Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2014

P. Eardley
BT

July 12, 2013

Survey of MPTCP Implementations
draft-eardley-mptcp-implementations-survey-02

Abstract

This document presents results from the survey to gather information from people who have implemented MPTCP, in particular to help progress the protocol from Experimental to Standards track.

The document currently includes answers from four teams: a Linux implementation from UCLouvain, a FreeBSD implementation from Swinburne, an anonymous implementation in a commercial OS, and a NetScalar Firmware implementation from Citrix Systems, Inc. Thank-you!

In summary, we have four independent implementations of all the MPTCP signalling messages, with the exception of address management, and some interoperabiity testing has been done by the other three implementations with the 'reference' Linux implementation. So it appears that the RFC is (at least largely) clear and correct. On address management, we have only one implementation of ADD_ADDR with two teams choosing not to implement it. We have one implementation of the working group's coupled congestion control (RFC6356) and none of the MPTCP-aware API (RFC6897).

The main suggested improvements are around

- how MPTCP falls back (if the signalling is interrupted by a middlebox): (1) corner cases that are not handled properly, (2) at the IETF, the MPTCP community should work with middlebox vendors, either to reduce or eliminate the need for fallback or to understand the middlebox interactions better.

- security: both better MPTCP security (perhaps building on SSL) and a lighter weight mechanism, preferably both in one mechanism.

It is hoped that the next version can include information from any other implementations. If you are an implementer and want to contribute your answers, please see the -01 version of this document for a blank survey ready to be filled in.

Status of this Memo

Table of Contents

1. Introduction

The document reports the results from a survey to gather information from people who have implemented MPTCP. The goal is to help progress the protocol from Experimental to Standards track.

Four responses have been received. Thank-you! They are independent implementations:

- o the Linux implementation from UCLouvain,
- o the FreeBSD implementation from Swinburne
- o an anonymous implementation in a commercial OS
- o a NetScaler Firmware implementation from Citrix Systems, Inc.

The Table below presents a highly-compressed summary, with each row corresponding to one question or sub-question of the survey. The following section highlights some interesting aspects of the replies in less compressed form. The full survey responses are in Appendix A, B, C and D.

It is hoped that the next version of this document can include information about a further (independent) implementation:

- o Georg Hampel's user-space implementation (publicly available but not longer maintained)
- o any other implementations.

2. Survey - summary of replies

The Table below presents a highly-compressed summary, with each row corresponding to one question or sub-question of the survey. A column is left blank for any future responses.

	1	2	3	4	
Institution	UCLouvain	Swinburne	Anon	Citrix	
Question 2 asks about some preliminary topics, including whether the implementation is publicly available and interoperability with the Linux implementation (#1).					
	UCLouvain	Swinburne	Anon	Citrix	
OS	Linux	FreeBSD-10	Commercial	NetScaler	
v4 & v6	Both	IPv4	Both	Both	

public	Yes	Yes	No	Yes (pay)
independent	Yes	Yes	Yes	Yes
interop	Yes(!)	Mostly	Mostly	Yes

Question 3: Support for MPTCP's signalling functionality
MPTCP's signalling messages are: MP_CAPABLE, MP_JOIN, Data transfer (DSS), ADD_ADDR, REMOVE_ADDR, MP_FASTCLOSE. There are sub-questions for MP_JOIN and DSS.

	UCLouvain	Swinburne	Anon	Citrix
MP_CAPABLE	Yes	Yes	Yes	Yes
MP_JOIN	Yes	Yes	Yes	Yes
initiated by	first end	either end	first end	first end
#subflows	32	8	no limit	6
DSS	Yes	Yes	Yes	Yes
DATA ACK	4 bytes	4 or 8 byte	4 or 8 byte	4 or 8 byte
Data seq num	4 bytes	4 or 8 byte	4 or 8 byte	4 or 8 byte
DATA_FIN	Yes	Yes	Yes	Yes
Checksum	Yes	No	Yes	Yes
ADD_ADDR	Yes	No	No (never)	No (never?)
REMOVE_ADDR	Yes	No	Partly	Yes
FAST_CLOSE	Yes	No	Yes	Yes

Question 4 asks about fallback from MPTCP: if a middlebox mangles MPTCP's signalling by removing MP_CAPABLE, MP_JOIN, DSS or DATA_ACK; if data is protected with Checksum in DSS option; if fallback to TCP uses an infinte mapping; and if any corner cases have been found.

	UCLouvain	Swinburne	Anon	Citrix
MP_CAPABLE	Yes	Yes	Yes	Yes
MP_JOIN	Yes	Yes	Yes	Yes
DSS	Yes	No	Yes	Yes
DATA_ACK	Yes	No	No	
Checksum	Yes	No	Yes	Yes
infinte map	Yes	Yes	Yes	Yes
corner cases	No		Yes	Yes

Question 5 asks about heuristics: aspects that are not required for protocol correctness but impact the performance. Questions are about sized the receiver and sender buffers, re-transmission policy, if additional subflows use the same port number as for the first subflow

	UCLouvain	Swinburne	Anon	Citrix
Recvr buffer	auto-tune	TCP_MAXWIN	no tuning	tuned
Sendr buffer	auto-tune	cwnd	no tuning	as TCP
Re-transmits	2nd subflow	2nd subflow	2nd subflow	1st subflow
Port usage	same ports	same ports	diff local	

Question 6 asks about what security mechanisms are implemented: the one defined in RFC6824 and any others.

	UCLouvain	Swinburne	Anon	Citrix

HMAC-SHA1	Yes	Yes	Yes	Yes
other	Yes	No	No	No

Question 7 asks whether the implementation follows the IANA-related definitions (for TCP Option Kind and sub-registries).

	UCLouvain	Swinburne	Anon	Citrix
RFC6824	Yes	Yes	Yes	Yes

Question 8 asks about congestion control and related issues: how traffic is shared across multiple subflows; support for 'handover'; and support of RFC6356 (or other) coupled congestion control.

	UCLouvain	Swinburne	Anon	Citrix
sharing	shared, RTT	shared	active/back	active/back
handover	Yes		Yes	Yes
coupled cc	Yes	No	No	No
other ccc	Yes, OLIA	No	No	No
MP-PRIO & B	Yes	No	Yes	Yes

Question 9 is about the API: how legacy applications interact with the MPTCP stack, and if implemented the RFC6897 API for MPTCP-aware applications.

	UCLouvain	Swinburne	Anon	Citrix
legacy apps	default	sysctl	private API	configured
MPTCP API	No	No	No	No
advanced API	No	No	No	No

Question 10 gathers some limited information about operational experiences and deployments.

	UCLouvain	Swinburne	Anon	Citrix
Scenario	several	several	mobile	proxy
environment	internet	controlled	internet	internet
ends / proxy	end hosts	end hosts	end hosts	proxy

3. Interesting aspects of replies

This section tries to highlight some interesting comments made in the surveys. The Appendices can be consulted for further detials.

3.1. Question 1: Your details

Implementation 1 has been implemented by Sebastien Barre, Christoph Paasch and a large team, mainly at UCLouvain. Implementation 2 has been implemented by Lawrence Stewart and Nigel Williams at Swinburne University of Technology. Both these implementations are publicly available. Implementation 3 comes from an anonymous team with a commercial OS. Implementation 4 comes from Citrix Systems, Inc.

3.2. Question 2: Preliminary information about your implementation

Three of the four implementations are publicly available, two for free (under GPLv2 and BSD licences) and one for a fee (NetScaler Firmware). Implementation 3 (commercial OS) is planned for use in a mobile environment, with MPTCP is used in active/backup mode.

All implementations support IPv4 and three of four support IPv6.

All implementations are being actively worked on, in order to improve performance and stability and conformance with the RFC.

3.3. Question 3: Support for MPTCP's Signalling Functionality

Three of the four implementations have implemented all the MPTCP signalling, with the interesting exception of address management, whilst Implementation 2 plans to add support for all those signalling capabilities it does not yet support.

On address management, two implementations have decided not to implement ADD_ADDR. (ADD_ADDR allows an MPTCP host to signal a new address explicitly to the other host to allow it to initiate a new subflow - as an alternative to using MP_JOIN to directly start a new subflow). Implementation 3 decided not to support sending ADD_ADDR or processing ADD_ADDR as it is considered a security risk. Implementation 4 decided not to support ADD_ADDR because it didn't think it would be useful as most clients are behind NATing devices. However, both implemented REMOVE_ADDR (in Implementation 3 the client can send a REMOVE_ADDR but ignores incoming REMOVE_ADDR).

In Implementations 1, 3 and 4 only the initiator of the original subflow can start a new subflow (a reason mentioned is that NATs make it hard for the server to reach the client).

All implementations support 4 bytes "Data ACK" and "Data sequence number" fields, and will interoperate with an implementation sending 8 bytes. Implementation 1 uses only 4 bytes fields; if an implementation sends an 8 byte data sequence number it replies with a 4 byte data ack.

3.4. Question 4: Fallback from MPTCP

Question 4 asks about action when there is a problem with MPTCP, for example due to a middlebox mangling MPTCP's signalling. The connection needs to fall back: if the problem is on the first subflow then MPTCP falls back to TCP, whilst if the problem is on an additional subflow then that subflow is closed with a TCP RST, as discussed in [Section 3.6 RFC6824].

Implementations 3 and 4 made several comments about fallback.

Implementation 3 suggests that both sender and receiver behaviours could be outlined with more detail, in particular when DSS checksum is not in use and the MPTCP options are stripped. Implementation 3 falls back to TCP when there's one sub flow, but not when there are multiple sub flows (MPTCP is used in active/backup mode, and it is assumed that the sub flow transferring data is most likely to be more usable than any other established sub flow, hence the sub flow on which fallback occurred is kept alive and other sub flows are closed).

Implementation 4 found a corner case where it is not clear what to do: if a pure ack or data packet without DSS is received in middle of transaction (which can happen if the routing changes and the new path drops MPTCP options). Also, Implementation 4 suggests that clarifying whether the infinite map exchange is unidirectional or bidirectional.

Implementation 1 has developed a publicly available test suite that tests MPTCP's traversal of middleboxes.

3.5. Question 5: Heuristics

Question 5 gathers information about heuristics: aspects that are not required for protocol correctness but impact the performance. We would like to document best practice so that future implementers can learn from the experience of pioneers.

There are several differences between the implementations.

For receiver buffer, Implementation 1 uses a slightly modified version of Linux's auto-tuning algorithm; Implementation 2 determines the receiver buffer by using "TCP_MAXWIN << tp->rcv_scale" (this is a temporary measure); Implementation 3 uses MPTCP in active/backup mode, so the receive buffer sizes at the MPTCP and subflow level is the same (automatic buffer tuning is turned off); Implementation 4 varies the receiver buffer size based on the services and application type.

For the sender buffer, Implementation 1 uses Linux auto-tuning, Implementation 2 scales based on occupancy, whilst Implementation 3 turns off automatic buffer tuning, and Implementation 4 uses MPTCP-level (sub)flow control that is (almost) the same as regular TCP flow control.

Implementations 1, 2 and 3 re-transmit unacknowledged data on a different subflow (and not the same subflow), whilst Implementation 4 re-transmits on original subflow for 3 RTOs and then uses another subflow.

For port usage, Implementations 1 and 2 uses the same ports for the additional subflows, whilst Implementation 3 uses the same detsination port but a different local port, so that on the wire it looks like two connections to the same remote destination.

Implementation 4 suggests that the RFC should more clearly /extensively define failure cases and how to handle unexpected signals.

3.6. Question 6: Security

Question 6 asks about security related matters.

All Implementations have implemented the hash-based, HMAC-SHA1 security mechanism defined in [RFC6824]. Implementation 3 suggests that a more secure mechanism could be tied with SSL. Implementation 4 suggests that a more secure and lightweight mechanism is needed, as keys are exchanged (in the MP_CAPABLE option) in plain text and the key generation mechanism is highly computational intensive. Implementation 1 has implemented two additional mechanisms in a separate Linux branch - one lightweight and the other SSL-based.

3.7. Question 7: IANA

All Implementations have followed the IANA-related definitions [Section 8 RFC6824] for: TCP Option Kind number (30); the sub-registry for "MPTCP Option Subtypes"; and the sub-registry for "MPTCP Handshake Algorithms".

3.8. Question 8: Congestion control and subflow policy

Question 8 asks how is shared across multiple subflows.

Implementation 1 has added support for coupled congestion control (both that defined in [RFC6356] and in OLIA, draft-khalili-mptcp-congestion-control. The other implementations do not include coupled congestion control. Whilst Implementation 2 plans to add it (currently it uses a simple algorithm spreads traffic across the subflows), Implementations 3 and 4 do not plan to add coupled congestion control - they use one subflow at a time, with others as a backup. Implementation 3 believes it is not currently useful to share load across all network interfaces on a mobile node, as the interfaces have different characteristics for cost, bring-up and power usage. They have both found the B bit (in MP-JOIN) and MP-PRIO option very useful for this active /backup operation.

Implementation 2 is also interested in experimenting with congestion control across paths with different path-cost metrics.

3.9. Question 9: API

Question 9 gathers information about the API. None have implemented the [RFC6897] "basic MPTCP API" for MPTCP-aware applications. For three implementations MPTCP is used for all applications (set by configuration), whilst Implementation 3 uses a private API that allows MPTCP to be used on a per application basis.

3.10. Question 10: Deployments, use cases and operational experiences

Question 10 takes the opportunity of this survey to gather some limited information about operational experiences and deployments.

The Implementations mention different use cases.

Implementation 2 is interested in using MPTCP for several use cases: vehicle to infrastructure (V2I) connectivity (to provide a persistent connection using 3G and roadside wifi); multi-homed "home-user" environments; high throughput data transfers. Implementation 3 is interested in the mobile scenario, with MPTCP providing an active /backup mode so achieving session continuity across changing network environments. Implementation 4 is interested in MPTCP giving reliability and fault tolerance via a proxy. Implementation 1 already uses MPTCP on www.multipath-tcp.org and for internal ssh servers at UCLouvain.

Implementation 4 uses a proxy (MPTCP connections from a client are terminated and the TCP connection established on the other side), whilst the other Implementations are on end hosts. Implementation 2 is so far within controlled testbeds, whilst Implementation 3 is on the Internet.

Implementation 2 is currently an alpha-quality build, so limited testing so far.

Implementation 3 suggests working at the IETF with firewall vendors, to get them to change their defaults to allow MPTCP signals. This would also reduce the "over-engineering" needed to handle fallback cases. Implementation 1 suggests retrieving logs from middleboxes, as the best approach to understanding the interactions of MPTCP signalling with middleboxes.

Implementation 3 discusses a scenario that should be handled better. A backup subflow may never sent data. If the initial subflow fails, data is retransmitted on the backup subflow, but that path has a middlebox stripping options. Then it may not be possible to recover the MPTCP session.

3.11. Question 11: Improvements to RFC6824

Question 11 asks if there are any areas where RFC6824 could be improved. The main topics have been mentioned earlier:

- o fallback: the need for more clarity in the fallback cases is mentioned by Implementations 3 and 4.
- o security: the need for both a more secure and a more lightweight mechanism is mentioned.

Implementation 3 also suggests several potential improvements, which are outside the scope of RFC6824: support for sub flow level automatic buffer scaling, varying QoS support, and varying window scaling support on each sub flow; also, additional work on option signlling will be brought up in future discussions.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

This survey does not impact the security of MPTCP, except to the extent that it uncovers security issues that can be tackled in a future version of the protocol.

6. Acknowledgements

Many thanks to the people who replied to the survey: Christoph Paasch, Nigel Williams, anon, and Krishna Khanal. Very many thanks to all of the teams who actually did the implementation and testing and are continuing to improve them.

7. Full survey response for Implementation 1

Question 1: Your details Question 1 gathers some information about the team that has implemented MPTCP.

1. Your institution: UCLouvain, IP Networking Lab (http://inl.info.ucl.ac.be)

2. Name(s) of people in your implementation and test teams: Initial design from Sebastien Barre. Since then, numerous code-contributors (ordered by number of commits): Christoph Paasch (UCLouvain) Gregory Detal (UCLouvain) Jakko Korkeaniemi (Aalto University) Mihai P. Andrei (Intel) Fabien Duchene (UCLouvain) Andreas Seelinger (RWTH Aachen) Stefan Sicleru (Intel) Lavkesh Lahngir Catalin Nicutar (PUB Bucharest) Andrei Maruseac (Intel) Andreas Ripke (NEC) Vlad Dogaru (Intel) Octavian Purdila (Intel) Niels Laukens (VRT Belgium) John Ronan (TSSG) Brandon Heller (Stanford University) Conformance Testing: Yvan Coene (UCLouvain)

3. Do you want your answers to Question 1.1 and 1.2 above to be anonymised? No.

3.2. Question 2: Preliminary information about your implementation Question 2 gathers some preliminary information.

1. What OS is your implementation for? (or is it application layer?) Linux Kernel.

2. Do you support IPv4 or IPv6 addresses or both? We support both.

3. Is it publicly available (or will it be?) (for free or a fee?) Publicly available (GPLv2) at www.multipath-tcp.org

4. Overall, what are you implementation and testing plans? (details can be given against individual items later) We plan to continue to align our implementation with the IETF specifications and improve its performance and stability.

5. Is it an independent implementation? Or does it build on another MPTCP implementation -which one? Independent implementation.

6. Have you already done some interop tests, for example with UCLouvain's "reference" Linux implementation? /

7. Would you be prepared to take part in an interop event, for example adjacent to IETF-87 in Berlin? Yes. We are also ready to help in organising such an event if needed.

3.3. Question 3: Support for MPTCP's Signalling Functionality Question 3 asks about support for the various signalling messages that the MPTCP protocol defines. *** For each message, please give a little information about the status of your implementation: for example, you may have implemented it and fully tested it; the implementation may be in progress; you have not yet implemented it but plan to soon (timescale?); you may you have no intention to implement it (why?); etc.

1. Connection initiation (MP_CAPABLE) [Section 3.1 RFC6824]

a. What is the status of your implementation? Fully support the MP_CAPABLE exchange.

b. Any other comments or information? We generate the random key as a hash of the 5-tuple, sequence number and a local secret. This significantly improves the performance, instead of using a pseudo-random number generator. The performance benefit has been shown during IETF85
http://tools.ietf.org/agenda/85/slides/slides-85-mptcp-2.pdf

2. Starting a new subflow (MP_JOIN) [Section 3.2 RFC6824]

a. What is the status of your implementation? Fully support the MP_JOIN exchange.

b. Can either end of the connection start a new subflow (or only the initiator of the original subflow)? Currently, only the initiator of the original subflow starts a new subflow. Given the widespread deployment of NATs, it is often difficult for the server to reach the client. This is the main reason why the server currently does not start new subflows in our implementation. But, the initiator would accept a SYN+MP_JOIN if sent by another implementation.

c. What is the maximum number of subflows your implementation can support? Currently 32.

d. Any other comments or information?

3. Data transfer (DSS) [Section 3.3 RFC6824]

a. What is the status of your implementation? Fully working implementation of data transfer.

b. The "Data ACK" field can be 4 or 8 octets. Which one(s) have you implemented? We use 4 bytes for the DATA-ACK field.

c. The "Data sequence number" field can be 4 or 8 octets. Which one(s) have you implemented? We use 4 bytes for the data sequence number.

d. Does your implementation support the "DATA_FIN" operation to close an MPTCP connection? Yes.

e. Does your implementation support the "Checksum" field (which is negotiated in the MP_CAPABLE handshake)? Yes. This is configurable via a sysctl.

f. Any other comments or information? We support interoperability with implementations that do send 64-bit data sequence numbers and data acks. However, even if the peer sends 64-bit data sequence numbers, we will only reply with a 32-bit data-ack. We do not have heuristics to trigger the sending of DATA_ACKs. We simply send the DATA_ACK in each packet.

4. Address management (ADD_ADDR and REMOVE_ADDR) [Section 3.4 RFC6824]

a. What is the status of your implementation? We support ADD_ADDR/ REMOVE_ADDR messages.

b. Can your implementation do ADD_ADDRESS for addresses that appear *after* the connection has been established? Yes, as shown in: "Exploring Mobile/WiFi Handover with Multipath TCP", C. Paasch et. al, ACM SIGCOMM workshop on Cellular Networks (Cellnet’12), 2012.

c. Any other comments or information? We do not send out TCP keepalive-messages upon the reception of a REMOVE_ADDR-message.

5. Fast close (MP_FASTCLOSE) [Section 3.5 RFC6824]

a. What is the status of your implementation? We support the MP_FASTCLOSE implementation.

b. Any other comments or information?

3.4. Question 4: Fallback from MPTCP Question 4 asks about action when there is a problem with MPTCP, for example due to a middlebox mangling MPTCP’s signalling. The connection needs to fall back: if the problem is on the first subflow then MPTCP falls back to TCP, whilst if the problem is on an additional subflow then that subflow is closed with a TCP RST, as discussed in [Section 3.6 RFC6824].

1. If the MP_CAPABLE option is removed by a middlebox, does your implementation fall back to TCP? Yes.

2. If the MP_JOIN option does not get through on the SYNs, does your implementation close the additional subflow? Yes.

3. If the DSS option does not get through on the first data segment(s), does your implementation fall back? (either falling back to MPTCP (if the issue is on the first subflow) or closing the

additional subflow (if the issue is on an additional subflow)) Yes. On the initial subflow we do a seamless fallback, additional subflows will be closed by a RST.

4. Similarly, if the "DATA ACK" field does not correctly acknowledge the first data segment(s), does your implementation fall back? Yes. Same as above.

5. Does your implementation protect data with the "Checksum" field in the DSS option [Section 3.3 RFC6824]? If the checksum fails (because the subflow has been affected by a middlebox), does your implementation immediately close the affected subflow (with a TCP RST) with the MP_FAIL Option? If the checksum fails and there is a single subflow, does your implementation handle this as a special case, as described in [Section 3.6 RFC6824]? Yes, we support the DSS-checksum. If the checksum is wrong and there exist other subflows, we close the current subflow with an RST. If there is no other subflow, we send an ACK + MP_FAIL and do a fallback to infinite mapping. This fallback has successfully been tested with different type of NAT middleboxes, while using FTP.

6. Does your implementation fall back to TCP by using an "infinite mapping" [Section 3.3.1 RFC6824] (so that the subflow-level data is mapped to the connection-level data for the remainder of the connection)? Yes.

7. Did you find any corner cases where MPTCP’s fallback didn’t happen properly? No. We have developped a test-suite to test the middlebox-traversal of MPTCP, available at http://multipath-tcp.org/pmwiki.php/Users/AboutMeasures

8. Any other comments or information about fallback?

3.5. Question 5: Heuristics Question 5 gathers information about heuristics: aspects that are not required for protocol correctness but impact the performance. We would like to document best practice so that future implementers can learn from the experience of pioneers. The references contain some initial comments about each topic.

1. Receiver considerations [S3.3.4, RFC6824]: What receiver buffer have you used? Does this depend on the retransmission strategy? What advice should we give about the receiver? Linux includes an autuning algorithm for the TCP receiver buffer. This algorithm has been slightly modified for Multipath TCP. The receive-buffer does not depend on the retransmission strategy.

2. Sender considerations [S3.3.5, RFC6824]: How do you determine how

much data a sender is allowed to send and how big the sender buffer is? What advice should we give about the sender? The send-buffer is autotuned similarly as the receive-buffer (see above). We send as much data as possible, filling the congestion windows of each subflow. The sender deploys the "Opportunistic Retransmission" and "Penalization" algorithms from the paper: "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP", C. Raiciu et. al, NSDI 2012.

3. Reliability and retransmissions [S3.3.6, RFC6824]: What is your retransmission policy? (when do you retransmit on the original subflow vs on another subflow or subflows?) When do you decide that a subflow is underperforming and should be reset, and what do you then do? What advice should we give about this issue? Upon an RTO on subflow A, we reinject all the unacknowledged data of subflow A on another subflows. We do not currently have a mechanism to detect that a subflow is underperforming.

4. Port usage [S3.3.8.1, RFC6824]: Does your implementation use the same port number for additional subflows as for the first subflow? Have you used the ability to define a specific port in the Add Address option? What advice should we give about this issue? We always use the same port number as for the first subflow. Except, if the ADD_ADDRESS option that has been received contained a specific port. We do not have a means to configure the specific port in the ADD_ADDRESS option, but we support reception of the port.

5. Delayed subflow start [S3.3.8.2, RFC6824]: What factors does your implementation consider when deciding about opening additional subflows? What advice should we give about this issue? As soon as we are sure that the initial subflow is fully MPTCP-capable (reception of a DATA_ACK), we create a full mesh among all IP-addresses between the two hosts. We do not explicitly delay the creation of new subflows.

6. Failure handling [S3.3.8.3, RFC6824]: Whilst the protocol defines how to handle some unexpected signals, the behaviour after other unexpected signals is not defined. What advice should we give about this issue? We did not implement the caching mentioned in Section 3.8.3.

7. Use of TCP options: As discussed in [Appendix A, RFC6824], the TCP option space is limited, but a brief study found there was enough room to fit all the MPTCP options. However there are constraints on which MPTCP option(s) can be included in packets with other TCP options - do the suggestions in Appendix A need amending or expanding? We do not implement specific heuristics to reduce the TCP option-space usage. If timestamp is enabled we will only be able to

send two SACK-blocks, because the DATA_ACK consumes the remaining bytes.

8. What other heuristics should we give advice about? Any other comments or information?

3.6. Question 6: Security Question 6 asks about Security related matters [Section 5 RFC6824].

1. Does your implementation use the hash-based, HMAC-SHA1 security mechanism defined in [RFC6824]? Yes.

2. Does your implementation support any other handshake algorithms? We have in a separate branch, an implementation of draft-paasch-mptcp-lowoverhead and draft-paasch-mptcp-ssl.

3. It has been suggested that a Standards-track MPTCP needs a more secure mechanism. Do you have any views about how to achieve this? We believe that the solution described in draft-paasch-mptcp-ssl would be a good starting point since it leverages the security of the upper layer.

4. Any other comments or information?

3.7. Question 7: IANA Question 7 asks about IANA related matters.

1. Does your implementation follow the IANA-related definitions? [Section 8 RFC6824] defines: TCP Option Kind number (30); the sub-registry for "MPTCP Option Subtypes"; and the sub-registry for "MPTCP Handshake Algorithms" Yes.

2. Any other comments or information?

3.8. Question 8: Congestion control and subflow policy Question 8 asks about how you share traffic across multiple subflows.

1. How does your implementation share traffic over the available paths? For example: as a spare path on standby ('all-or- nothing'), as an 'overflow', etc? Does it have the ability to send /receive traffic across multiple subflows simultaneously? The implementation is able to send and receive traffic on all subflows simultaneously. Our scheduler first tries to send traffic on the subflow with the lowest RTT. As this subflow's congestion window is full, we pick the subflow with the next lower RTT.

2. Does your implementation support "handover" from one subflow to another when losing an interface? Yes, as described in: "Exploring Mobile/WiFi Handover with Multipath TCP", C. Paasch et. al, ACM

SIGCOMM workshop on Cellular Networks (Cellnet'12), 2012.

3. Does your implementation support the coupled congestion control defined in [RFC6356]? Yes.

4. Does your implementation support some other coupled congestion control (ie that balances traffic on multiple paths according to feedback)? We also support the OLIA congestion control (draft-khalili-mptcp-congestion-control-00).

5. The MP_JOIN (Starting a new subflow) Option includes the "B" bit, which allows the sender to indicate whether it wishes the new subflow to be used immediately or as a backup if other path(s) fail. The MP_PRIO Option is a request to change the "B" bit - either on the subflow on which it is sent, or (by setting the optional Address ID field) on other subflows. Does your implementation support the "B" bit and MP_PRIO mechanisms? Do you think they're useful, or have another suggestion? Yes, we support the "B"-bit of the MP_JOIN and the MP_PRIO option. It is configurable on a per-interface basis. Experiences with the "B"-bit can be found in our paper: "Exploring Mobile/WiFi Handover with Multipath TCP", C. Paasch et. al, ACM SIGCOMM workshop on Cellular Networks (Cellnet'12), 2012.

6. Any other comments or information or suggestions about the advice we should give about congestion control [S3.3.7 RFC6824] and subflow policy [S3.3.8 RFC6824]?

3.9. Question 9: API Question 9 gathers information about your API. [RFC6897] considers the MPTCP Application Interface.

1. With your implementation, can legacy applications use (the existing sockets API to use) MPTCP? How does the implementation decide whether to use MPTCP? Should the advice in [Section 4, RFC6897] be modified or expanded? Yes, a standard TCP socket API can be used. By default MPTCP is enabled on all connections.

2. The "basic MPTCP API" enables MPTCP-aware applications to interact with the MPTCP stack via five new socket options. For each one, have you implemented it? has it been useful? None of them are part of the current stable release MPTCP v0.86. http://multipath-tcp.org/pmwiki.php?n=Main.Release86 a. TCP_MULTIPATH_ENABLE? b. TCP_MULTIPATH_ADD? c. TCP_MULTIPATH_REMOVE? d. TCP_MULTIPATH_SUBFLOWS? e. TCP_MULTIPATH_CONNID?

3. Have you implemented any aspects of an "advanced MPTCP API"? ([Appendix A, RFC6897] hints at what it might include.) No.

4. Any other comments or information?

3.10. Question 10: Deployments, use cases and operational experiences Question 10 takes the opportunity of this survey to gather some limited information about operational experiences and deployments. Any very brief information would be appreciated, for example: 1. What deployment scenarios are you most interested in? 2. Is your deployment on "the Internet" or in a controlled environment? 3. Is your deployment on end hosts or with a MPTCP-enabled proxy (at one or both ends?)? 4. What do you see as the most important benefits of MPTCP in your scenario(s)? 5. How extensively have you deployed and experimented with MPTCP so far?

Our implementation is open-source and has been discussed for various types of tests/deployments based on the messages received on the mptcp-dev mailing list. We currently use Multipath TCP on www.multipath-tcp.org and also on internal ssh servers at UCLouvain. 6. MPTCP's design seeks to maximise the chances that the signalling works through middleboxes. Did you find cases where middleboxes blocked MPTCP signalling? We have implemented a test suite based on a slightly modified version of the Multipath TCP implementation that allows to check the interoperability between Multipath TCP and middleboxes. We have used it over Internet paths and identified some potential problems. However, the best approach to test these interactions would be to control the middlebox and analyse its logs during the Multipath TCP test. The test suite can be retrieved from http://multipath-tcp.org/pmwiki.php/Users/AboutMeasures

7. MPTCP's design seeks to ensure that, if there is a problem with MPTCP signalling, then the connection either falls back to TCP or removes the problematic subflow. Did you find any corner cases where this didn't happen properly? See above.

8. Have you encountered any issues or drawbacks with MPTCP?

9. Any other comments or information?

3.11. Question 11: Improvements to RFC6824

1. Are there any areas where [RFC6824] could be improved, either in technical content or clarity? 2. Any other issues you want to raise?

8. Full survey response for Implementation 2

Question 1: Your details

1.1 Swinburne University of Technology, Hawthorn, Victoria, Australia

1.2 Lawrence Stewart, Nigel Williams

1.3 No

Question 2: Preliminary information about your implementation
----

2.1 FreeBSD-10

2.2 Currently IPv4 only (IPv6 support will eventually be added)

2.3 Publicly available (http://caia.swin.edu.au/urp/newtcp/mptcp/). The code is released under the BSD license. 2.3

2.5 Independent

2.6 Yes, some limited testing to establish interoperability.

2.7 Yes, with some additional work this should be possible (if not then IETF-88). Q

uestion 3: Support for MPTCP’s Signaling Functionality

3.1 a) MP_CAPABLE Implemented

b) Do not currently honour checksum flag (to be implemented)

3.2 a) MP_JOIN Implemented

b) Either end can initiate a MP_JOIN

c) 8 (controlled via sysctl)

d) Currently do not include HMAC verification during handshake, but this will be enabled in the next patch (several weeks from time of submission)

3.3 a) DSS Implemented

b) 4 (default) and 8

c) 4 (default) and 8

d) Yes, however the connection tear-down exchange is not fully implemented - the connection shuts down but the DFIN may not be correctly acknowledged.

e) No. This will be supported eventually (time-frame unknown)

3.4 a) ADD_ADDR implemented, REMOVE_ADDR not implemented (to be done, timeframe unknown)

b) No. Functionality to be added

3.5 MP_FASTCLOSE not implemented. Plan to implement eventually

Question 4: Fallback from MPTCP

4.1 Yes

4.2 The subflow PCBs remain allocated, however the subflow is not used to send data.

4.3 No, tbd

4.4 No, tbd

4.5 No, checksumming not implemented

4.6 Yes

4.8 Fallback hasn’t really been put through any structured tests yet

Question 5: Heuristics

5.1 We use "TCP_MAXWIN << tp->rcv_scale". This is temporary and we will use a call into the "multipath" control layer to determine this value in future releases (we need to investigate a suitable way of calculating this).

5.2 cwnd determines the amount of data to send (given that rcv window is always very large). Sendbuffer is scaled based on occupancy.

5.3 We currently don’t have Data-level retransmits enabled. However our policy is to retransmit on the next subflow that requests data to send that is suitable. There is no intelligence in the packet schedular currently,

5.4 The same port numbers are re-used for additional subflows.

Question 6: Security

6.1 Yes

6.2 No

Question 7: IANA

7.1 Yes

Question 8: Congestion Control and subflow policy

8.1 A simple algorithm is used to divide the send buffer between subflows, so that traffic is spread across the subflows.

8.3 No. (to be added)

8.4 No

8.5 No

Question 9: API

9.1 Legacy applications are able to use MPTCP. MPTCP is set globally via a sysctl variable.

9.2 No

9.3 No

Question 10: API

10.1 Some current project work is based on MPTCPs use in vehicle to infrastructure (V2I) connectivity (to provide a persistent connection using 3G and roadside wifi). Other interests are in multi-homed "home-user" environments, high throughput data transfers.... We are also interested in experimenting with congestion control across paths with different path-cost metrics.

10.2 So far only within controlled testbeds

10.3 End hosts

10.4 Depending on the scenario, connection persistence, throughput...

10.5 Still an alpha-quality build, so limited testing so far.

9. Full survey response for Implementation 3

Survey 3.1. Question 1: Your details Question 1 gathers some information about the team that has implemented MPTCP.

1. Your institution: anonymized.

2. Name(s) of people in your implementation and test teams: There were several folks involved in the implementation and testing.

3. Do you want your answers to Question 1.1 and 1.2 above to be anonymised? Yes.

3.2. Question 2: Preliminary information about your implementation Question 2 gathers some preliminary information.

1. What OS is your implementation for? (or is it application layer?) anonymized (commercial OS)

2. Do you support IPv4 or IPv6 addresses or both? Both.

3. Is it publicly available (or will it be?) (for free or a fee?) No.

4. Overall, what are you implementation and testing plans? (details can be given against individual items later) We plan to use it in a mobile environment.

5. Is it an independent implementation? Or does it build on another MPTCP implementation -which one? It is an independent implementation.

6. Have you already done some interop tests, for example with UCLouvain’s "reference" Linux implementation? Most MPTCP option formats were tested with the reference Linux implementation.

7. Would you be prepared to take part in an interop event, for example adjacent to IETF-87 in Berlin? Unsure at this point.

3.3. Question 3: Support for MPTCP’s Signalling Functionality Question 3 asks about support for the various signalling messages that the MPTCP protocol defines. *** For each message, please give a little information about the status of your implementation: for example, you may have implemented it and fully tested it; the implementation may be in progress; you have not yet implemented it but plan to soon (timescale?); you may you have no intention to implement it (why?); etc.

1. Connection initiation (MP_CAPABLE) [Section 3.1 RFC6824] a. What is the status of your implementation? Fully implemented and tested against the reference Linux implementation.

b. Any other comments or information?

2. Starting a new subflow (MP_JOIN) [Section 3.2 RFC6824] a. What is the status of your implementation? Fully implemented and tested against the reference Linux implementation.

b. Can either end of the connection start a new subflow (or only the initiator of the original subflow)? Only the initiator of the original sub flow can start other sub flows.

c. What is the maximum number of subflows your implementation can support? There is no hard limit.

d. Any other comments or information?

3. Data transfer (DSS) [Section 3.3 RFC6824] a. What is the status of your implementation? Fully implemented and tested.

b. The "Data ACK" field can be 4 or 8 octets. Which one(s) have you implemented? Both have been implemented but the use of the 4-byte field is the default. When an 8 byte DSS is received, an 8 byte Data ACK is sent in response.

c. The "Data sequence number" field can be 4 or 8 octets. Which one(s) have you implemented? Both have been implemented but the use of the 4-byte field is the default. When a wraparound of the lower 32-bit part of the DSS is detected, the full 8 byte DSS is sent.

d. Does your implementation support the "DATA_FIN" operation to close an MPTCP connection? Yes. There are cases however where the sub flows are closed (TCP FIN’d) but the DATA_FIN is not sent - in this case the MPTCP connection must be closed through a garbage collector after some idle time.

e. Does your implementation support the "Checksum" field (which is negotiated in the MP_CAPABLE handshake)? Yes.

f. Any other comments or information?

4. Address management (ADD_ADDR and REMOVE_ADDR) a. What is the status of your implementation? It does not support sending ADD_ADDR

or processing ADD_ADDR as it is considered a security risk. Also, we only have a client side implementation at the moment which always initiates the sub flows. The remote end does not send ADD_ADDR in our configuration. The client can send REMOVE_ADDR however when one of the established sub flow's source address goes away. The client ignores incoming REMOVE_ADDR options also.

b. Can your implementation do ADD_ADDRESS for addresses that appear *after* the connection has been established? No. c. Any other comments or information?

5. Fast close (MP_FASTCLOSE) [Section 3.5 RFC6824] a. What is the status of your implementation? It is supported. Though Retransmission of Fast close is not supported yet.

b. Any other comments or information?

3.4. Question 4: Fallback from MPTCP Question 4 asks about action when there is a problem with MPTCP, for example due to a middlebox mangling MPTCP's signalling. The connection needs to fall back: if the problem is on the first subflow then MPTCP falls back to TCP, whilst if the problem is on an additional subflow then that subflow is closed with a TCP RST, as discussed in [Section 3.6 RFC6824].

1. If the MP_CAPABLE option is removed by a middlebox, does your implementation fall back to TCP? Yes.

2. If the MP_JOIN option does not get through on the SYNs, does your implementation close the additional subflow? Yes.

3. If the DSS option does not get through on the first data segment(s), does your implementation fall back? (either falling back to MPTCP (if the issue is on the first subflow) or closing the additional subflow (if the issue is on an additional subflow)) Yes it falls back to TCP when there's one sub flow. When there are multiple sub flows, since MPTCP is used in active/backup mode, it is assumed that the sub flow transferring data is most likely to be more usable than any other established sub flow. So the sub flow on which fallback occurred is kept alive and other sub flows are closed. Fallback though is not guaranteed to occur safely when there are more than one sub flows because the infinite mapping option may be stripped like other DSS options and the MP_FAIL option if used in scenarios other than for reporting checksum failure can also be stripped.

4. Similarly, if the "DATA ACK" field does not correctly acknowledge the first data segment(s), does your implementation fall back? No. Current implementation just ignores the unexpected data ack.

5. Does your implementation protect data with the "Checksum" field in the DSS option [Section 3.3 RFC6824]? If the checksum fails (because the subflow has been affected by a middlebox), does your implementation immediately close the affected subflow (with a TCP RST) with the MP_FAIL Option? If the checksum fails and there is a single subflow, does your implementation handle this as a special case, as described in [Section 3.6 RFC6824]? Yes.

6. Does your implementation fall back to TCP by using an "infinite mapping" [Section 3.3.1 RFC6824] (so that the subflow-level data is mapped to the connection-level data for the remainder of the connection)? Yes.

7. Did you find any corner cases where MPTCP's fallback didn't happen properly? If the very first sub flow does not send any data and is disconnected right away, then the current implementation allows a join to occur with the addition of another sub flow which then becomes a fully mp capable sub flow. Thus we allow break before make by letting additional sub flows to be joined if the very first one disconnected even without sending any data. This is a very corner case but an instance where we do not follow the rules of fallback (allow second sub flow even when first sub flow did not send/receive data/data acks).

8. Any other comments or information about fallback? Fallback after connection establishment and after a few data packets were transferred with MPTCP options is complicated. The spec does not clearly cover the cases of options being stripped by middle boxes. It goes into good detail about what to do when the DSS checksum fails, but not when DSS checksum is not in use and the MPTCP options are stripped. Both sender/receiver behaviors could be outlined with more detail.

3.5. Question 5: Heuristics Question 5 gathers information about heuristics: aspects that are not required for protocol correctness but impact the performance. We would like to document best practice so that future implementers can learn from the experience of pioneers. The references contain some initial comments about each topic.

1. Receiver considerations [S3.3.4, RFC6824]: What receiver buffer have you used? Does this depend on the retransmission strategy? What advice should we give about the receiver? We are just using MPTCP in active/backup mode. This mode is simpler wrt receive buffer utilization. The receive buffer sizes at the MPTCP and sub flow level is the same. Automatic buffer tuning is turned off when MPTCP is in use.

2. Sender considerations [S3.3.5, RFC6824]: How do you determine how much data a sender is allowed to send and how big the sender buffer is? What advice should we give about the sender? Automatic buffer tuning is turned off when MPTCP is in use.

3. Reliability and retransmissions [S3.3.6, RFC6824]: What is your retransmission policy? (when do you retransmit on the original subflow vs on another subflow or subflows?) When do you decide that a subflow is underperforming and should be reset, and what do you then do? What advice should we give about this issue? Retransmissions at MPTCP level do not occur on the same sub flow except when MP_FAIL option is received. A sub flow is said to be underperforming when its network connectivity goes away.

4. Port usage [S3.3.8.1, RFC6824]: Does your implementation use the same port number for additional subflows as for the first subflow? Have you used the ability to define a specific port in the Add Address option? What advice should we give about this issue? The destination port is the same. The local port changes for additional sub flows so on the wire it is like two tcp connections to the same remote destination. We have not used Add Address option at all.

5. Delayed subflow start [S3.3.8.2, RFC6824]: What factors does your implementation consider when deciding about opening additional subflows? What advice should we give about this issue? The client implementation is aware of network interfaces coming up or going down and establishes new sub flows or removes existing sub flows accordingly.

6. Failure handling [S3.3.8.3, RFC6824]: Whilst the protocol defines how to handle some unexpected signals, the behaviour after other unexpected signals is not defined. What advice should we give about this issue? Fallback, post establishment is probably a case that needs to be more clearly defined.

7. Use of TCP options: As discussed in [Appendix A, RFC6824], the TCP option space is limited, but a brief study found there was enough room to fit all the MPTCP options. However there are constraints on which MPTCP option(s) can be included in packets with other TCP options - do the suggestions in Appendix A need amending or expanding? Looks good already.

8. What other heuristics should we give advice about? Any other comments or information?

3.6. Question 6: Security Question 6 asks about Security related matters [Section 5 RFC6824].

1. Does your implementation use the hash-based, HMACSHA1 security mechanism defined in [RFC6824]? Yes.

2. Does your implementation support any other handshake algorithms? No.

3. It has been suggested that a Standards-track MPTCP needs a more secure mechanism. Do you have any views about how to achieve this? No. But the mechanism could be tied with SSL because SSL is used wherever security is deemed important.

4. Any other comments or information?

3.7. Question 7: IANA Question 7 asks about IANA related matters.

1. Does your implementation follow the IANA-related definitions? [Section 8 RFC6824] defines: TCP Option Kind number (30); the sub-registry for "MPTCP Option Subtypes"; and the Page 12 of 17 Survey 6/22/13, 5:55 PM sub-registry for "MPTCP Handshake Algorithms" Yes.

2. Any other comments or information? No.

3.8. Question 8: Congestion control and subflow policy Question 8 asks about how you share traffic across multiple subflows.

1. How does your implementation share traffic over the available paths? For example: as a spare path on standby ('all-ornothing'), as an 'overflow', etc? Does it have the ability to send /receive traffic across multiple subflows simultaneously? It uses active/ backup where one sub flow is preferred or has higher priority over other sub flows. When the preferred sub flow fails or begins to experience retransmission timeouts, the other sub flows are used.

2. Does your implementation support "handover" from one subflow to another when losing an interface? Yes.

3. Does your implementation support the coupled congestion control defined in [RFC6356]? No.

4. Does your implementation support some other coupled congestion control (ie that balances traffic on multiple paths according to feedback)? No.

5. The MP_JOIN (Starting a new subflow) Option includes the "B" bit, which allows the sender to indicate whether it wishes the new subflow to be used immediately or as a backup if other path(s) fail. The MP_PRIO Option is a request to change the "B" bit - either on the subflow on which it is sent, or (by setting the optional Address ID

field) on other subflows. Does your implementation support the "B" bit and MP_PRIO mechanisms? Do you think they're useful, or have another suggestion? Yes the implementation uses the B bit and the MP_PRIO option. They are very useful for the active/backup mode of operation.

6. Any other comments or information or suggestions about the advice we should give about congestion control [S3.3.7 RFC6824] and subflow policy [S3.3.8 RFC6824]?

3.9. Question 9: API Question 9 gathers information about your API. [RFC6897] considers the MPTCP Application Interface.

1. With your implementation, can legacy applications use (the existing sockets API to use) MPTCP? How does the implementation decide whether to use MPTCP? Should the advice in [Section 4, RFC6897] be modified or expanded? The implementation does not support MPTCP with existing sockets API. MPTCP is exposed through a private SPI today. If MPTCP becomes prolific over the next few years, MPTCP use shall be expanded.

2. The "basic MPTCP API" enables MPTCP-aware applications to interact with the MPTCP stack via five new socket options. For each one, have you implemented it? has it been useful? a. TCP_MULTIPATH_ENABLE? b. TCP_MULTIPATH_ADD? c. TCP_MULTIPATH_REMOVE? d. TCP_MULTIPATH_SUBFLOWS? e. TCP_MULTIPATH_CONNID? This mode of API is not used. Proprietary methods are used for achieving these basic operations.

3. Have you implemented any aspects of an "advanced MPTCP API"? ([Appendix A, RFC6897] hints at what it might include.) No.

4. Any other comments or information?

3.10. Question 10: Deployments, use cases and operational experiences Question 10 takes the opportunity of this survey to gather some limited information about operational experiences and deployments. Any very brief information would be appreciated, for example:

1. What deployment scenarios are you most interested in? MPTCP in mobile environments is very powerful when used in the active/backup mode. Since the network interfaces available on mobile devices have different cost characteristics as well as different bring up and power usage characteristics, it is not useful to share load across all available network interfaces - at least not currently. Providing session continuity across changing network environments is the key deployment scenario.

2. Is your deployment on "the Internet" or in a controlled environment? The deployment is on the Internet.

3. Is your deployment on end hosts or with a MPTCPenabled proxy (at one or both ends?)? The deployment supports MPTCP on both ends.

4. What do you see as the most important benefits of MPTCP in your scenario(s)? Described in point 1 of this section.

5. How extensively have you deployed and experimented with MPTCP so far? Deployment is still in early stages. We have been experimenting with MPTCP for about a year.

6. MPTCP’s design seeks to maximise the chances that the signalling works through middleboxes. Did you find cases where middleboxes blocked MPTCP signalling? Corporate firewalls block MPTCP signaling by default. IETF is one venue where Cisco, and other firewall vendors can be asked to change their defaults to allow MPTCP signals.

7. MPTCP’s design seeks to ensure that, if there is a problem with MPTCP signalling, then the connection either falls back to TCP or removes the problematic subflow. Did you find any corner cases where this didn’t happen properly? This has been covered a bit in the Fallback section. When using two sub flows in active/backup mode, there is a possibility that a backup sub flow that never sent data starts being used for retransmitting data that is not going through on the active path. While it is preferable to keep the initial sub flow that successfully sent MPTCP options and drop the backup path, the initial sub flow may be the failing one, and we may want to move to the backup path. But the backup path can be retransmitting data that did not get sent successfully on the active path and if there is a middle box in the backup sub flow’s path stripping options, then we have a case where the MPTCP session may not be recoverable as it may not be evident from what point in the MPTCP sequence space, data was being sent. The spec does talk of retaining the initial sub flow and closing the failed flow. So perhaps doing the reverse is not recommended, however, it would certainly be advantageous to support MPTCP better in such a failing environment. Also, in parallel working with firewall vendors to allow MPTCP options always to not have to over-engineer these cases.

8. Have you encountered any issues or drawbacks with MPTCP?

9. Any other comments or information?

3.11. Question 11: Improvements to RFC6824 1. Are there any areas where [RFC6824] could be improved, either in technical content or clarity? Discussed in the fallback section. Other areas around

MPTCP performance such as support for sub flow level automatic buffer scaling, varying QoS support, varying window scaling support on each sub flow may be worth discussing further, although they are outside the scope of the current spec.

2. Any other issues you want to raise? Some additional work on option signaling that we will bring up in future discussions.

10. Full survey response for Implementation 4

1. Your institution: Citrix Systems, Inc.

2. Name(s) of people in your implementation and test teams: NA

3. Do you want your answers to Question 1.1 and 1.2 above to be anonymised? No

3.2. Question 2: Preliminary information about your implementation Question 2 gathers some preliminary information.

1. What OS is your implementation for? (or is it application layer?) NetScaler Firmware

2. Do you support IPv4 or IPv6 addresses or both? Both

3. Is it publicly available (or will it be?) (for free or a fee?) It is available for purchase

4. Overall, what are you implementation and testing plans? (details can be given against individual items later)

5. Is it an independent implementation? Or does it build on another MPTCP implementation -which one? It is an independent implementation

6. Have you already done some interop tests, for example with UCLouvain’s "reference" Linux implementation? Yes, our implementation is extensively tested with Linux reference implementation

7. Would you be prepared to take part in an interop event, for example adjacent to IETF-87 in Berlin?

3.3. Question 3: Support for MPTCP’s Signalling Functionality Question 3 asks about support for the various signalling messages that the MPTCP protocol defines. *** For each message, please give a little information about the status of your implementation: for example, you may have implemented it and fully tested it; the implementation may be in progress; you have not yet implemented it but plan to soon (timescale?); you may you have no intention to implement it (why?); etc.

1. Connection initiation (MP_CAPABLE) [Section 3.1 RFC6824] a. What is the status of your implementation? Fully implemented and tested

b. Any other comments or information? One security concern here is that the keys are exchanged in plain text which is prone to attacks and also the key generation mechanism is highly computational intensive

2. Starting a new subflow (MP_JOIN) [Section 3.2 RFC6824] a. What is the status of your implementation? Fully implemented and tested

b. Can either end of the connection start a new subflow (or only the initiator of the original subflow)? Only the initiator of the original subflow can initiate additional subflows.

c. What is the maximum number of subflows your implementation can support? we support maximum 6 subflows.

d. Any other comments or information?

3. Data transfer (DSS) [Section 3.3 RFC6824] a. What is the status of your implementation? Fully implemented and tested

b. The "Data ACK" field can be 4 or 8 octets. Which one(s) have you implemented? Our implementation supports both 4 or 8 Octets Data Ack in both the directions

c. The "Data sequence number" field can be 4 or 8 octets. Which one(s) have you implemented? Our implementation supports both 4 or 8 Octets DSN in both the directions

d. Does your implementation support the "DATA_FIN" operation to close an MPTCP connection? YES

e. Does your implementation support the "Checksum" field (which is negotiated in the MP_CAPABLE handshake)? YES

f. Any other comments or information?

4. Address management (ADD_ADDR and REMOVE_ADDR) [Section 3.4 RFC6824]

a. What is the status of your implementation? REMOVE_ADDR is implemented and tested

b. Can your implementation do ADD_ADDRESS for addresses that appear *after* the connection has been established? NO

c. Any other comments or information? ADD_ADDRESS may not be much useful in the real environment situation given that most of the clients are behind the NATing devices.

5. Fast close (MP_FASTCLOSE) [Section 3.5 RFC6824] a. What is the status of your implementation? Implemented and tested b. Any other comments or information?

3.4. Question 4: Fallback from MPTCP Question 4 asks about action when there is a problem with MPTCP, for example due to a middlebox mangling MPTCP’s signalling. The connection needs to fall back: if the problem is on the first subflow then MPTCP falls back to TCP, whilst if the problem is on an additional subflow then that subflow is closed with a TCP RST, as discussed in [Section 3.6 RFC6824].

1. If the MP_CAPABLE option is removed by a middlebox, does your implementation fall back to TCP? YES

2. If the MP_JOIN option does not get through on the SYNs, does your implementation close the additional subflow? YES

3. If the DSS option does not get through on the first data segment(s), does your implementation fall back? (either falling back to MPTCP (if the issue is on the first subflow) or closing the additional subflow (if the issue is on an additional subflow)) YES

4. Similarly, if the "DATA ACK" field does not correctly acknowledge the first data segment(s), does your implementation fall back? If the sender receives pure ack for its first DSS packet then it fallsback to regular TCP.

5. Does your implementation protect data with the "Checksum" field in the DSS option [Section 3.3 RFC6824]? If the checksum fails (because the subflow has been affected by a middlebox), does your implementation immediately close the affected subflow (with a TCP RST) with the MP_FAIL Option? If the checksum fails and there is a single subflow, does your implementation handle this as a special case, as described in [Section 3.6 RFC6824]? Yes, our implementation supports DSS checksum and will close the subflow with RST if the checksum validation fails and there are more than one subflows and sends MP_FAIL if there is a single subflow expecting infinite map from the peer.

6. Does your implementation fall back to TCP by using an "infinite mapping" [Section 3.3.1 RFC6824] (so that the subflow-level data is

mapped to the connection-level data for the remainder of the connection)? YES.

7. Did you find any corner cases where MPTCP's fallback didn't happen properly? We have found few cases where the draft is not clear about the recommended action and fallback strategy, like: 1. what is the expected behavior when pure ack or data packet without dss is received in middle of transaction? How the hosts should fallback in this case? This can happen if the routing changes and the new path drops mptcp options. In this case MP_FAIL/infinite map exchange may not be possible and so could not decide whether both parties are in sync to fallback to tcp. 2. whether infinite map is unidirectional or bidirectional? If one host is sending infinite map to peer, does the peer also needs to send infinite map to the host? Exchanging infinite map and falling back to TCP from both ends is easy from implementation point of view. 8. Any other comments or information about fallback?

3.5. Question 5: Heuristics Question 5 gathers information about heuristics: aspects that are not required for protocol correctness but impact the performance. We would like to document best practice so that future implementers can learn from the experience of pioneers. The references contain some initial comments about each topic.

1. Receiver considerations [S3.3.4, RFC6824]: What receiver buffer have you used? Does this depend on the retransmission strategy? What advice should we give about the receiver? Our implementation uses varying buffer size based on the services and application type.

2. Sender considerations [S3.3.5, RFC6824]: How do you determine how much data a sender is allowed to send and how big the sender buffer is? What advice should we give about the sender? The send side flow control is handled at mptcp level and is independent to subflows. The mptcp level flow control is (almost) same as the regular TCP flow control.

3. Reliability and retransmissions [S3.3.6, RFC6824]: What is your retransmission policy? (when do you retransmit on the original subflow vs on another subflow or subflows?) When do you decide that a subflow is underperforming and should be reset, and what do you then do? What advice should we give about this issue? The retransmission is done by the subflows as long as the subflow is alive and is not removed by the REM_ADDR/RST/.. . If 3 RTO happens on the subflow doing retransmission and multiple subflows are available then the mptcp starts retransmission from additional subflow. The original subflow continues retransmission for 7RTO and will be closed after that with RST.

4. Port usage [S3.3.8.1, RFC6824]: Does your implementation use the same port number for additional subflows as for the first subflow? Have you used the ability to define a specific port in the Add Address option? What advice should we give about this issue? Our current implementation doesnot support ADD_ADDR and subflow initiation.

5. Delayed subflow start [S3.3.8.2, RFC6824]: What factors does your implementation consider when deciding about opening additional subflows? What advice should we give about this issue? NA

6. Failure handling [S3.3.8.3, RFC6824]: Whilst the protocol defines how to handle some unexpected signals, the behaviour after other unexpected signals is not defined. What advice should we give about this issue? RFC should clearly define failure case handling otherwise it creates interoperability problems among various implementations. Our strategy in most of the unexpected failuire case is to send MP_FAIL RST with expected DSN if there are multiple subflows and MP_FAIL if there is a single subflow expecting infinite map from the peer.

7. Use of TCP options: As discussed in [Appendix A, RFC6824], the TCP option space is limited, but a brief study found there was enough room to fit all the MPTCP options. However there are constraints on which MPTCP option(s) can be included in packets with other TCP options - do the suggestions in Appendix A need amending or expanding? Looks fine now. Atleast timestamp can be included with every dss packet (28bytes for dss and 12bytes for Timestamp), but if there are any other options which needs to be included in data packets then the implementation has to choose which one to include among them.

8. What other heuristics should we give advice about? Any other comments or information?

3.6. Question 6: Security Question 6 asks about Security related matters [Section 5 RFC6824].

1. Does your implementation use the hash-based, HMAC-SHA1 security mechanism defined in [RFC6824]? YES.

2. Does your implementation support any other handshake algorithms? NO.

3. It has been suggested that a Standards-track MPTCP needs a more secure mechanism. Do you have any views about how to achieve this? Yes we also feel more secure and light weight mechanism is required.

4. Any other comments or information?

3.7. Question 7: IANA Question 7 asks about IANA related matters.

1. Does your implementation follow the IANA-related definitions? [Section 8 RFC6824] defines: TCP Option Kind number (30); the sub-registry for "MPTCP Option Subtypes"; and the sub-registry for "MPTCP Handshake Algorithms" YES. 2. Any other comments or information?

3.8. Question 8: Congestion control and subflow policy Question 8 asks about how you share traffic across multiple subflows.

1. How does your implementation share traffic over the available paths? For example: as a spare path on standby ('all-or- nothing'), as an 'overflow', etc? Does it have the ability to send /receive traffic across multiple subflows simultaneously? We give preference to the path that client is currently using to send data/ack and also has policy based on primary/backup setup. We accept data from multiple subflows simultaneously but don't send it simultaneously out.

2. Does your implementation support "handover" from one subflow to another when losing an interface? YES.

3. Does your implementation support the coupled congestion control defined in [RFC6356]? NO.

4. Does your implementation support some other coupled congestion control (ie that balances traffic on multiple paths according to feedback)? NO.

5. The MP_JOIN (Starting a new subflow) Option includes the "B" bit, which allows the sender to indicate whether it wishes the new subflow to be used immediately or as a backup if other path(s) fail. The MP_PRIO Option is a request to change the "B" bit - either on the subflow on which it is sent, or (by setting the optional Address ID field) on other subflows. Does your implementation support the "B" bit and MP_PRIO mechanisms? Do you think they're useful, or have another suggestion? YES, our implementation supports both 'B' flag and MP_PRIO options, they are much useful to change the priority of the subflows and to decide which subflow to use for data transfer.

6. Any other comments or information or suggestions about the advice we should give about congestion control [S3.3.7 RFC6824] and subflow policy [S3.3.8 RFC6824]?

3.9. Question 9: API Question 9 gathers information about your API. [RFC6897] considers the MPTCP Application Interface.

1. With your implementation, can legacy applications use (the existing sockets API to use) MPTCP? How does the implementation decide whether to use MPTCP? Should the advice in [Section 4, RFC6897] be modified or expanded? NA.

2. The "basic MPTCP API" enables MPTCP-aware applications to interact with the MPTCP stack via five new socket options. For each one, have you implemented it? has it been useful? a. TCP_MULTIPATH_ENABLE? b. TCP_MULTIPATH_ADD? c. TCP_MULTIPATH_REMOVE? d. TCP_MULTIPATH_SUBFLOWS? e. TCP_MULTIPATH_CONNID? NA.

3. Have you implemented any aspects of an "advanced MPTCP API"? ([Appendix A, RFC6897] hints at what it might include.) NA. 4. Any other comments or information?

3.10. Question 10: Deployments, use cases and operational experiences Question 10 takes the opportunity of this survey to gather some limited information about operational experiences and deployments. Any very brief information would be appreciated, for example:

1. What deployment scenarios are you most interested in? MPTCP Proxy deployment where the mptcp connections from the clients are terminated and the tcp connection is established on the other side.

2. Is your deployment on "the Internet" or in a controlled environment? Targeted for the Internet deployment.

3. Is your deployment on end hosts or with a MPTCP-enabled proxy (at one or both ends?)? Proxy.

4. What do you see as the most important benefits of MPTCP in your scenario(s)? Reliability and fault tolerance.

5. How extensively have you deployed and experimented with MPTCP so far?

6. MPTCP’s design seeks to maximise the chances that the signalling works through middleboxes. Did you find cases where middleboxes blocked MPTCP signalling? Yes some firewalls seem dropping MPTCP options.

7. MPTCP’s design seeks to ensure that, if there is a problem with MPTCP signalling, then the connection either falls back to TCP or removes the problematic subflow. Did you find any corner cases where this didn’t happen properly? Few cases listed above.

8. Have you encountered any issues or drawbacks with MPTCP? 9. Any other comments or information?

3.11. Question 11: Improvements to RFC6824

1. Are there any areas where [RFC6824] could be improved, either in technical content or clarity? More clarity required in fallback cases.

2. Any other issues you want to raise?

11. Normative References

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

Author's Address

Philip Eardley
BT