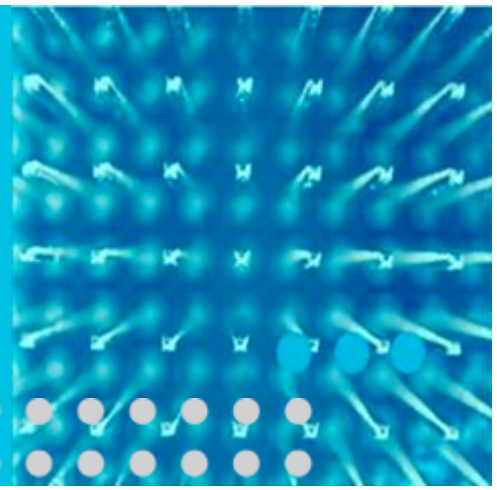
MPTCP Application Considerations

draft-scharf-mptcp-api-00.txt



Michael Scharf <michael.scharf@alcatel-lucent.com>
Alan Ford <alan.ford@roke.co.uk>

November 9, 2009

Scope

- From MPTCP Charter:
  - e. **An extended API describing how applications** can exploit additional features of multipath transport. While MPTCP needs to be usable without any application changes, this API should be an optional extension that provides access to multipath information and enables control over the usage of multipath.

    ➔ **Spec of an API, e. g., sockets interface extension**
  - f. An **application considerations document**, presenting the impacts that MPTCP may have on applications, such as performance changes. It should also discuss issues it create for applications, such as its effects on the usage of IPsec.

    ➔ **Tutorial-style doc for application developers**
- draft-scharf-mptcp-api-00 addresses currently both aspects
  - Closely related, since they both deal with the interaction of MPTCP and applications
  - Application developers probably have to know both of the implications and the API
  - Draft could alternatively be split into two separate documents

Impact of MPTCP on Applications: Performance Improvement

- General statement: Should never be worse than legacy, single-path TCP
- Throughput
  - Higher throughput due to resource pooling
  - Potentially exceeding the capacity of a single interface
  - Small overhead due to TCP options
- Delay
  - Delay jitter of the packet arrivals could be larger
  - Application-layer RTT estimation determines average RTT only
- Resilience
  - Communication can continue even if a subflow fails
  - Automatically handled by MPTCP, but an API may offer some control

Impact of MPTCP on Applications: Potential Problems

- Impact of middleboxes
  - Potential problems of using new TCP options
  - Fallback to regular TCP can result in additional handshake delays
- Outdated implicit assumptions
  - No one-to-one mapping of socket interface to flow through the network
  - Applications that want to use a single path can disable MPTCP
- Security implications
  - TBD
  - Could include a brief summary of the analysis in draft-bagnulo-mptcp-threat-00

Implications of MPTCP on Existing Interfaces

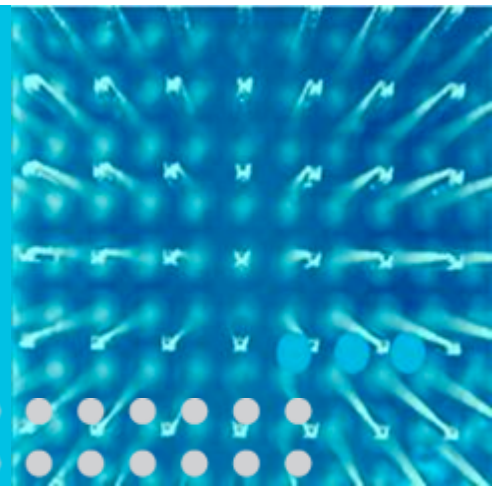
- Disabling the Nagle algorithm (TCP_NODELAY option)
  - Can be used with MTCP in the same way and affects then all subflows
  - Open question: Activation of a different path scheduler algorithm?
- Send and receive buffers (SO_SNDBUF/SO_RCVBUF options)
- Usage of a specific address by app (binding or SO_BINDTODEVICE option)
  - MPTCP respects the application's choice
  - TBD: Preferences could be expressed by the extended sockets API
- Impact on existing API calls
  - TBD: What data should be returned from getpeername() or getsockname()?
  - Idea: Always return first IP address pair

Open Issues

- Security
  - Effects on the usage of IPsec
  - Interaction with TLS
- Interactions with other API extensions (SHIM6, HIP, etc.)
- Impact on other interfaces (e. g., routing table)
- Anything we else have overlooked so far?

MPTCP Extended API

draft-scharf-mptcp-api-00.txt

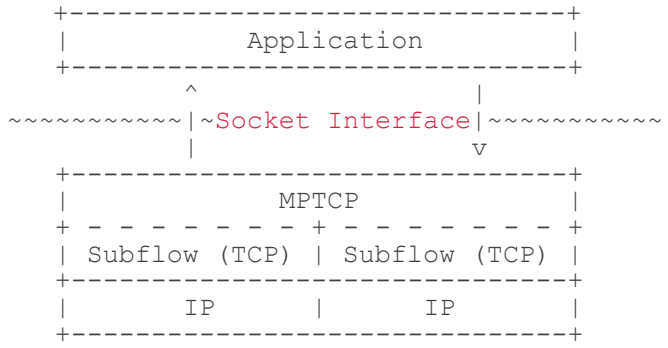


Michael Scharf, Alan Ford

November 9, 2009

MPTCP Extended API

- MPTCP is designed to be totally backward compatible
- But interface extensions still make sense



- Objectives of the extended API
  - Provide access to multipath information
    Examples: Get number of currently used subflows, ...
  - Enable control of some aspects of the MPTCP implementation's behaviour
    Examples: Turn on/off MPTCP, limit the maximum number of subflows, ...

Requirements and Key Design Aspects

- Question: *Is there a need for different application profiles?*
  Examples:
  - Default: Bulk data transport 

Pooling
  - Latency-sensitive transport (with overflow) 

Overflow
  - Latency-sensitive transport (hot-standby) 

Hot standby

- Question: *Features and expressiveness of the API?*

- Turn on/off
- Set/get number of subflows

- Set/get application profile
- Set/get preferences

- Subflow setup/termination
- Get/set fine-grained policies

Summary

- Current status: Identification of the requirements on the API
  - What control functions make sense for an app?
  - What information from an app would be beneficial for MPTCP?
  - What aspects could perhaps be implicitly determined?
- There is an initial list of requirements in the draft
  - As a starting point for the discussion
  - At a rather early stage so far
- Any feedback would be welcome!