SWINBURNE UNIVERSITY OF TECHNOLOGY

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
  - http://caia.swin.edu.au/urp/newtcp/mptcp/
  - 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

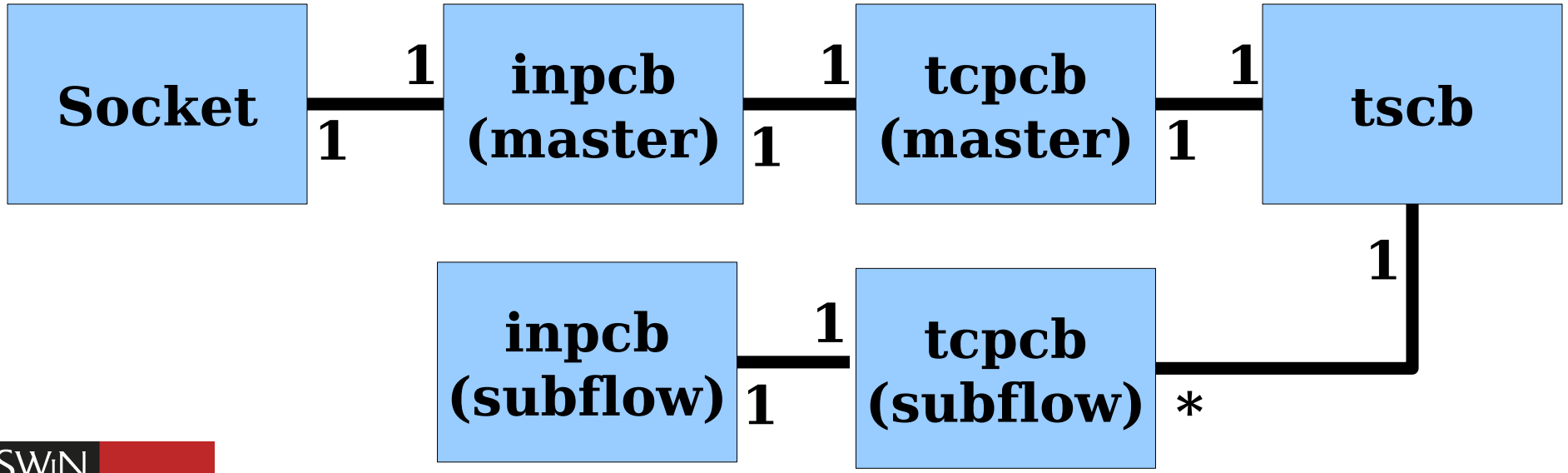
Architecture: Integration With TCP

- 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

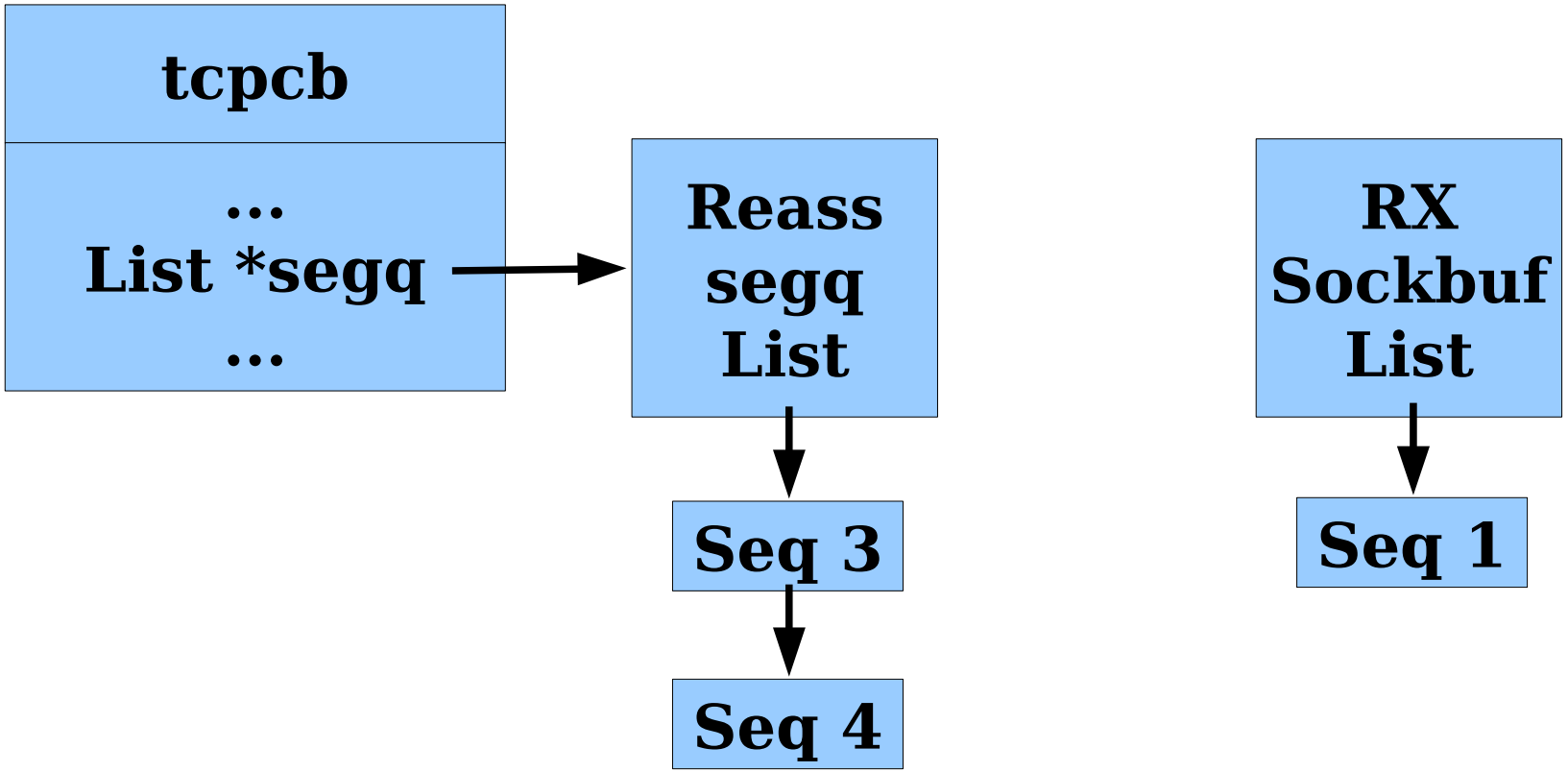
Before:

After:



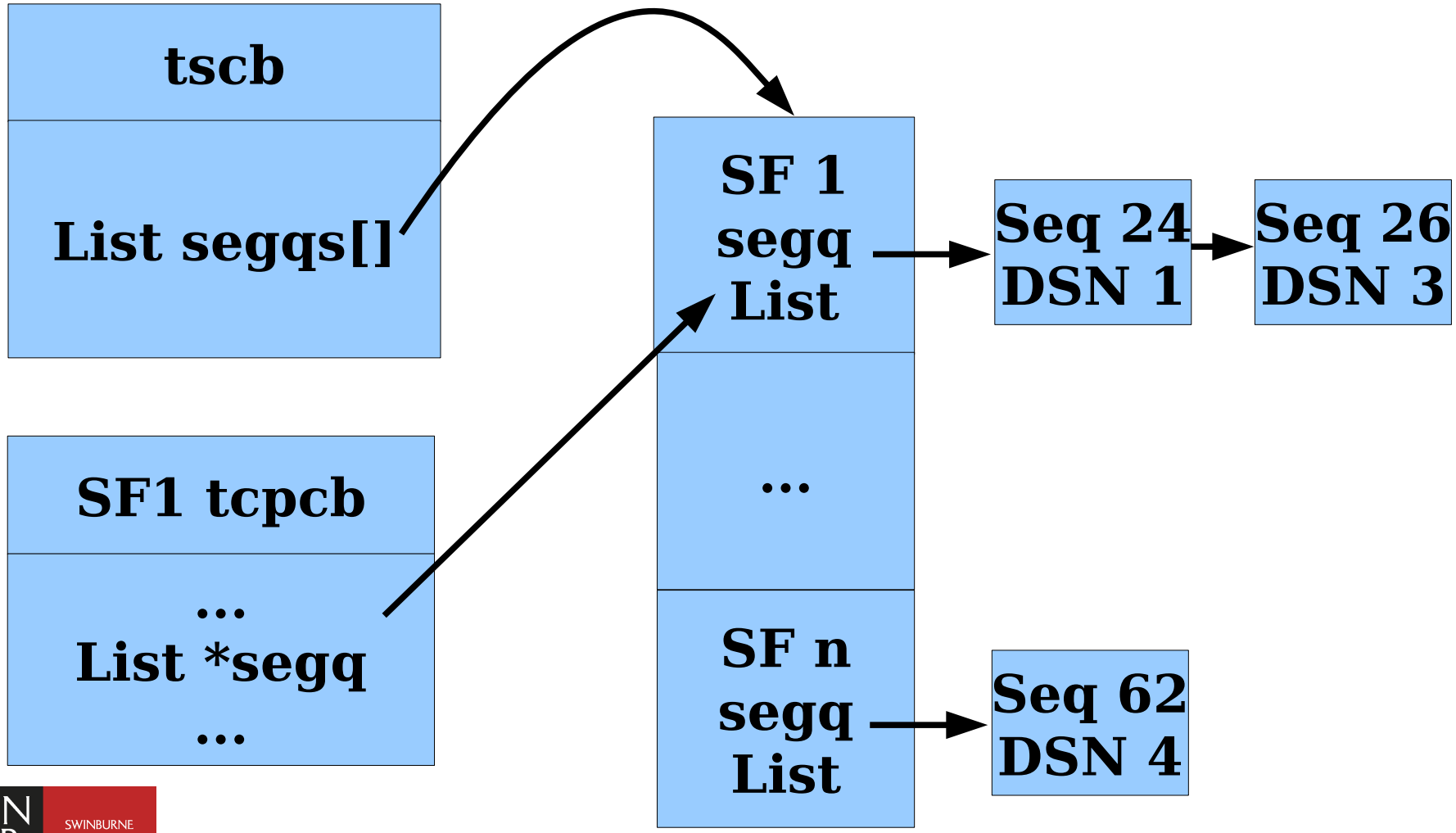
Architecture: RX Data Structures

Before:



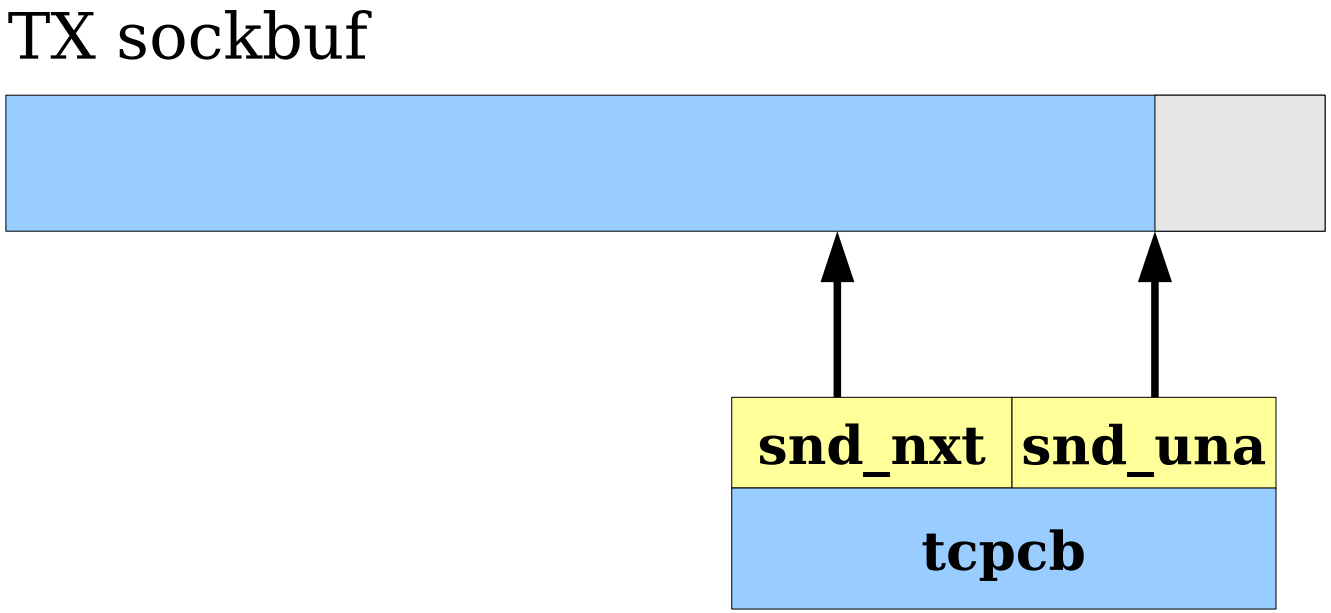
Architecture: RX Data Structures

After:



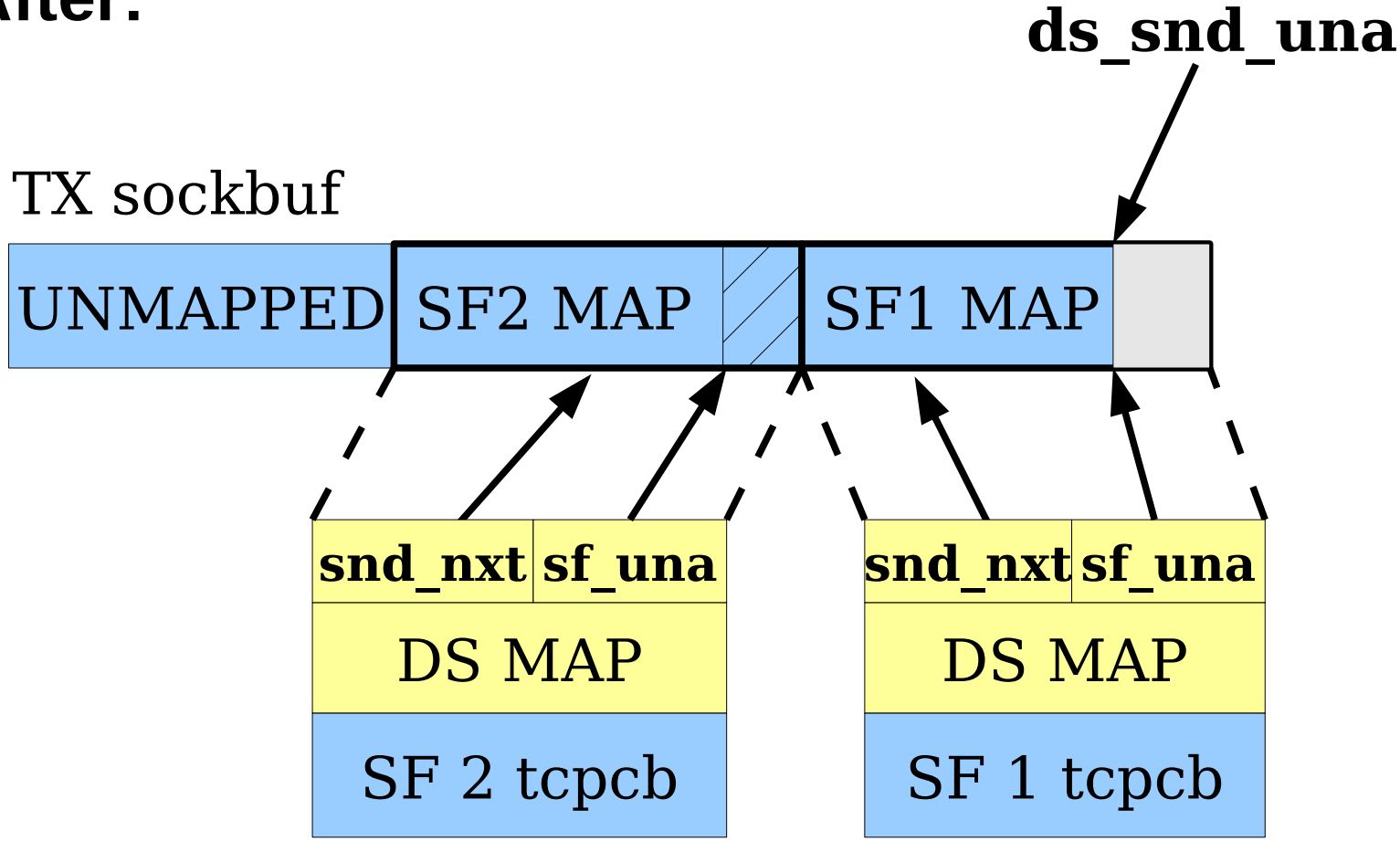
Architecture: TX Data Structures

Before:



Architecture: TX Data Structures

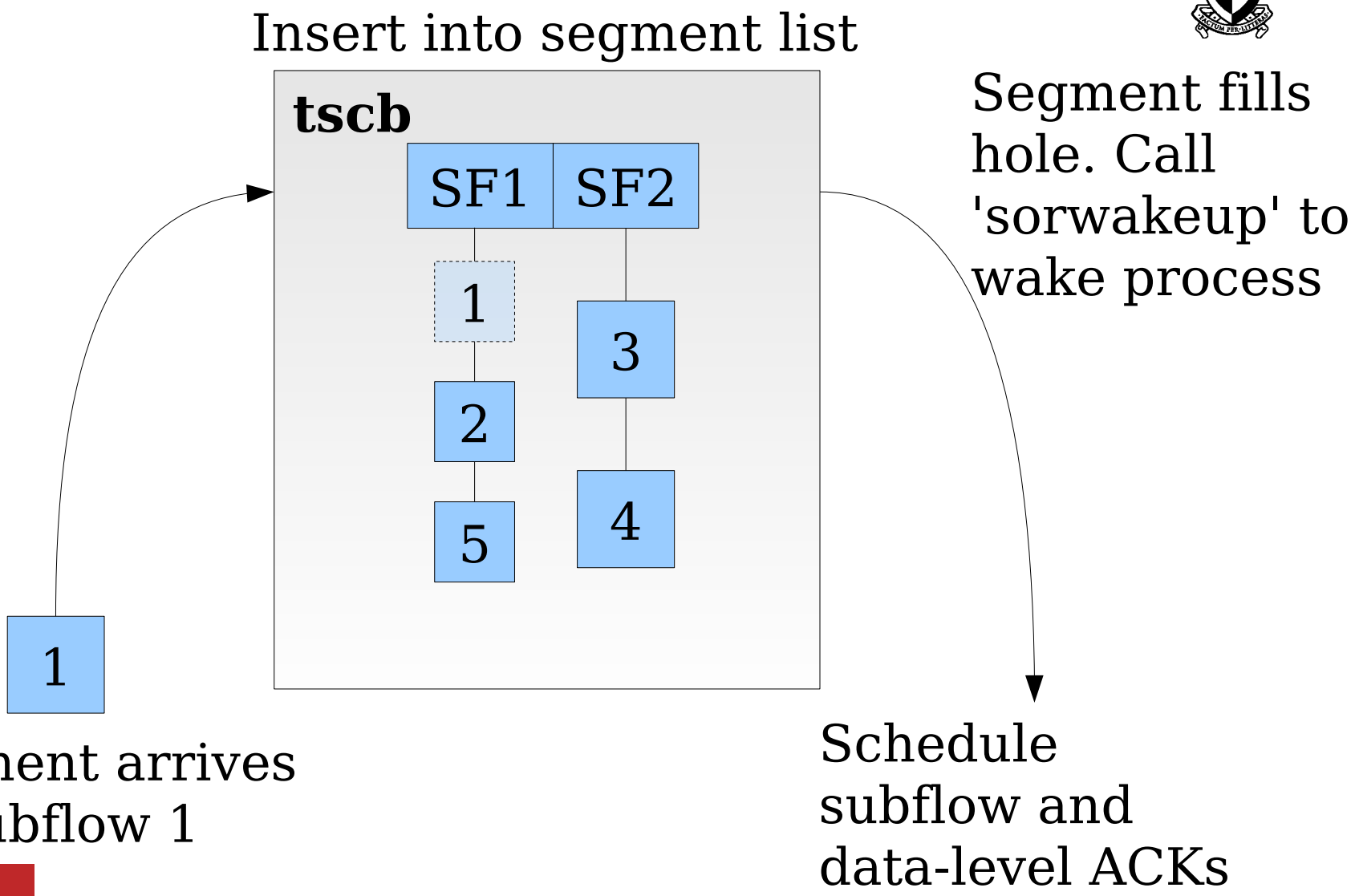
After:



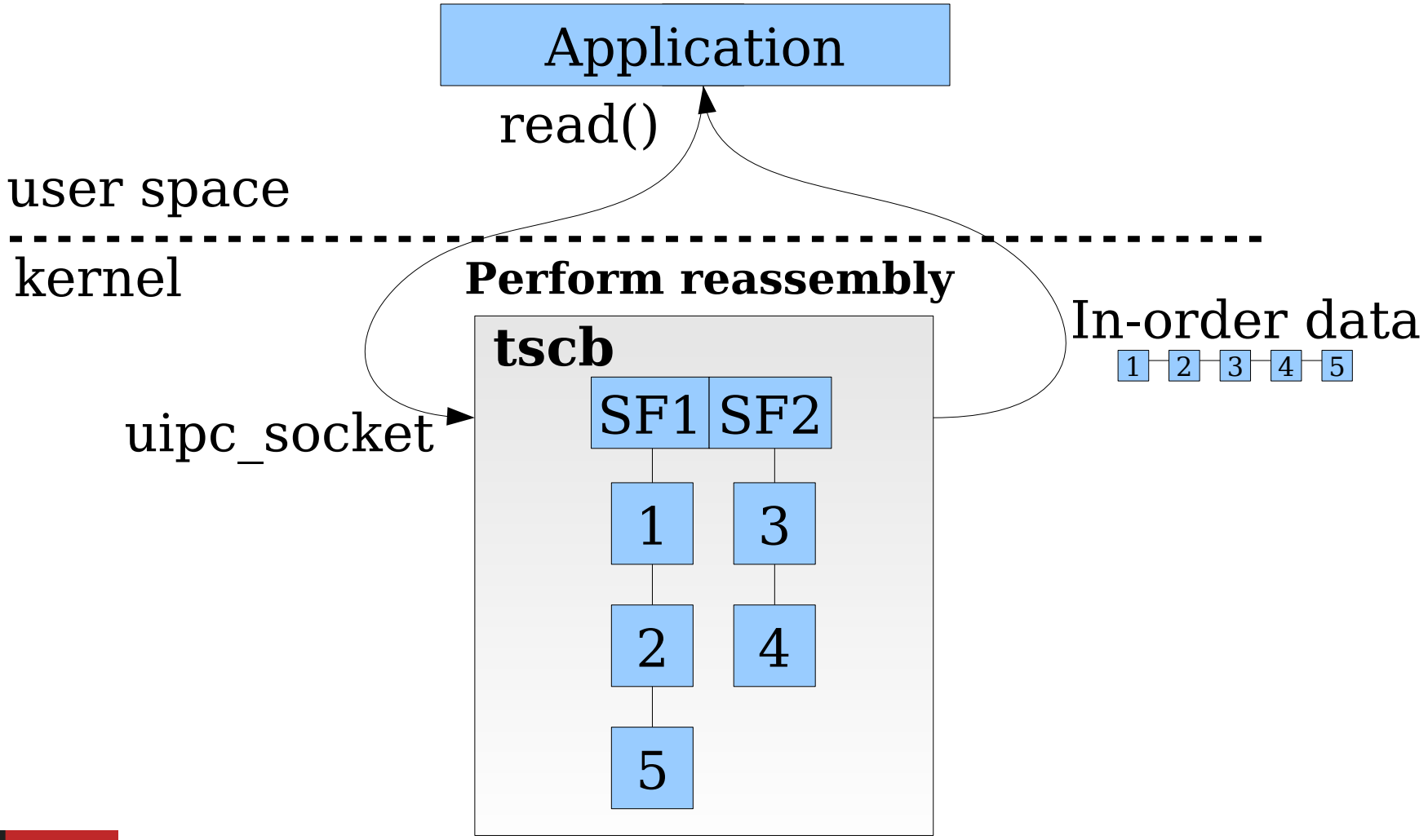
Architecture: SMP

- 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 enque
  - User context wlocks seg queues array to reassemble

Architecture: RX Data Delivery



Architecture: RX Data Delivery



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)