very large to cover all the possible scenarios. In other words, the buffer size can be limited and the performance-degradation problem is possible to exist. Therefore, it needs a solution for MPTCP protocol that can solve this problem. On this issue, some reactive methods have been proposed, such as penalizing and opportunistic retransmission. They take actions after the head-of-line blocking occurring and their purpose is to send data across all paths as possible as they can. Experiments show that even with these methods, the capabilities of the multiple paths may not be efficiently utilized. Meanwhile, instead of bonding all paths together, [RFC6182] indicates that it would be better ‘to only use some of the fastest available paths for the MPTCP connection in extreme cases’.

This draft focuses on this problem and proposes a proactive approach, which dynamically employs part or all of the paths for higher utilization efficiency. In particular, this approach first measures the aggregated throughput of the multiple paths and the throughput of the best single path in real time. Then, if the performance-degradation phenomenon is derived through the throughput comparison, only the best path is used to send data.

2. Acronyms and Terminology

MPTCP: Multiple Path Transport Control Protocol

RTT: Round-robin Transmit Time

PLR: Packet Loss Ratio

BW: BandWidth

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

3. A Proactive Approach to Avoid MPTCP Performance Degradation

This section introduces the basic principles of the proposed approach: 1) how to efficiently measure the characteristics of multiple paths; 2) how to select the path(s). More details regarding to this approach can be found in Section 4. Briefly, through measuring and comparing the path characteristics, the best path(s) is selected to satisfy a special purpose, such as the high throughput or

low RTT. In theory, path characteristics can be RTT, throughput, congestion window, etc., while only throughput is considered in this draft for a case study. As shown in Figure 1, the proposed approach is implemented in the Performance-Degradation-Free Module of the MP layer.

3.1 Throughput Measurement

This section describes the method of the throughput measurement for multipath transmissions. As shown in [CMT-SCTP], one possible method is modeling the aggregated throughput of multiple paths and measuring the related path characteristics, i.e. buffer size, PLR, BW and RTT. However, it is hard to accurately model the throughput, since 1) lacking of BW evaluation methods, 2) and the small PLR can be obtained only after a huge number of data exchanging.

This draft proposes a method that directly measures the multipath and best-single-path throughputs, in order to avoid the measurements of BW and PLR. At first, two modes are defined for the throughput measurement:

1) Redundant transmission mode: Multiple paths transmit the same data;

2) MultiPath (MP) transmission mode: Multiple paths transmit different data with using a MPTCP scheduling scheme, such as minRTT (the default scheduling method) in version 0.92. By the way, the opportunistic retransmission and penalization mechanisms can be enabled in MP mode.

The redundant transmission mode is a kind of scheduling scheme in MPTCP v0.92, which is usually used to achieve lower packet delay than that of a single path. Because of wasting bandwidth (BW), the redundant transmission mode is ignored when MPTCP aggregates the BWs of multiple paths. However, this draft utilizes the redundant transmission mode to measure the throughputs of each path and the best single path, while using the multipath transmission mode to measure the aggregated throughput. To be convenient, the redundant/MP transmission mode is also called as the redundant/MP mode in the following part of this draft. Moreover, the throughput measurement is executed during the data transmission without introducing extra packets, and is periodic for self-adapting to the dynamic network environment.

The Figure 1 briefly shows the relationship between the two transmission modes and our Performance-Degradation-Free Module in MP layer.

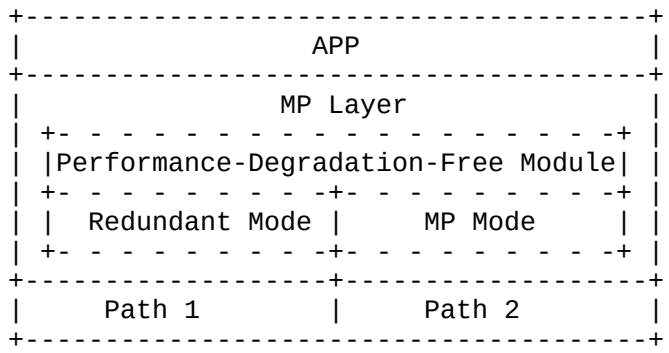

Figure 1: The performance-degradation-free module in the MP layer.

3.2 Path Selection Strategy

All these paths or the best one of them are used to send data depending on the corresponding throughput.

4. Operation Overview

Section 3 presents the main principles, i.e. throughput measurement and path selection. This section introduces how the proposed approach works in details. According to different congestion-control schemes, the proposed approach may have slow-start stage and congestion-control stage. At the first stage, the approach ensures that the slow start of the scheme has been finished. At the second stage, the approach periodically measures the average throughputs of multiple paths and the best path, and employs the suitable path(s) through throughput comparisons.

4.1 Slow-Start Stage

During the slow-start stage, MPTCP schedules the data in the redundant mode, where the same data are transmitted across multiple paths. After each round, we get a measured throughput from the view of MP layer, where the round time could be from sending a packet until receiving its responsive ACK. The throughput (defined as ‘B_rd’) is calculated by dividing the total number of delivered data with the time period of this round. We define a threshold ‘h’, where h belongs to (0,1), e.g. h=0.2. As shown in Figure 2, if the measured throughput is h times larger than the last measured value (i.e. B_rd(i) >= h * B_rd(i-1)) , keep the redundant mode and set the round counter Nr as 0, or else increase the round counter (defined as Nr) by 1.

4.2 Congestion-Avoidance Stage

If the round counter Nr > N (e.g. N = 3), then the transmission switches to the congestion-avoidance stage, as shown in Figure 2. Two timers are set to decide when we need measure the throughputs of the redundant and MP modes again. Two indicators of updating the redundant and MP measurement values, Indicator_Redundant and Indicator_MP, are both set as 1. The indicator timers for the redundant mode and MP modes are set as REDUNDANT_MODE_TIMER and MP_MODE_TIMER, separately. The timeout of both timers is set as a period of time TIMEOUT (e.g. 10s).

At first, the redundant mode is used to measure the throughput.

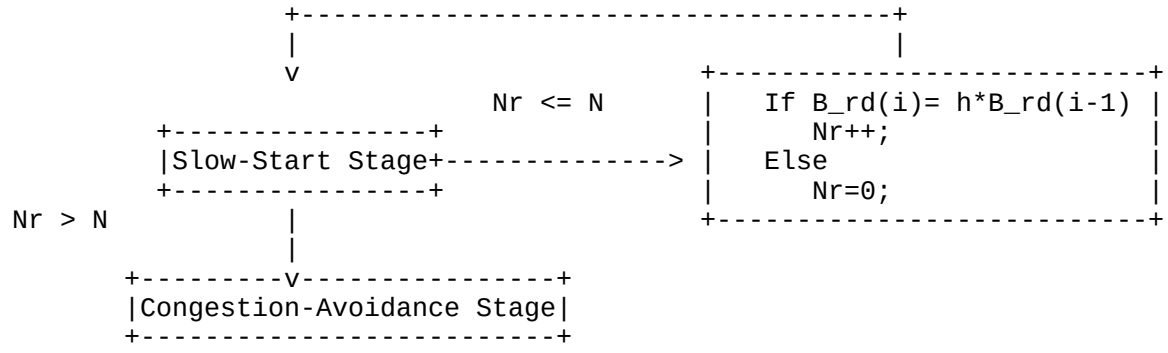

Figure 2: Stage transition from slow start to congestion avoidance.

4.2.1 Redundant Transmission Mode

As shown in Figure 3, when running the redundant mode, MP layer and each path would calculate the average throughputs during a period of time (e.g. 1s). Meanwhile, the timer REDUNDANT_MODE_TIMER is reset and the variable Indicator_Redundant is set as 0. Before sending each segment, if Indicator_MP = 1, the redundant mode is switched to the MP mode to measure the throughput, as described in Section 4.2.2. Otherwise, the two throughputs obtained from the redundant and MP modes are compared, which is introduced in Section 4.2.3.

Figure 3: The redundant-and-MP mode transition when the corresponding indicator is equal to 1.

4.2.2 Multipath Transmission Mode

As shown in Figure 3, when running the MP mode, MP layer would calculate its average throughput during a period of time (e.g. 1s). Meanwhile, the timer MP_MODE_TIMER is reset and the variable Indicator_MP is set as 0. Before sending each segment, if Indicator_Redundant = 1, the MP mode is switched to the redundant mode to measure the throughputs (refer to Section 4.2.1). Otherwise,

the two throughputs obtained from the redundant and MP modes are compared (see Section 4.2.3).

4.2.3 Throughput Comparison

By comparing the throughputs measured by the redundant and MP modes, a transmission mode corresponding to the larger throughput would be selected.

If the redundant mode (described in Section 4.2.1) is selected, the

path with the highest throughput is employed for data transmission. Before sending each segment, if Indicator_redundant = 1 or Indicator_MP = 1, we will go back to the redundant mode or the MP mode to measure the corresponding throughput(s). If both of the indicators are 0, keep transmitting data at the best path.

If the MP mode (described in Section 4.2.2) is selected, the multiple paths are bonded together to achieve the aggregated throughput.

4.2.4 Indicator Timer Timeout

During data transmission, if the timer for the redundant/MP mode is timeout, set Indicator_Redundant/Indicator_MP = 1 and reset the timer of the redundant/MP mode.

5. Security Considerations

TBD.

6. IANA Considerations

7. References

7.1. Normative References

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

[RFC6356] Raiciu, C., Handly, M., and Wischikf, D., "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://rfc-editor.org/rfc/rfc6356.txt>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and Bonaventure, O., "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., Iyengar, J., "Architectural Guidelines for Multipath TCP Development", RFC 6182, DOI 10.17487/RFC6182, March 2011,

<http://www.rfc-editor.org/info/rfc6182>.

7.2. Informative References

[CMT-SCTP] Yang, W., Li, H., Li, F., Wu, Q., & Wu, J., "RPS: range-based path selection method for concurrent multipath transfer", June 2010, In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (pp. 944-948). ACM.

Author’s Addresses

Fanzhao Wang
Huawei Technologies
Bantian, Longgang District,
Shenzhen 518129 P.R. China
EMail: wangfanzhao@huawei.com

Jing Zuo
Huawei Technologies
Bantian, Longgang District,
Shenzhen 518129 P.R. China
EMail: jing.zuo@huawei.com

Zhen Cao
Huawei Technologies
No.156 Beiqing Rd. Haidian District,
Beijing 100095 P.R. China
EMail: zhencao.ietf@gmail.com

Kai Zheng
Huawei Technologies
No.156 Beiqing Rd. Haidian District,
Beijing 100095 P.R. China
EMail: kai.zheng@huawei.com