SWINBURNE UNIVERSITY OF TECHNOLOGY

Multipath TCP for FreeBSD

Nigel Williams

njwilliams@swin.edu.au

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

Outline

- MPTCP Overview
- Our implementation
  - Design overview.
  - Where it's at and future.
  - Experience implementing a draft protocol.

CAIA MPTCP Development Team

- Nigel Williams (njwilliams@swin.edu.au)
- Lawrence Stewart (lastewart@swin.edu.au)
- Project Website: http://caia.swin.edu.au/newtcp/mptcp/
  - Documentation and Kernel Patch (v0.3)

About me

- B. Eng (Telecoms), Swinburne University of Technology
- Diploma of Music Industry (Technical Production)
- …and almost half a Computer Science degree
- Centre for Advanced Internet Architectures, Swinburne University of Technology (2005-2007, 2012-)
  - Mostly traffic classification/QoS work
- FreeBSD Newbie

MPTCP – What is it?

- RFC 6824 (Experimental): TCP Extensions for Multipath Operation with Multiple Addresses
  - A. Ford, C. Raiciu, M. Handley, O. Bonaventure, S. Barre
- Allows a multi-homed host to spread a single TCP connection across multiple interfaces.
- Maintains backwards-compatibility with existing TCP applications.

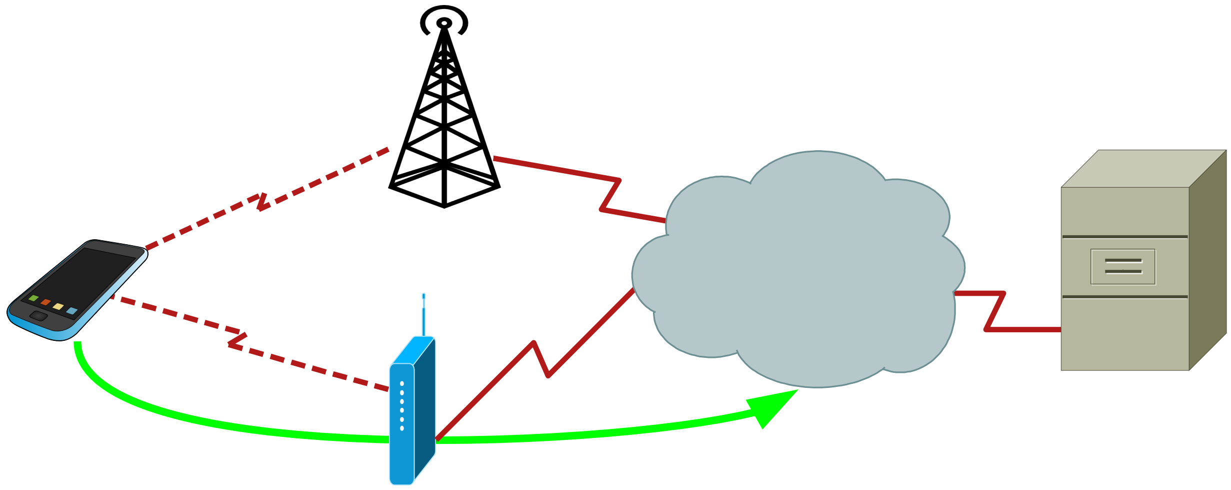
Motivations

- Devices have more interfaces
  - Mobile devices: Wifi + 3G
  - Netbooks: Wifi + 3G + Ethernet
  - Data centres with multi-homed servers
- Many applications use TCP
  - But TCP doesn't take advantage of these extra interfaces.
  - Thus TCP connections are broken and re-established when end hosts shift network connectivity between interfaces.
- Middleboxes/NAT make it difficult to use SCTP, other solutions.

Example Scenario

- Mobile TCP Session

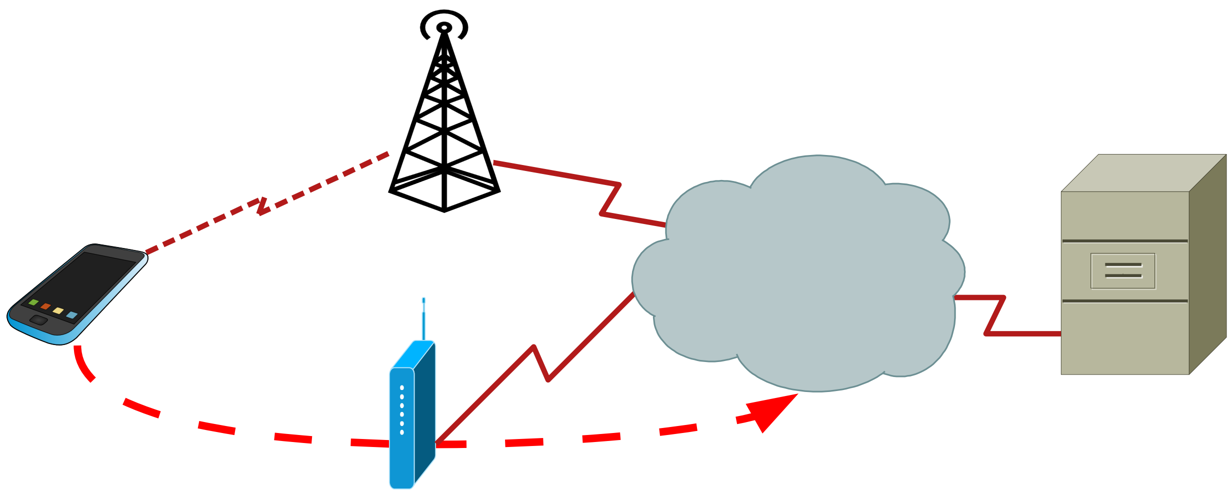
  Uses only one of the available paths.



Example Scenario

- Mobile TCP Session

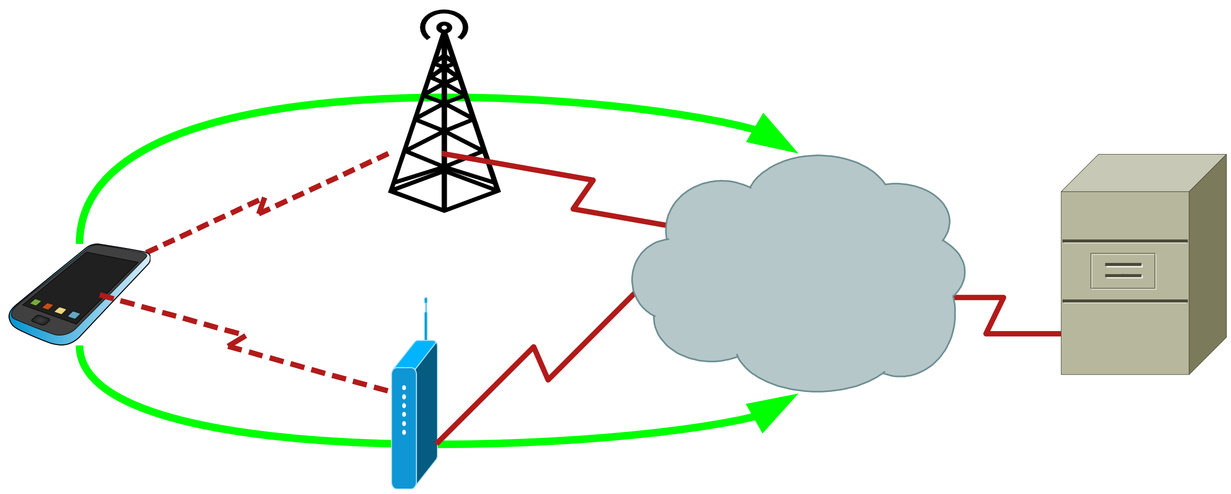
The connection drops when wifi is no longer available.



Example Scenario

- Mobile MPTCP Session

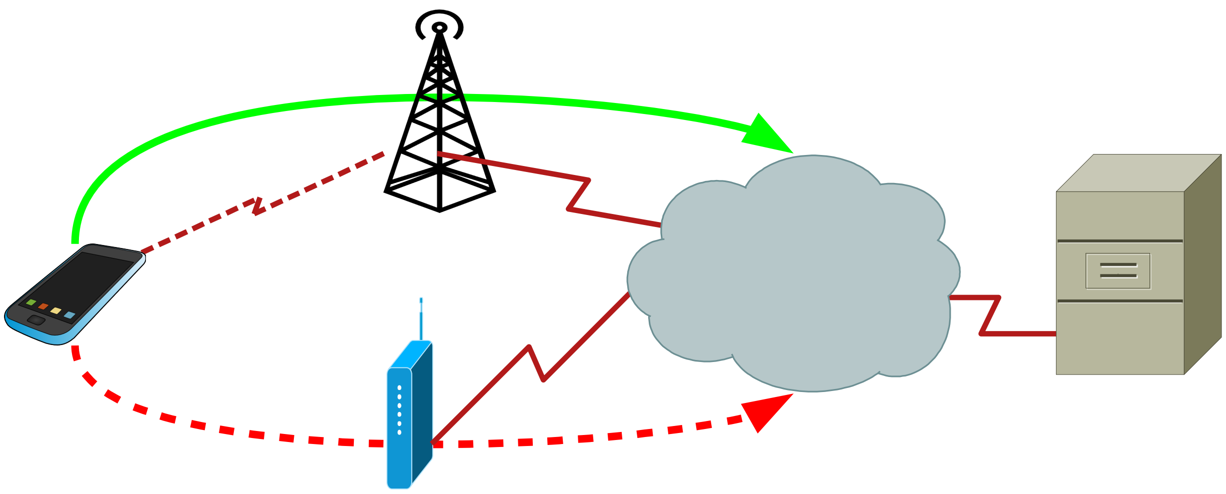
  The connection now has multiple paths associated with it.



Example Scenario

- Mobile MPTCP Session

The connection is maintained by moving traffic to 3G when wifi fails.

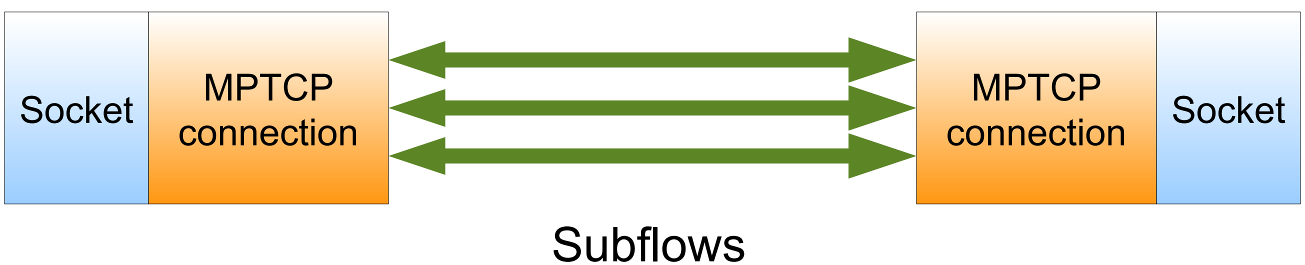


Benefits

- Adds Redundancy and Persistence
  - Maintains a connection when links fail.
  - Break before make.
- Reduces Congestion
  - Paths aren't fixed - use congestion control to steer traffic away from congested links.
- Increases efficiency
  - Take advantage of additional interfaces, parallel paths.
- Works with existing TCP applications
  - In-kernel, backwards compatible with existing TCP socket APIs.

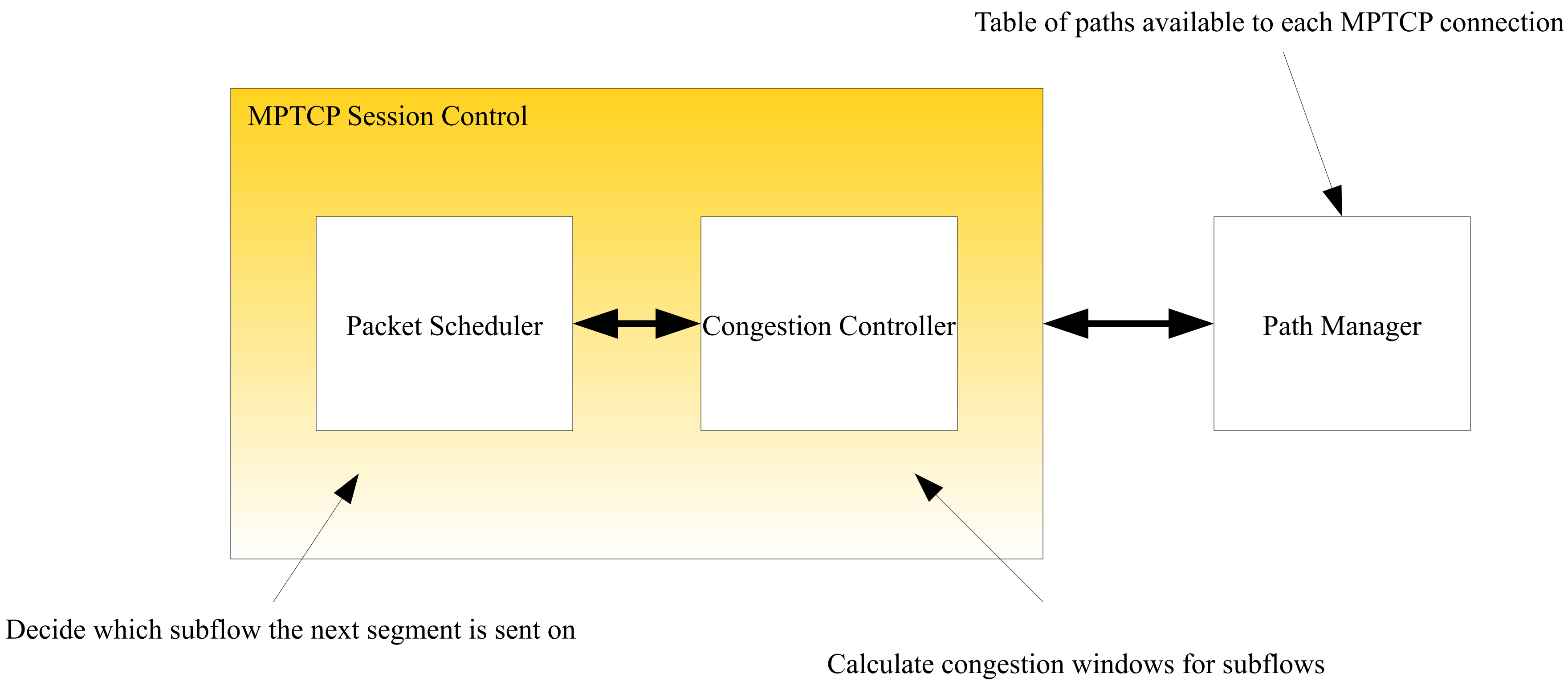
MPTCP Design

- A MPTCP connection is made up of one or more *subflows*
  - Each of the subflows acts much like a standard TCP session.
  - Application 'sees' a single, standard TCP connection.
  - Signalling uses TCP Options field – there is a new MPTCP option, which has its own subtypes.
  - Use coupled congestion control (but possibly other methods) to help decide which subflow to send traffic on.



MPTCP Design

- Logical Components



Signaling

- MPTCP Control messages passed in TCP options field

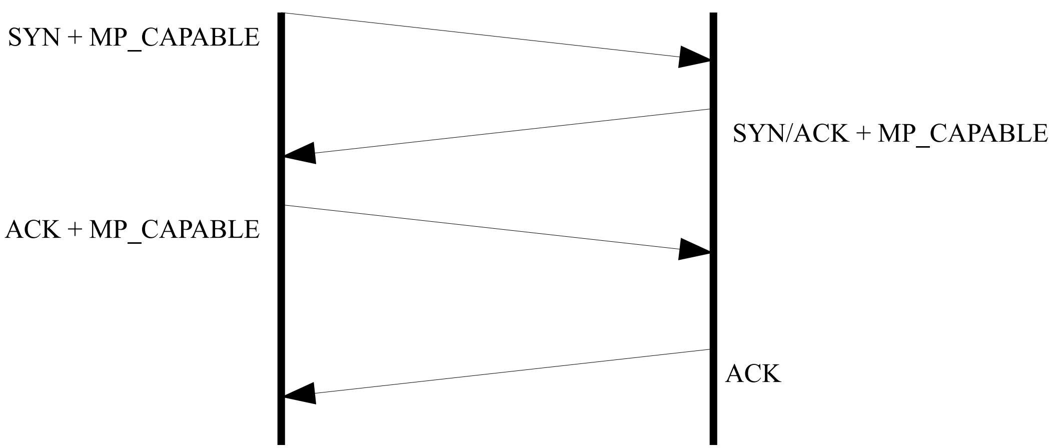
20 Bytes	40 Bytes	
TCP Header	TCP Options	Data

- MPTCP subtypes

Symbol	Name	Value
MP_CAPABLE	Multipath Capable	0x0
MP_JOIN	Join Connection	0x1
DSS	Data Sequence Signal (Data ACK and Data Sequence Mapping)	0x2
ADD_ADDR	Add Address	0x3
REMOVE_ADDR	Remove Address	0x4
MP_PRIO	Change Subflow Priority	0x5
MP_FAIL	Fallback	0x6
MP_FASTCLOSE	Fast Close	0x7

Signaling

- E.g Connection Setup
  - Piggyback on 3-way handshake
  - The MP_CAPABLE option is included in the handshake phase
  - Though it's actually a 4-way handshake before multipath is enabled
  - MP_JOIN adds new subflows to established connections

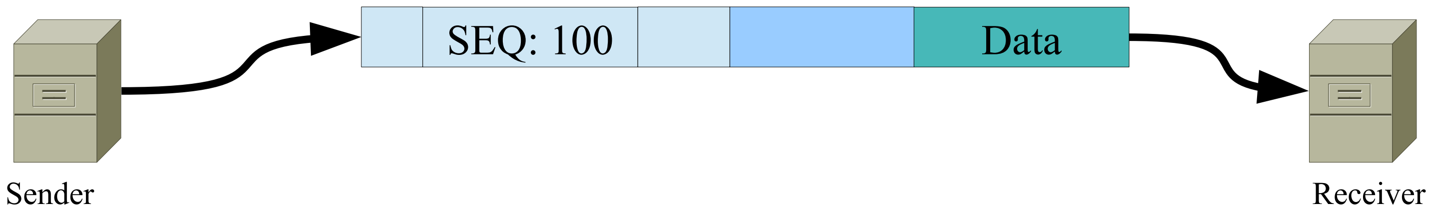


Data Sequence Space

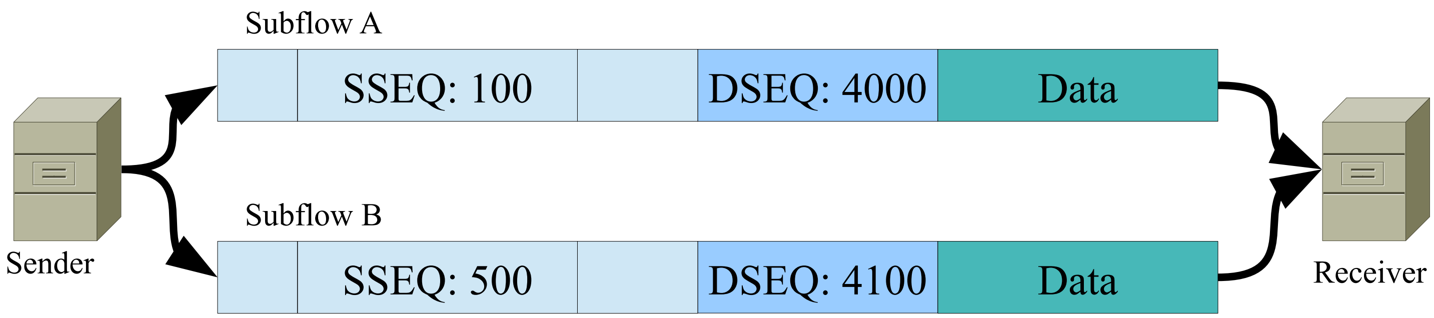
- How does regular TCP work?
  - Segment data, then use sequence numbers to track the segments.
  - Allows acknowledgment, retransmits, out-of-order reassembly etc.
- MPTCP needs to aggregate segments from multiple subflows
  - Paths may have different BW, RTT, so re-ordering can happen.
  - Each subflow should retain its own sequence space (e.g. to prevent trouble with middleboxes).
- Connection-level sequence space
  - Use 64-bit *data sequence numbers* to aggregate segments from multiple subflows. Pass data sequence numbers as TCP options.

Data Sequence Space

Standard TCP sequence numbering



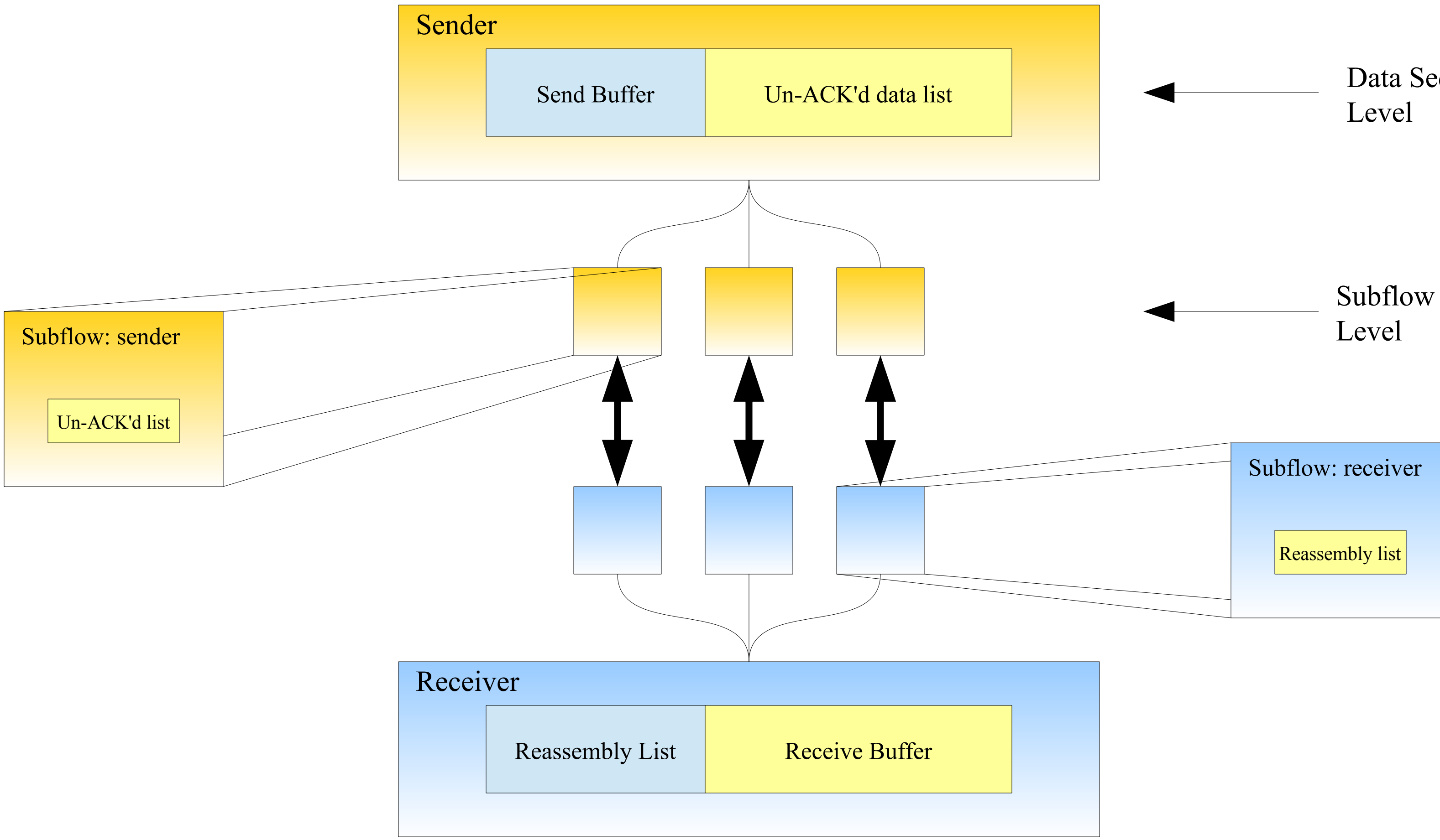
MTCP sequence numbering – with subflow sequence and 64-bit data sequence numbers



Data Sequence Space

- Map DS Space to subflow space
  - DSN Maps cover one or more segments.
    - Subflow A: DSN: 4000, Len: 100
    - Subflow B: DSN: 4100, Len: 300
- Data-Level acknowledgements, RTO, retransmits
  - ds_rcv_nxt
  - ds_snd_nxt
  - ds_snd_una

Logical Sequence Spaces



Congestion Control

- Designers recommend a “resource pooling” congestion control algorithm
  - The CC algorithm uses subflow cwnd, aggregate cwnd and RTT to adjust the window.
    - Moves segments away from congested links (Favours links with higher capacity).
    - Fair to standard TCP at bottlenecks.
    - Treats multiple links as a single pool of capacity.
- Not part of the MPTCP specification – other approaches can be investigated.

Our Implementation

- Designed with research in mind
  - Hooks that make it easy to twist knobs and pull levers (adjust cc, retransmit strategies, access to subflows).
  - BSD-licensed implementation benefits other researchers and vendors.
  - Non-goal: an optimised & commit-ready implementation
- An interoperable FreeBSD implementation assists standardisation efforts.

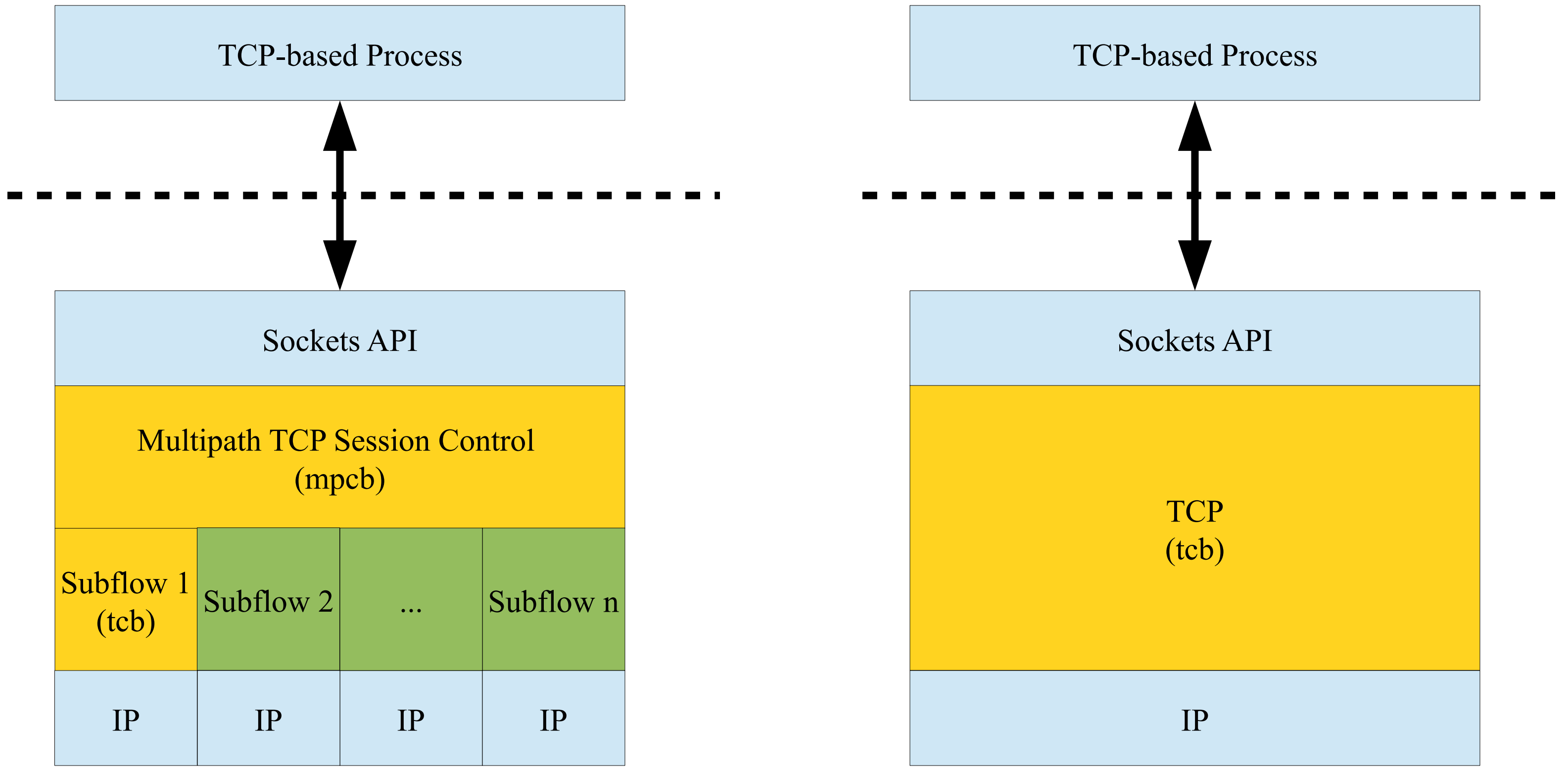
Architecture: Where to start

- New stack protocol (e.g. hooks) or shim?
- Tight or loose coupling with existing TCP code for subflows?
    - Sockets within a socket for subflows?
- Data structures?
    - Aggregating segments, data-level reassembly.
- SMP
    - Minimise lock contention between subflows.

Architecture: Integration With TCP

- Shim tightly coupled with TCP code.
- Tweak control data structure relationships.
  - MPCB for connection, TCBs maintain state for subflows.
- Receive side
  - Merge TCP reassembly and in-order delivery queue (segment queue).
  - Defer data-level reassembly to user context*
- Send side
  - Map chunks of socket buffer to subflows.

Architecture: MPTCP Shim



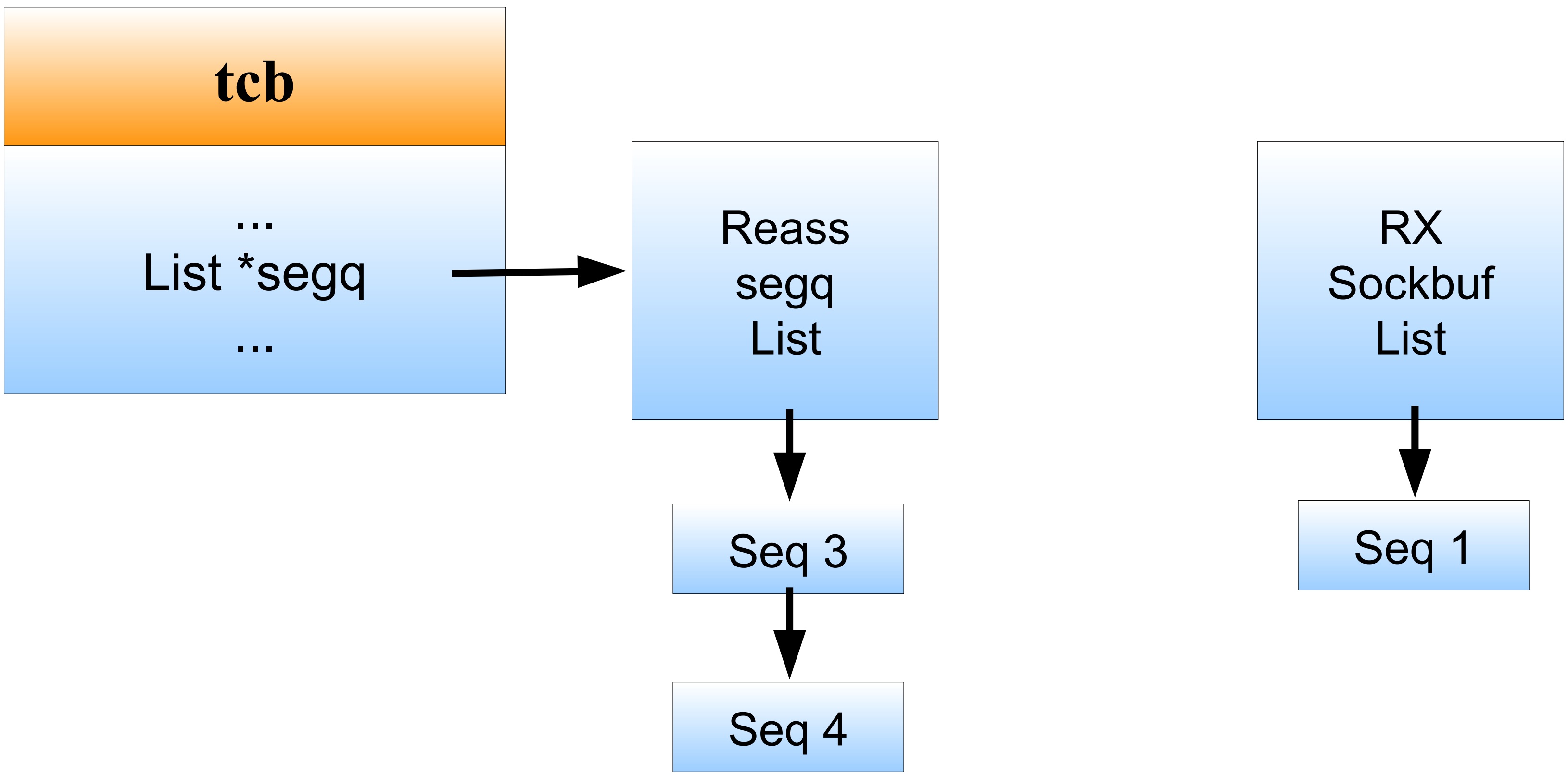
*An MPTCP Connection with a single subflow acts like standard TCP

Architecture: ds_map

- Subflows share socket buffers (so_snd and so_rcv)
  - Must map data to subflows to minimise lock contention.
  - Subflows deal with ds_maps rather than socket buffers.
- Used for send and receive functions (rxmaps, txmaps)
  - Tx: Mediate access between subflow and socket buffer.
  - Rx: Track accounting information for received DSN maps.

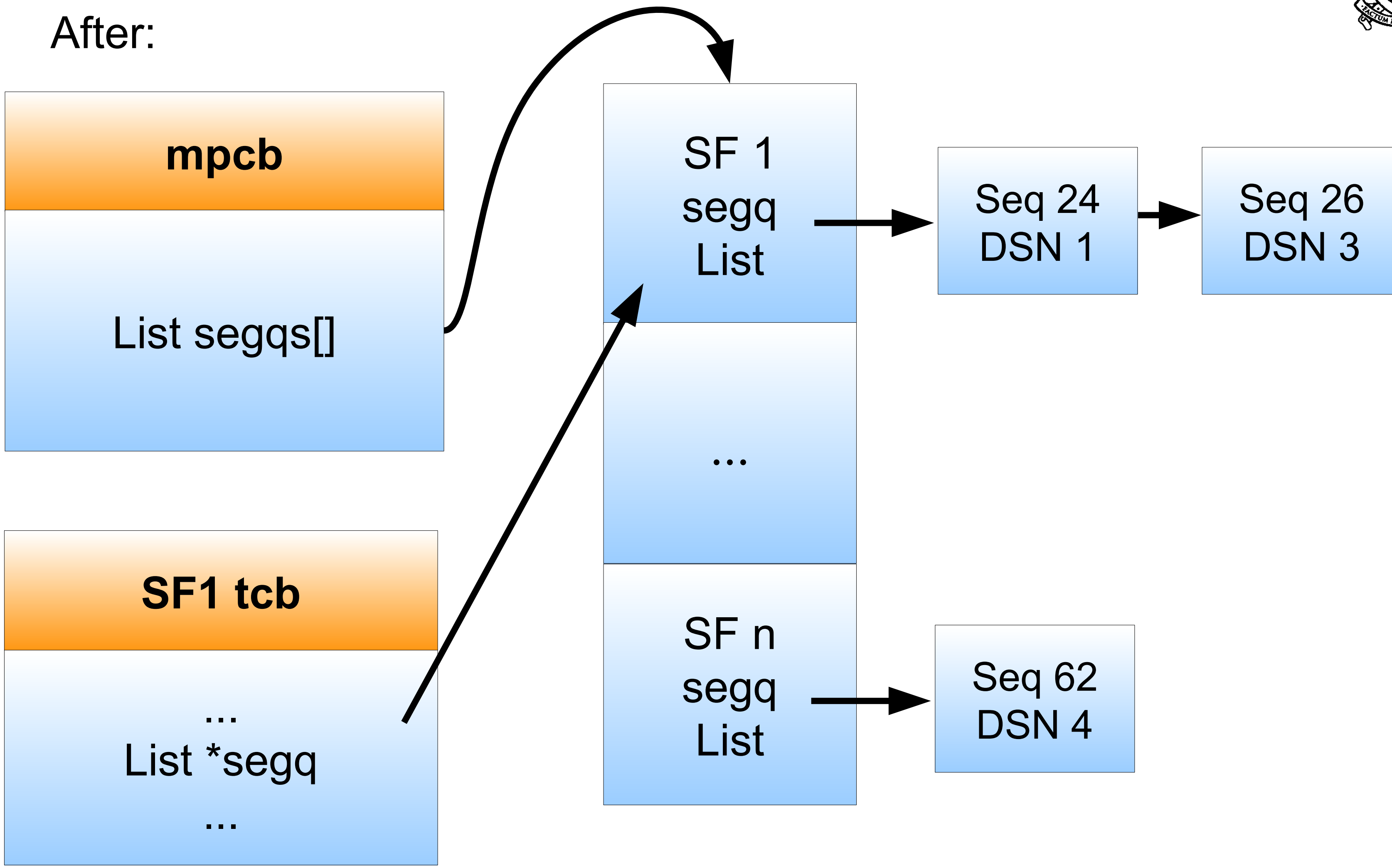
Architecture: RX Data Structures

Before:

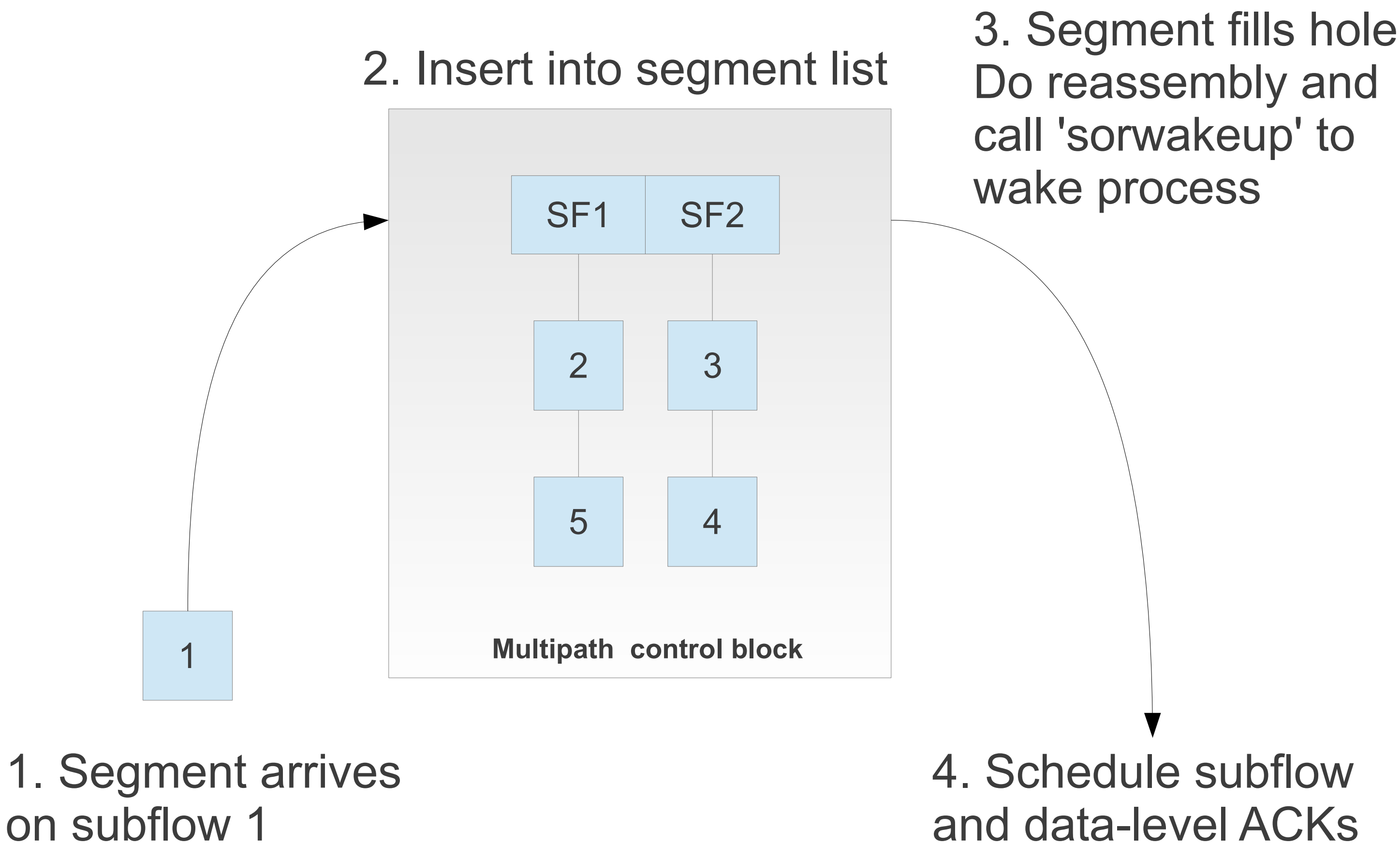


Architecture: RX Data Structures

After:

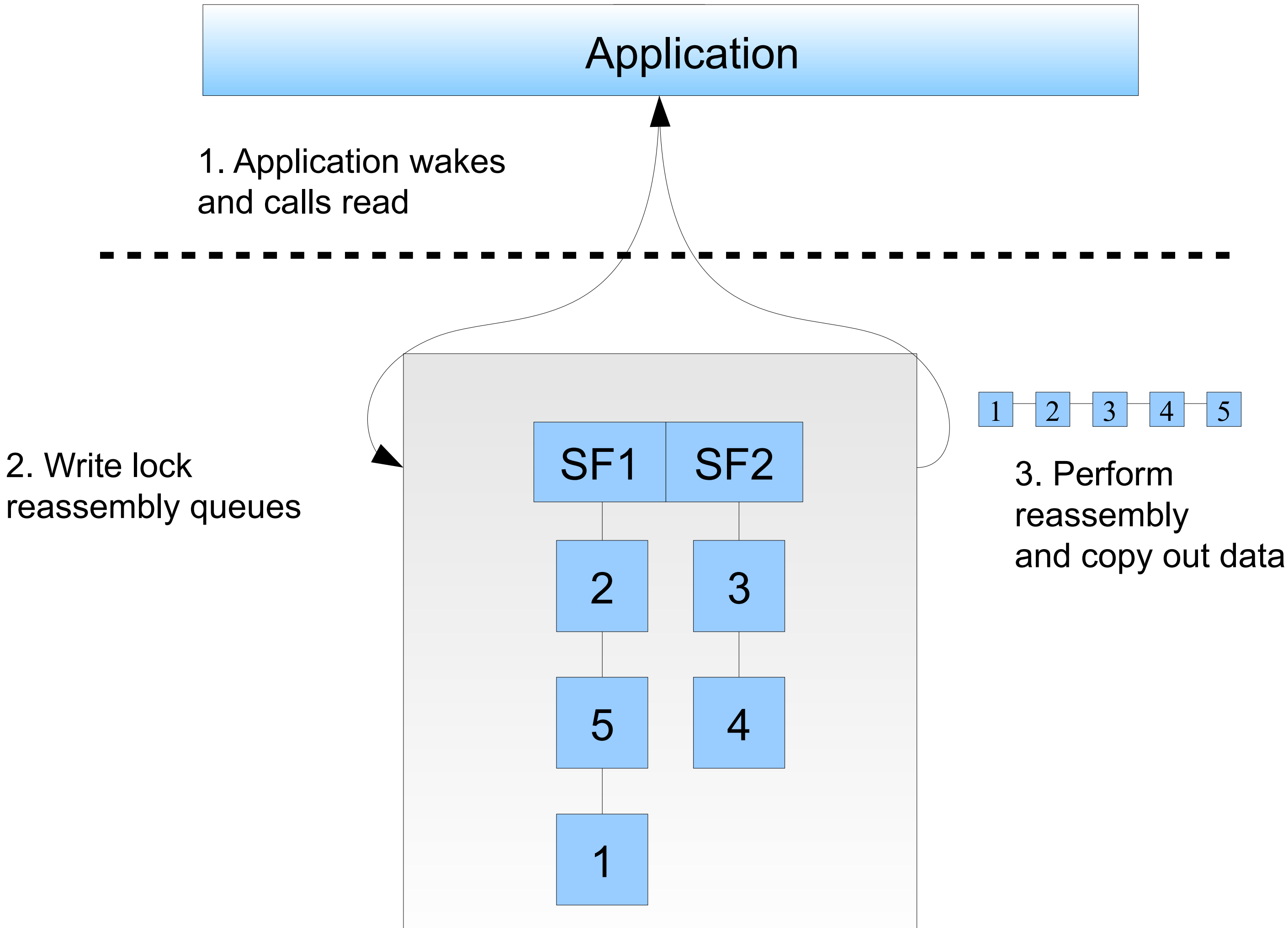


Architecture: RX Data Delivery



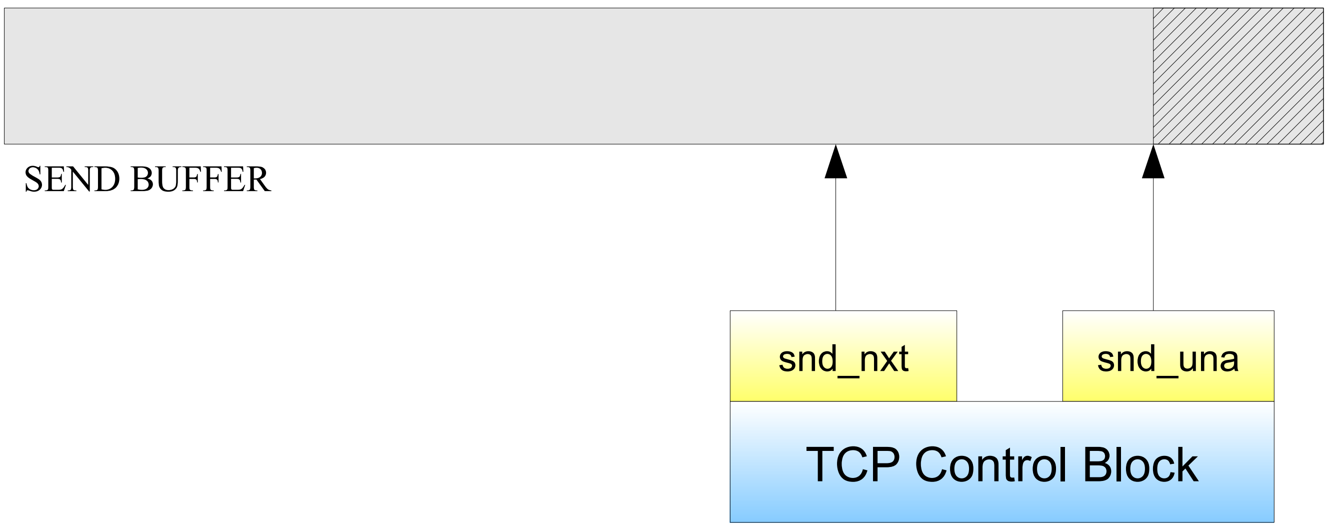
*Segments are in subflow sequence order. Data Sequence numbers shown

Architecture: RX Data Delivery



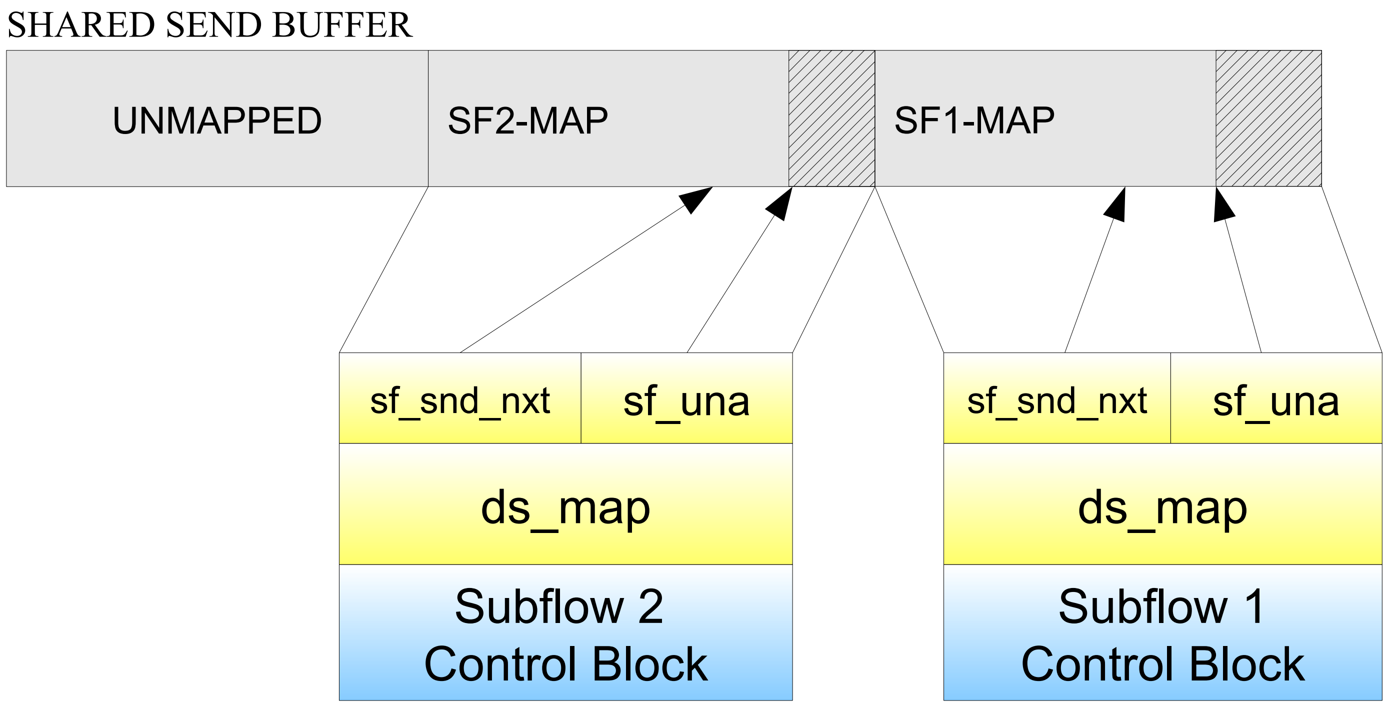
Architecture: TX Data Structures

Before:



Architecture: TX Data Structures

After:



Socket Buffer Accounting

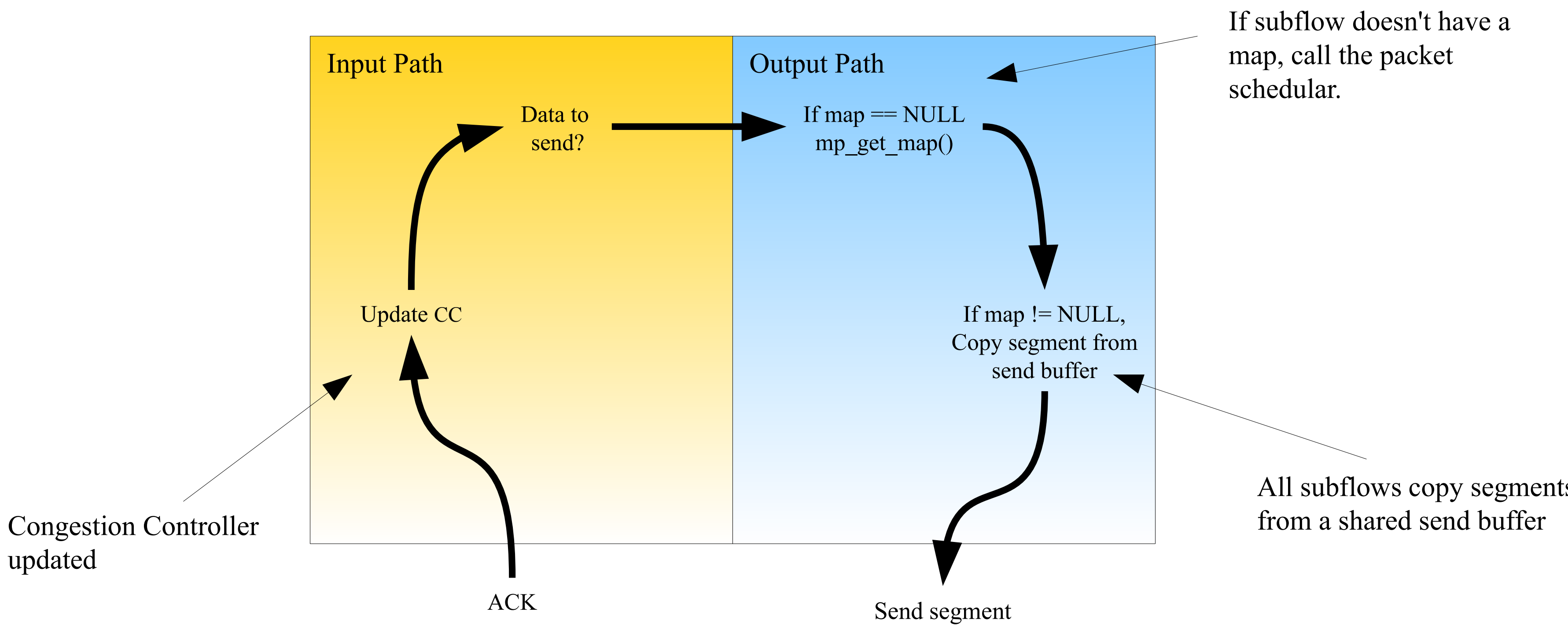
- Need to track socket buffer logically
  - Drop sent bytes based on Data ACKs
  - Overlapping sections of so_snd can be mapped
  - Map might only partially cover Mbuf chain
  - sb_cc is the 'logical' number for accounting, sb_actual is the true length of the buffer.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Packet Scheduler

- Determine which subflows are able to send data, and how much then can send
    - Currently basic schedular only (this is an area of research)
- Allocates ds_maps – A subflow will not call back in until the map is exhausted

Simplified Sender Path



Other Stuff

- Lots of supporting code
  - Option parsing, hashing, list manipulation..
  - Modifications to: tcp_input, tcp_output, tcp_subr, tcp_syncache, tcp_usrreqs, tcp_reass, uipc_socket, uipc_sockbuf … plus new source files

Research Projects

- MPTCP for Vehicle to Infrastructure communications
  - Mobile Data and 802.11p
  - Entertainment, telemetry applications
- Congestion Control
  - Per subflow CC selection, dynamic adjustment
  - Combine loss-based and delay-based CC
- Packet Scheduling

Observations

- Harder than it looks
  - Impact of early design decisions.
  - Some changes required extensive re-factoring (and many late nights!).
  - Accounting sucks.
- Interoperability
  - Some holes and assumptions in specification.
  - Unexpected on-wire behaviour.
  - Ongoing draft revisions.

Future Work

- Ongoing improvements
  - More functionality, bug-fixes
  - Documentation
  - Interoperability
  - Userspace API
- Research

Acknowledgments

- Cisco Systems

- BSDCan Committee

Links

- Further Reading
  - Design Overview of Multipath TCP version 0.3 for FreeBSD-10: http://caia.swin.edu.au/reports/130424A/CAIA-TR-130424A.pdf
  - RFC 6824 - TCP Extensions for Multipath Operation with Multiple Addresses: http://tools.ietf.org/html/rfc6824
  - How hard can it be? Designing and Implementing a Deployable Multipath TCP: http://inl.info.ucl.ac.be/system/files/nsdi12-final125.pdf
- Project website: http://caia.swin.edu.au/urp/newtcp/mptcp/
- Linux kernel MultiPath TCP project: http://mptcp.info.ucl.ac.be/