Multipath TCP
Protocol Design

Alan Ford
alan.ford@roke.co.uk
Scope

• *To build TCP modifications to support multipath operation*

• We have more than one implementation already, but this presentation is about the details needed to be solved in any implementation – for WG evolution
Usage and Design Considerations

Source
9 8 7 6 5 4

9 8 6 4

7 5

Destination
9 8 7 6 5 4
Usage and Design Considerations

How do you:
• Discover paths and create subflows?
• Do sequence numbering to identify and reorder data to the application?
• Deal with changes in semantics and implementation, e.g. sequence numbering and SYN/FIN flags?
• Handle flow control and receive buffer depletion?
• Schedule appropriately?
Scenarios

- Bulk client/server transfers (e.g. HTTP/FTP)
- Short transactions (e.g. HTTP)
- Peer-to-peer transfers
- Interactive services (e.g. SSH, IM)
- Streaming services (NB buffered vs live)

- Where to deploy multipath TCP to give benefit?
Compatibility Goals

• Deployability is the key driver
• Performance should, in the worst case, be no worse than regular TCP over the best path
• It should appear compatible with regular TCP to unaware boxes on the wire
  – It should be able to seamlessly operate with legacy middleboxes (particularly NATs)
API Compatibility

• It should appear as regular TCP to applications
  – It provides the same service model: byte-oriented, in-order stream delivery
  – No mandatory API changes
• Essentially: is standard TCP, but with the potential to use multiple paths
Scheduling

- A scheduler decides how to distribute application data across available paths
- The scheduler also handles retransmissions, which may be over alternative paths
- Congestion coupling will be the subject of the next presentation
  - Goal: maximised throughput
- Other scheduling logic, e.g.
  - Goal: increased resilience and failover
  - Dependencies on path properties, e.g. cost, b/w
Signalling

If signalling is required (e.g. addresses, sequence numbering), how to do this?

• In the payload?
  – A chunking mechanism (using types) would be very clearly an application-layer rather than a transport-layer solution

• As TCP options?
  – Currently preferred in the draft solutions
  – Existing extension mechanism
  – Limited space so keep signalling to a minimum
Sequence Space

Shared or separate sequence spaces?

• Single sequence space, across all paths
  – Simply send each TCP segment on one of the available paths

• Create a data sequence space, leaving the individual subflow TCP sequence spaces untouched
  – Both ends aware of multiple TCP connections: clear distinction between paths and data.
Two Proposals

We have two example proposals for locating functionality, for different usage scenarios:

• “One-ended”
• “Two-ended”

• Both appropriately schedule packets over multiple paths
• These are implemented examples – but not the only way to solve the problem!
One-Ended MPTCP

draft-van-beijnum-1e-mp-tcp

- Multihomed hosts with PI addressing can distribute packets across multiple links
- Only sender needs to be modified
- One source, one destination address
- Need to recover per-path acknowledgements from SACK
- Do per-path congestion control
Two-ended MPTCP

draft-ford-mptcp-multiaddressed

• Start with single TCP “subflow”

• Initiate additional subflows
  – Which have different source/destination address pairs
  – Use identifier to merge with existing subflow

• Can be done from a host's additional interfaces, or signalled to the other endpoint
  – To get around NATs/firewalls
  – Can also allow simultaneous IPv6/4 usage
Two-ended MPTCP: Details

• To middleboxes, subflows look like regular TCP sessions (with extra options)
  – Operate independently regarding FIN etc

• Two sequence spaces:
  – Data-level sequence number in TCP option for reassembly
  – Each subflow maintains its own TCP-level sequencing
Security

• We want a *no worse than TCP* security
  – And quite possibly a migration path to improve
• One-ended is basically TCP as it stands
• Two-ended solution must consider similar issues to mobility/shim6
• Need to avoid redirection attacks when adding and removing subflows
Summary

- For more information: [http://trac.tools.ietf.org/area/tdv/trac/wiki/MultipathTcp](http://trac.tools.ietf.org/area/tdv/trac/wiki/MultipathTcp)
- See current proposals:
  - draft-ford-mptcp-multiaddressed-01
  - draft-van-beijnum-1e-mp-tcp-00
  - Design space discussion document