Multipath TCP Architecture: Towards Consensus

draft-ford-mptcp-architecture-01

Alan Ford <alan.ford@roke.co.uk>

Goals of Draft

Overview of MPTCP

- Motivation and goals for Multipath TCP
- Functional architecture
- **High-level design decisions**
- How components (drafts) fit together

Motivation and Functional Goals

- Increase resilience of connectivity
  - => Use of multiple paths interchangeably
- Increase throughput
  - => Use of multiple paths simultaneously
  - Also increases efficiency of global resource utilisation

Compatibility Goals

- Application Compatibility
  - Appearance of MPTCP to the application
  - Maintain API compatibility (extensions permitted)
  - Maintain regular TCP service model
- Network compatibility
  - Look like regular TCP
  - Traverse common middleboxes
- Compatibility with other network users
  - Coexist gracefully with other TCP flows

MPTCP Functional Modules

- Path Management
  - Detection and use of multiple paths between endpoints: achieved through multiple IP addresses
- Packet Scheduling
  - Breaking bytestream from application into segments for transmission on subflows
- Subflow Interface
  - Sends scheduled packets with necessary metadata to lower TCP layers
- Congestion Control
  - Managed across subflows

High-Level Design Decisions

Sequence Numbering

- Two layers of sequence numbering
  - Keep each subflow-level independent
  - Add mapping to data-level sequence numbering
  - Full mapping of (data_seq, subflow_seq, length) but this could be implied in cases
- Reasoning
  - Appear as continuous flow to middleboxes
  - Allow packet loss and acknowledgement to be attributed to the correct subflow in the case of retransmission

Reliability

- ACKs at subflow level only
- Connection-level ACKs are therefore derived
- Pros:
  - Reduced complexity and overhead
- Cons:
  - If a subfow is ACKed and the data is later lost, the connection stalls permanently
    - Cases: middlebox failure; memory pressure
    - Are these things we need to worry about?
- *Do we need an explicit connection-level ACK?*

Retransmissions

- Dual-level sequence numbering and subflow-level ACKs can determine lost data
- Retransmission algorithms are TBD
- To maintain integrity of subflows, different data cannot be retransmitted on previously allocated subflow sequence space
- So even if data is retransmitted on another subflow, must also still be retransmitted on the previous one
  - This will affect optimal retransmission algorithms

Path and Connection Management

- Paths identified by IP address pairs
- MPTCP-aware applications can use a MPTCP-specific connection identifier (analogous to ephemeral port for demultiplexing)
- Legacy applications must be presented with a 5-tuple – the 5-tuple of the first subflow. Complications if this subflow closes:
  - Does connection have to close?
  - Note that binding to an address will not use MPTCP
  - Proposal: close connection unless extended API is used (or overridden in OS – out of scope)

Middleboxes (1)

- Dedicated section is still a TBD in the draft, but impact is felt throughout
- Has been a factor in the protocol design so far
- But where can we draw the line?

Some examples:

- Middleboxes that prevent connections in one direction (firewalls, NATs)
  - Solved by signalling addresses

Middleboxes (2)

- Terminating middleboxes (e.g. PEPs)
  - May do proactive ACKing
  - May drop TCP Options
  - (Fall back to behaviour as regular TCP)

- Middleboxes that care about TCP
  - May change sequence numbering
  - May do packet coalescing or splitting
  - Do not put holes in sequence space

Other Issues in the Draft

- Buffer sizes
  - Optimality TBD
- Signalling
  - See options vs payload discussion
- Support for both v4 and v6
  - Uncontentious?
- Congestion control
  - See separate draft
- Receive windows
  - Per-connection only – per-subflow could lead to deadlocks

Next steps?