Multipath TCP for FreeBSD

Lawrence Stewart, Nigel Williams, Grenville Armitage

{lastewart,njwilliams,garmitage}@swin.edu.au

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Outline

- NewTCP/MPTCP Project Overview
- Key Architectural Points
- Implementation Update
- Some Initial Discussion Points
NewTCP/MPTCP Project Overview

- **NewTCP**: broader context for MPTCP work
  - CC algorithms (HTCP, Vegas, CUBIC, HD, CHD, CDG), frameworks (mod_cc, khelp/hhook), tools (SIFTR, DPD)
  - Various incarnations supported by Cisco

- **MPTCP project details**
  - PI: Grenville Armitage
  - R&D Engineer: Nigel Williams
  - Technical advisor: Lawrence Stewart
  - Cisco: Fred Baker, Alan Ford
MPTCP Project Overview

- Goals
  - Investigate per-subflow CC algo use & issues
  - Interoperability
  - Flexible implementation to support above & future work e.g. research into TX scheduling, retransmission strategies/scheduling, etc.
  - Release BSD licenced FreeBSD patches
  - Publish peer-reviewed outcomes & findings

- Non-goals
  - Optimised & commit-ready implementation
  - Full specification compliance
Architecture: Where to start

- New stack protocol (e.g. hooks) or shim?
- Tight or loose coupling with existing TCP code for subflows?
- Data structures?
- SMP
- Platform differences
Architecture: Integration With TCP

- Shim tightly coupled with TCP code
- Migrate session management into shim
- Tweak control data structure relationships

**RX side**
- Merge TCP reassembly + in-order delivery queue
- Defer data-level reassembly to user context

**TX side**
- Map chunks of socket buffer to subflows
Architecture: Control Data Structures

Before:

Socket —> inpcb —> tcpcb

After:

Socket —> inpcb (master) —> tcpcb (master) —> tscb

inpcb (subflow) —> tcpcb (subflow)
Architecture: RX Data Structures

Before:

- tcpcb
- List *segq
- Reass segq List
- Seq 3
- Seq 4
- RX Sockbuf List
- Seq 1
Architecture: RX Data Structures

After:

```
tscb
List segqs[]

SF1 tcpcb
... List *segq ...

SF 1 segq List
... Seq 24 DSN 1 ...
Seq 26 DSN 3

SF n segq List
Seq 62 DSN 4
```
Architecture: TX Data Structures

Before:

TX sockbuf

```plaintext
snd_nxt  snd_una

tcpcb
```
Architecture: TX Data Structures

After:

TX sockbuf

UNMAPPED  SF2 MAP  SF1 MAP

snd_nxt  sf_una

DS MAP

SF 2 tcppcb

snd_nxt  sf_una

DS MAP

SF 1 tcppcb

ds_snd_una
Architecture: SMP

- Reader-writer locks, sensible data structures & access patterns to minimise lock contention

- **TX**
  - Sub-flows rlock sockbuf to send mapped data
  - MP shim wlocks sockbuf when allocating new map or freeing ACKed data
  - User context wlocks sockbuf to write()

- **RX**
  - Sub-flows rlock seg queues array to enqueue
  - User context wlocks seg queues array to reassemble
Segment arrives on subflow 1

Segment fills hole. Call 'sorwakeup' to wake process

Schedule subflow and data-level ACKs

Insert into segment list

1
2
3
4
5

SF1
SF2

tscb
Architecture: RX Data Delivery

Application

read()

user space

kernel

Perform reassembly

tscb

uipc_socket

In-order data

1 2 3 4 5

1 2 3 4 5
Implementation: BSD ain't Linux

- Stack architecture is very different
  - Be mindful to give generally relevant advice

- Stack inherently byte based
  - Socket buffer is contiguous
  - Differing MSS between subflows a non-issue
  - Be mindful of pkt-based stack assumptions
Implementation: Status

- Code
  - Control data structures, option parsing, RX reassembly → segment queue, DS MAP & ACK

- WIP
  - Fully deferred RX data delivery
  - Mapping >1 subflows onto TX sockbuf
  - MP_JOIN + associated machinery
Implementation: Interop

- FreeBSD ↔ FreeBSD
  - Single MP-enabled subflow works
- FreeBSD ↔ Linux
  - Handshake
  - DS map exchange
Points of Discussion

- DS map checksumming (pg24)
- Initial DS ACK from passive → active opener? (maybe pg14 p.3?)
- Extending right-edge of existing DS map?
- Expand implementor's draft (SMP)
- DS SND.NXT rename/clarification? (pg55)