UNSW SYDNEY
Australia's Global University

School of Computer Science & Engineering

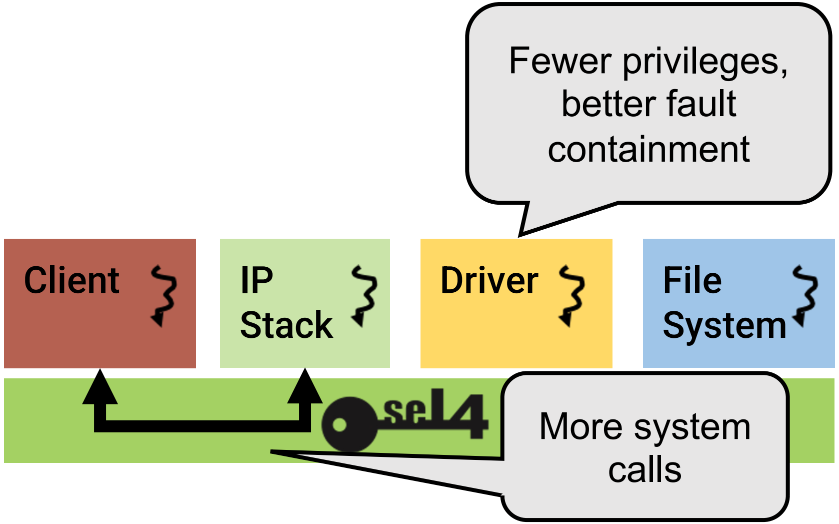
Trustworthy Systems Group

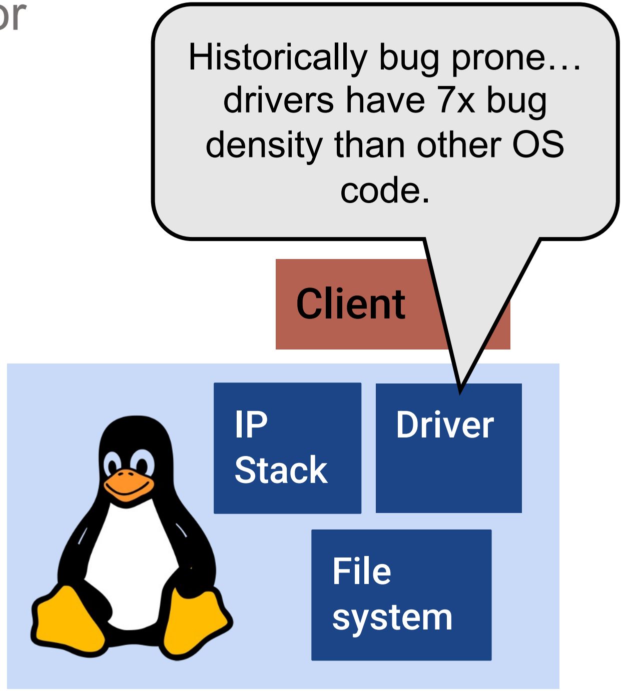
The seL4 Device Driver Framework

Lucy Parker
lucy.parker@student.unsw.edu.au

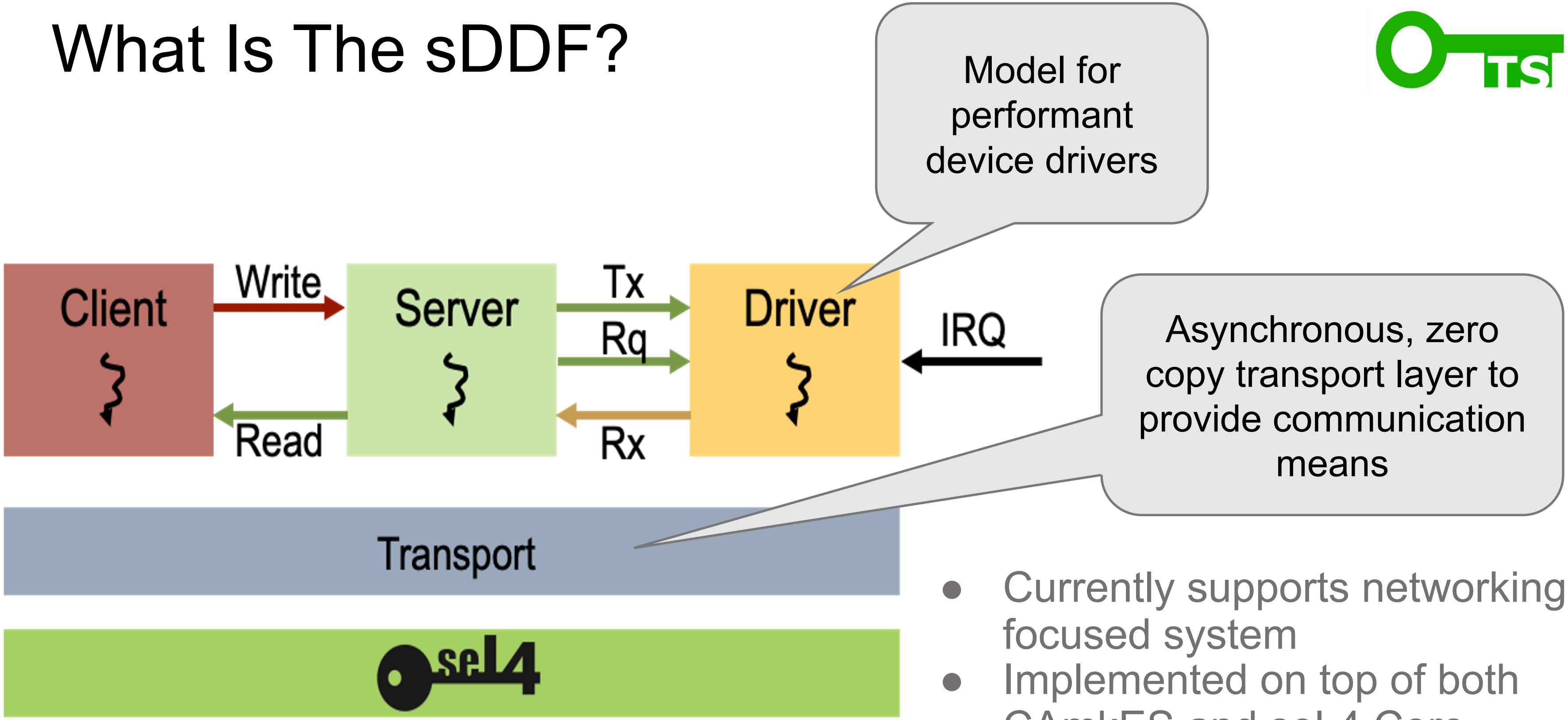
What Is The seL4 Device Driver Framework?

Framework to provide interfaces and protocol for writing performant device drivers as seL4 user level programs.





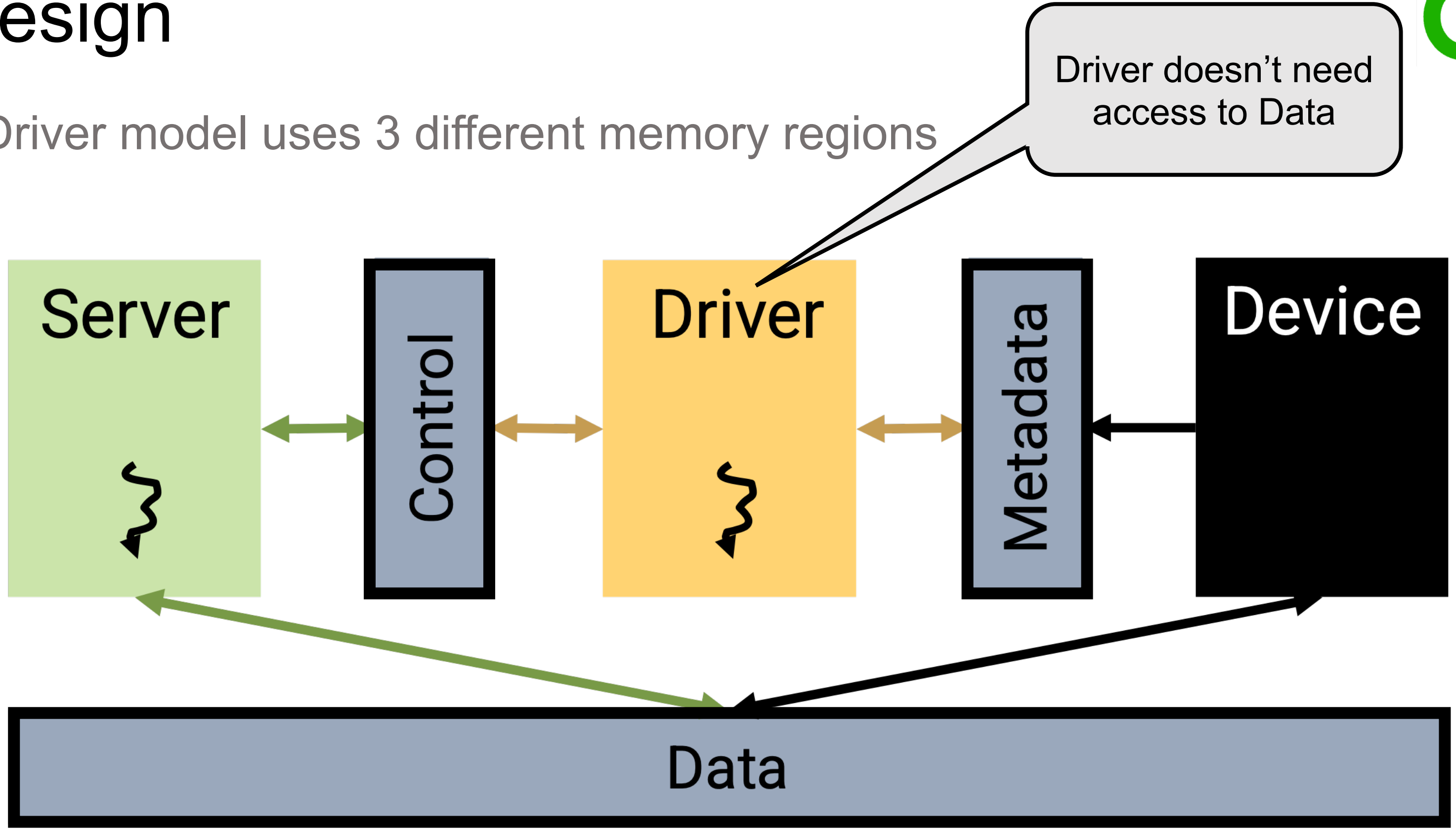
What Is The sDDF?



- Currently supports networking focused system
- Implemented on top of both CAmkES and seL4 Core Platform

Design

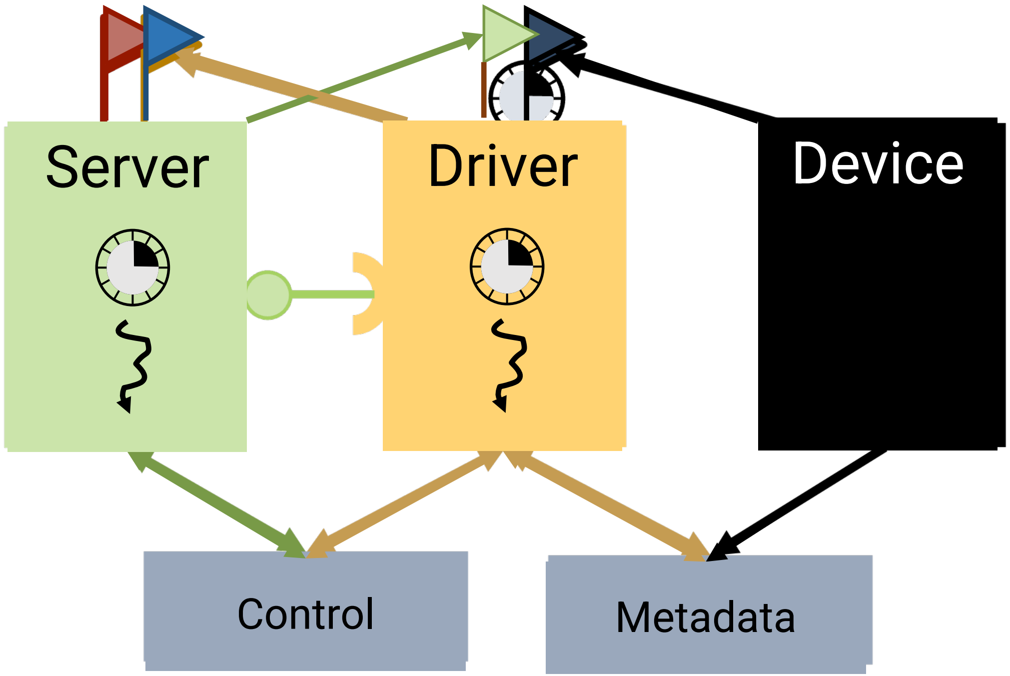
- Driver model uses 3 different memory regions



Driver Model

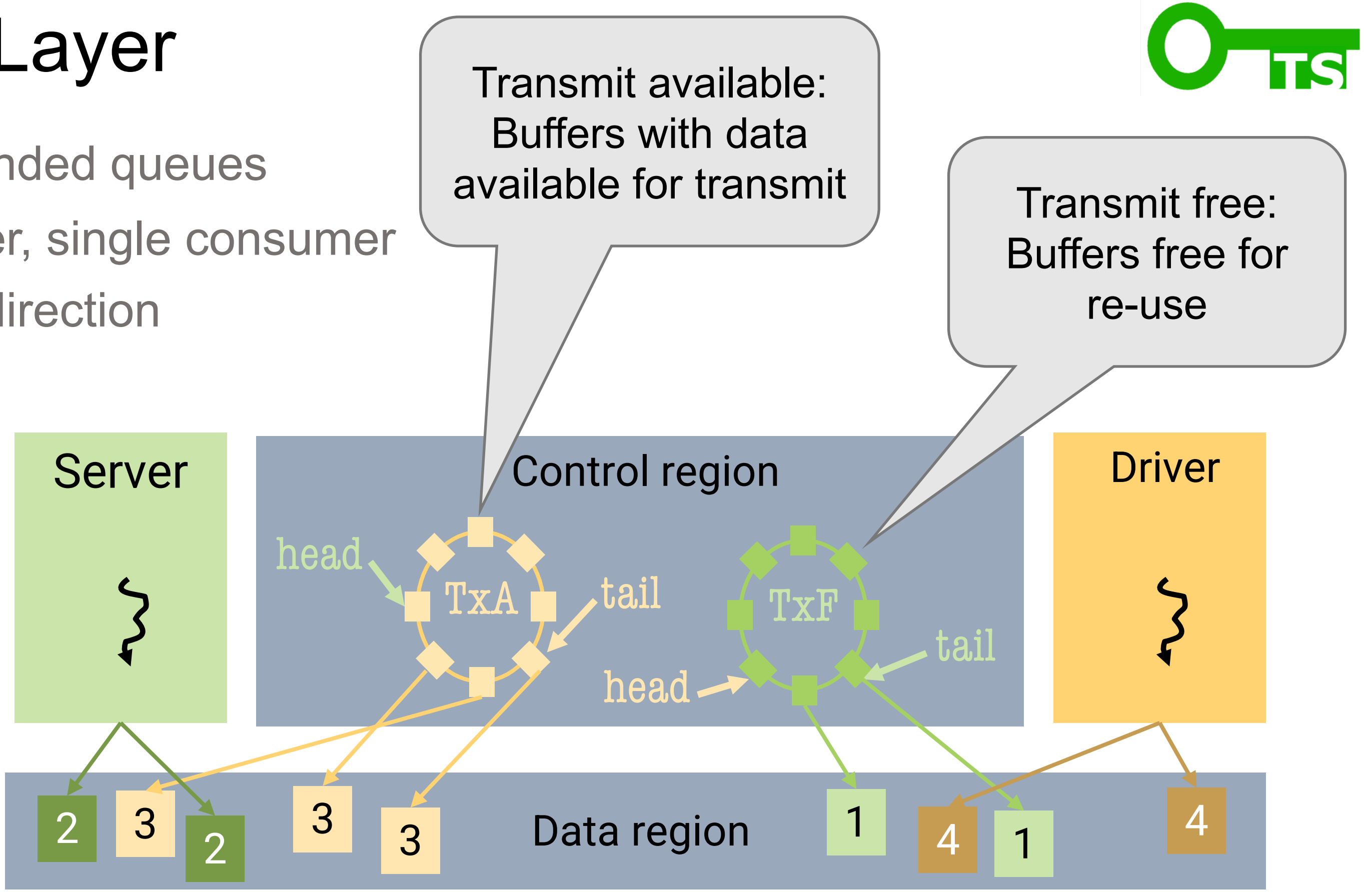
- Purely reactive
- Single threaded
- Simple

- Active
  … or passive?



Transport Layer

- Lock free, bounded queues
- Single producer, single consumer
- 2 queues per direction
- Zero copy



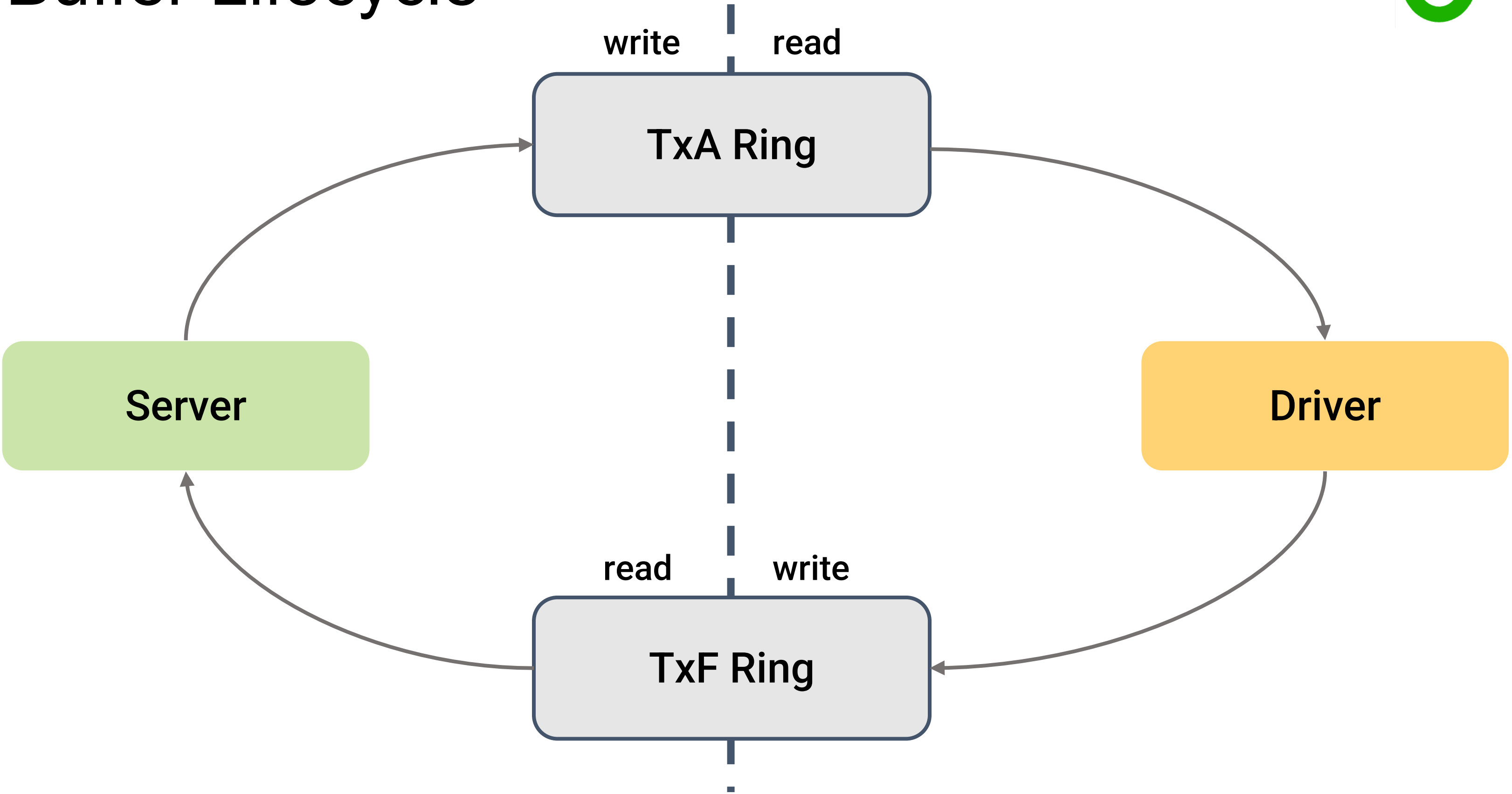
Transport Layer

```
struct buffer_descr {
  void *address;
  size_t length;
}

struct ring_buffer {
  uint32_t head;
  uint32_t tail;
  struct buffer_descr buffer[RING_SIZE];
}
```

```
void enqueue(struct buffer_descr *buffer,
              struct ring_buffer *ring) {
   assert(!full(ring));
   ring->buffer[ring->head % RING_SIZE] = *buffer;
   barrier();
   ring->head += 1;
}

struct buffer_desc* dequeue(struct ring_buffer *ring) {
  assert( !empty(ring) );
  struct buffer_descr *buf = \
           ring->buffer[ring->tail % RING_SIZE];
  barrier();
  ring->tail += 1;
}
```

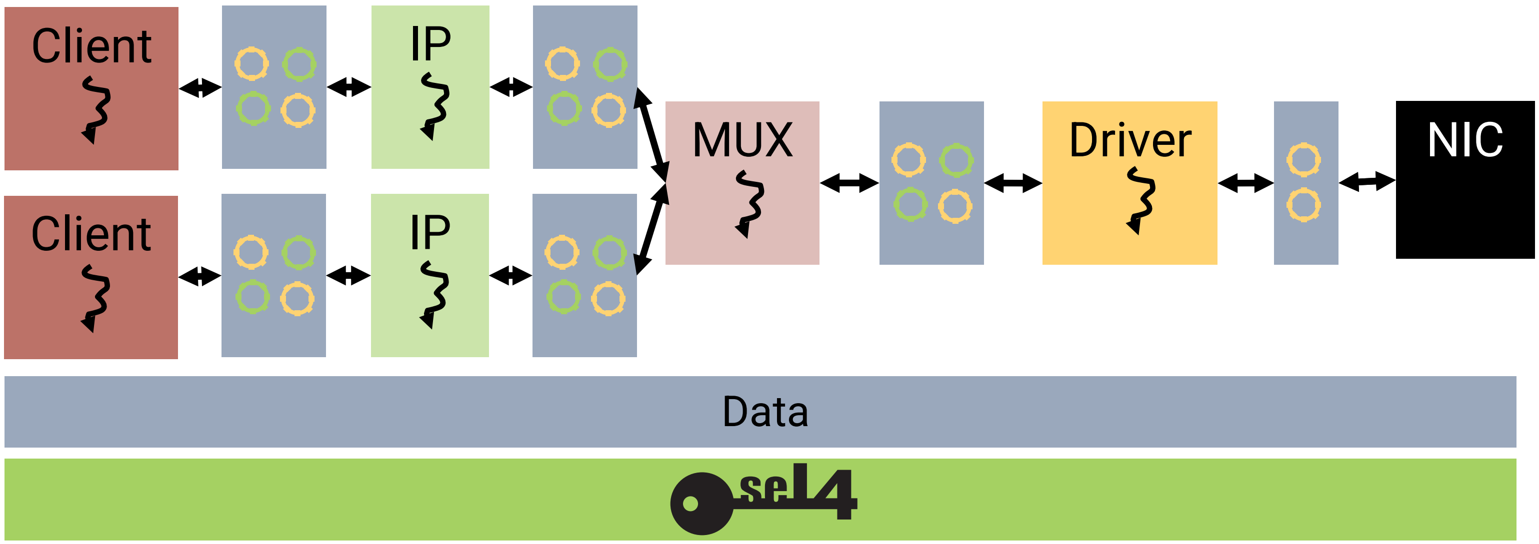
Barrier ensures no writes are re-ordered by the compiler or processor across this point

Buffer Lifecycle



Transport Architecture Scales

- Components can run on separate cores
- Only MUX and driver are trusted.



Driver Code: Active Model

```
main()
   init();
   notif = NULL;
   while(true)
      event = Signal_and_Wait(notif);
      notif = NULL;
      if (event & IRQ)
         handle_irq();
         notif = ack;
         continue;
      if (event & Transmit)
         /* process output */
         while (!full(HW_Tx) && buf = dequeue(TxA))
            enqueue(buf, HW_Tx);
         notif = server;
```

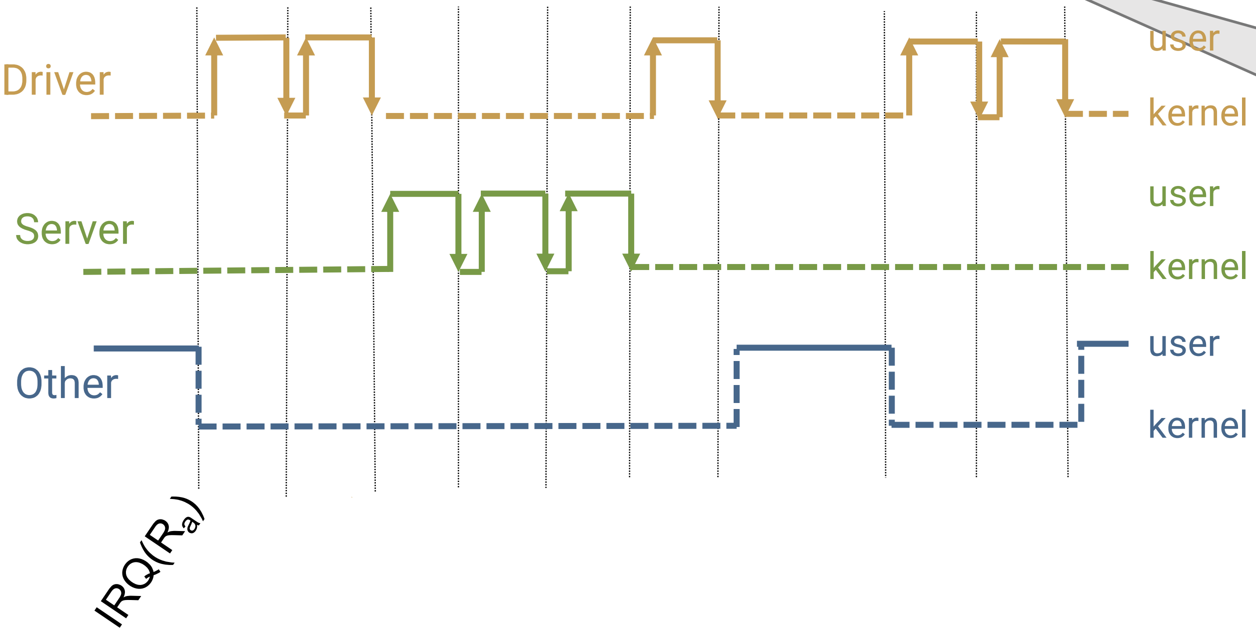
Combine syscalls for performance

Process multiple packets in one invocation

```
handle_irq()
   while (event = clear_hw_events())
      if (event & Tc)
         while (!full(TxF) && buf = dequeue(HW_Tx))
            /* return free Tx buffers to server */
            enqueue(buf, TxF);
      if (event & Ra)
         while (!full(RxA) && buf = dequeue(HW_Rx))
            /* process input */
            enqueue(buf, RxA);
         Signal(server);
         while (!full(HW_Rx) && buf = dequeue(RxF))
            /* return free Rx buffers */
            enqueue(buf, HW_Rx);
```

Can process multiple IRQ events in one invocation

Kernel Entries: Active Model



6 more than a monolithic kernel requires.

Compare system call cost vs packet processing cost.

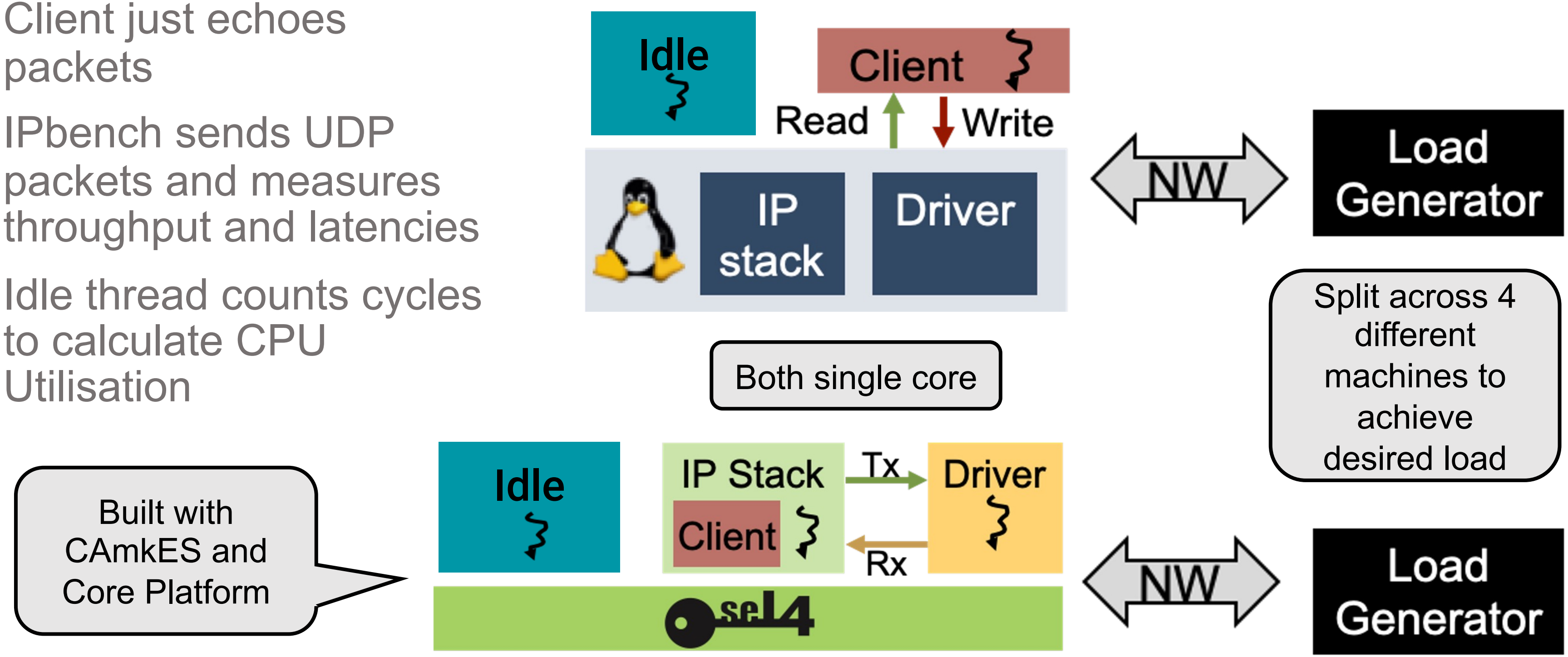
Rate Limiting

- Driver runs at highest priority to minimise round trip times
- Regardless of active/passive model, the protocols for transmit are synchronous.
- Requires limiting CPU time by configuring budgets and periods of scheduling contexts used by higher priority components.
  And/Or limiting the queue size.

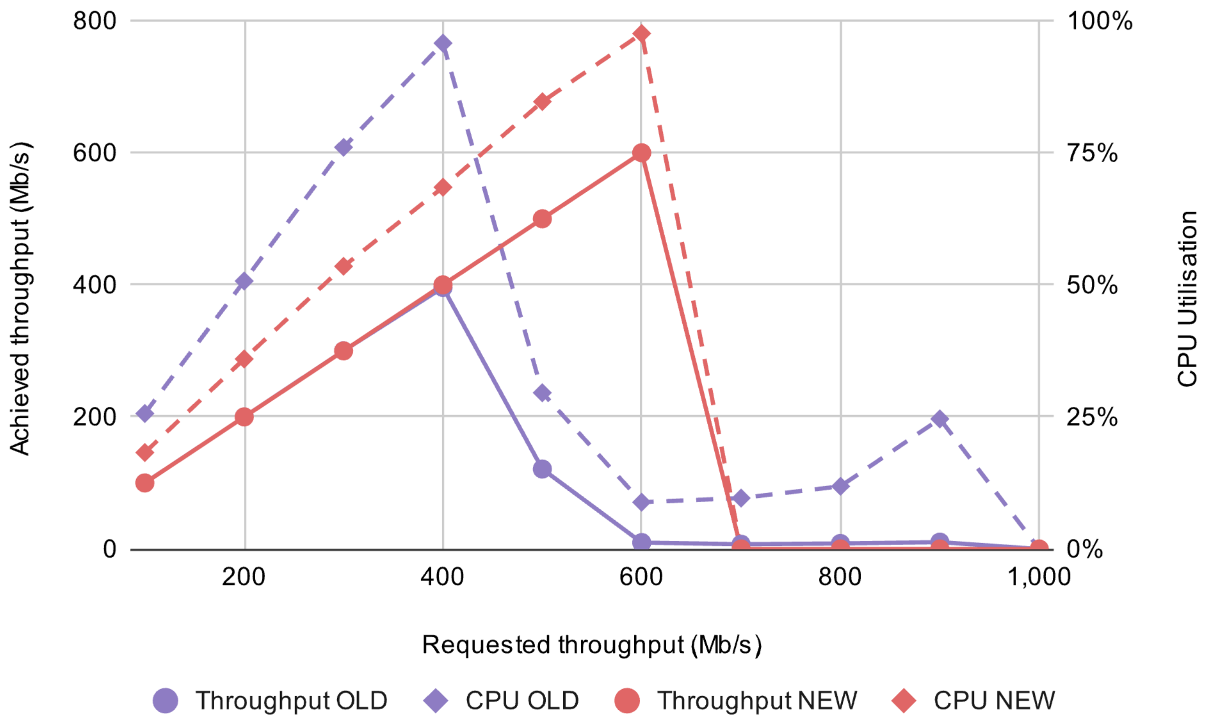
Performance

Set Up

- Client just echoes packets
- IPbench sends UDP packets and measures throughput and latencies
- Idle thread counts cycles to calculate CPU Utilisation



Old Transport vs New: CAmkES Networking Performance

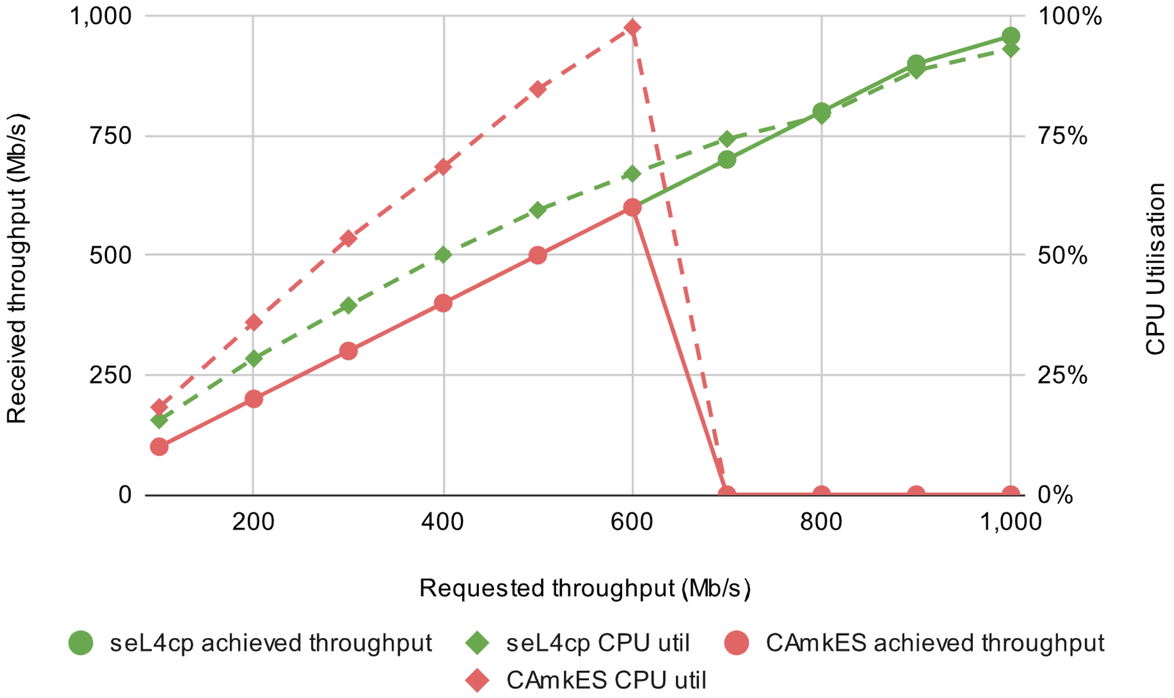


- Simpler transport
- Adjustment of priorities

- Showed 50% improvement!

- But could not combine system calls easily…

seL4 Core Platform vs CAmkES Networking Performance

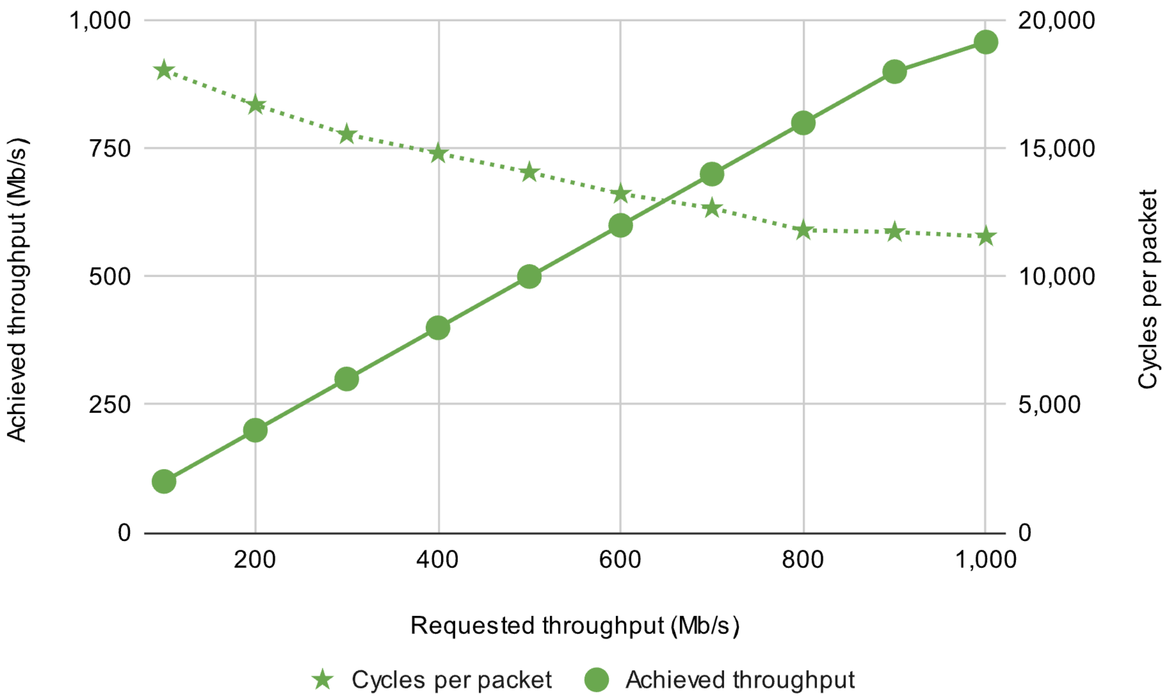


- Reduced kernel entries
- Simpler platform

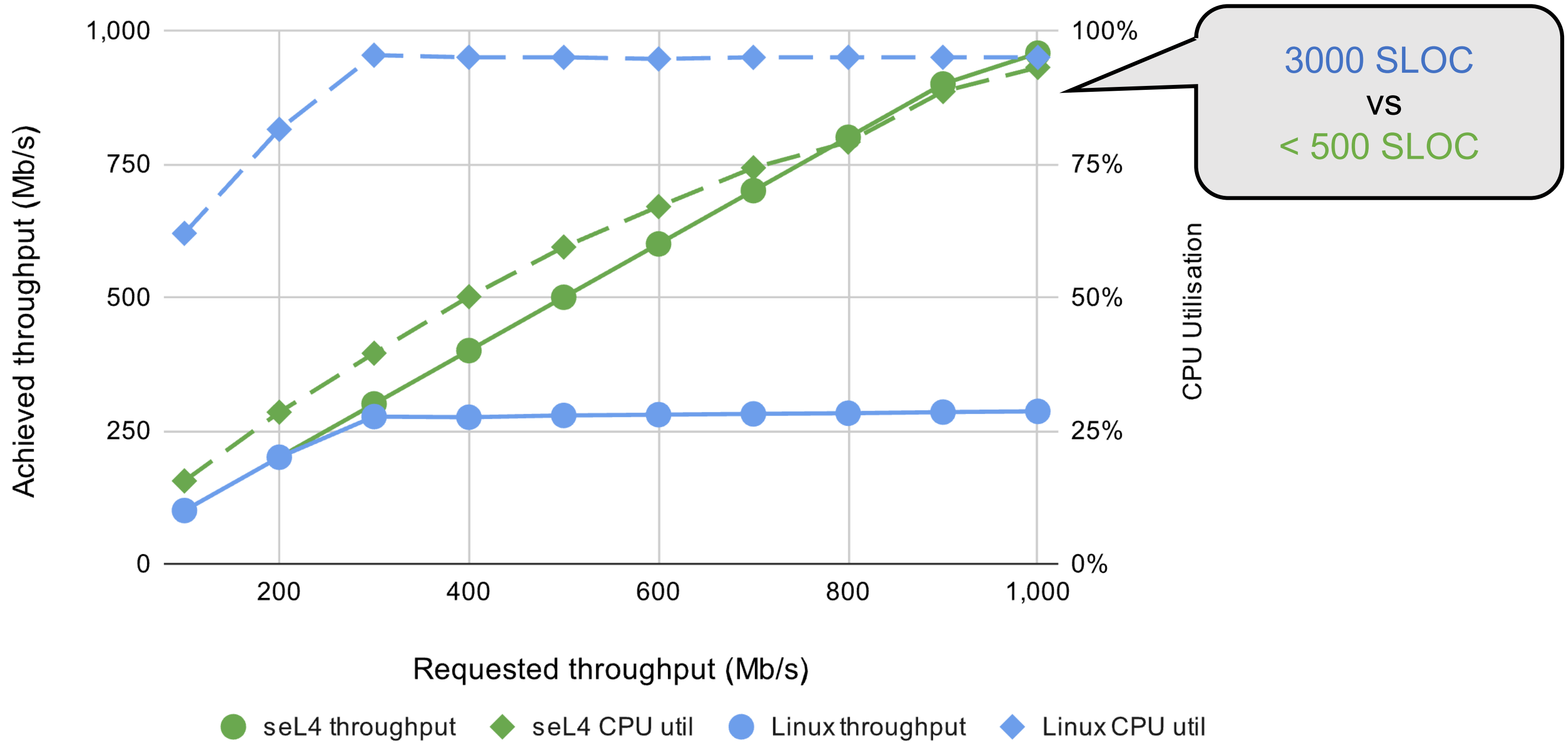
- Showed 70% improvement, 150% over old CAmkES!

- Limited drivers budget to remove performance collapse.

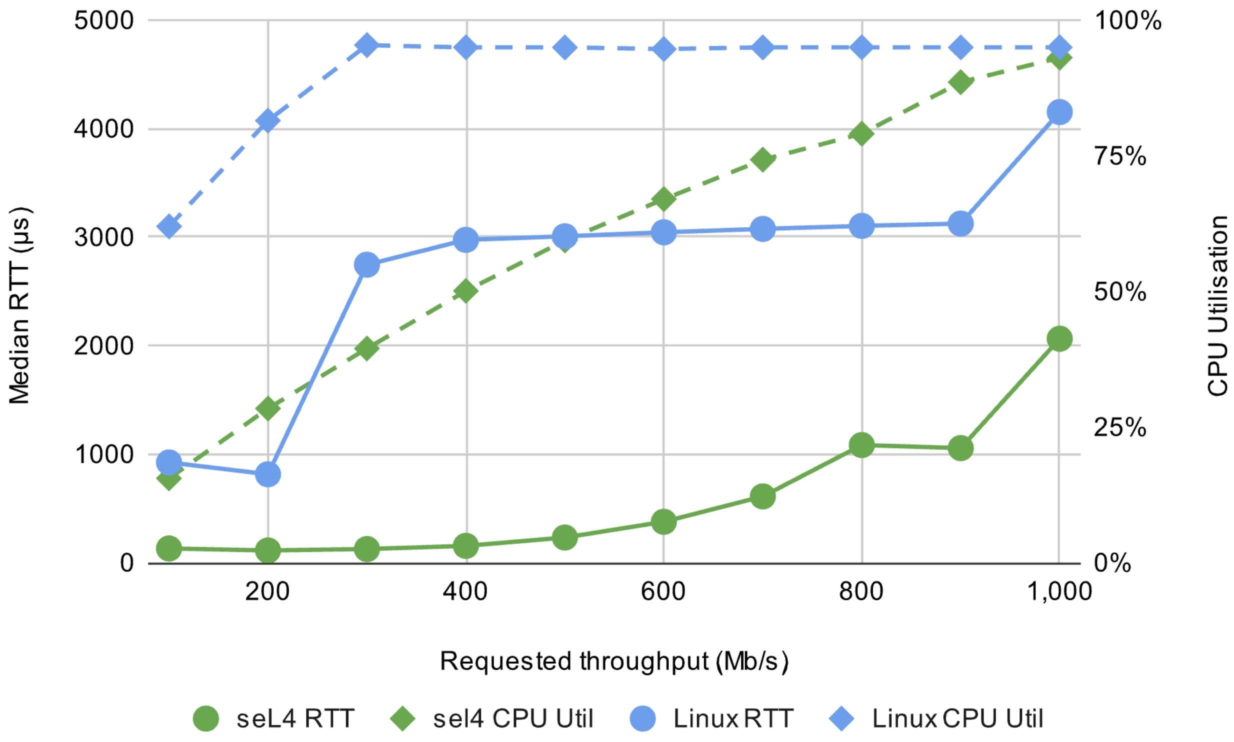
seL4 Packet Processing Cost



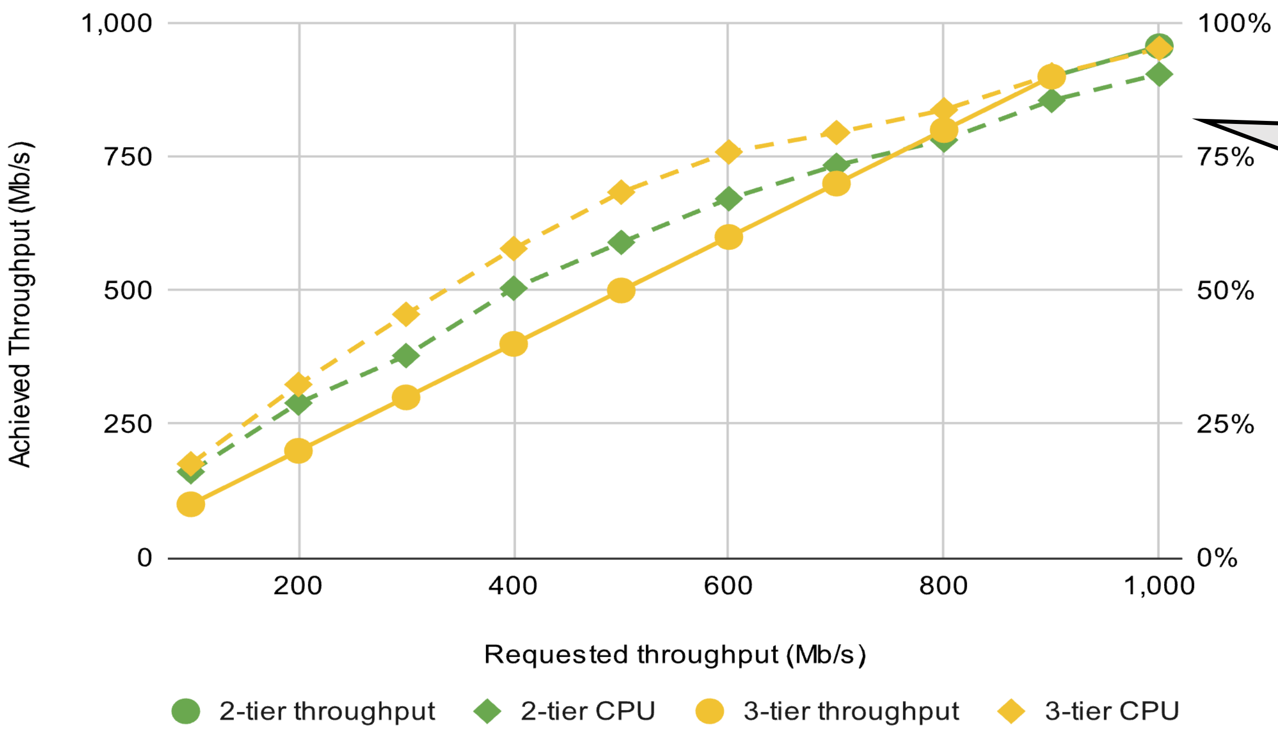
seL4 vs Linux Networking Performance



seL4 vs Linux RTT Comparison



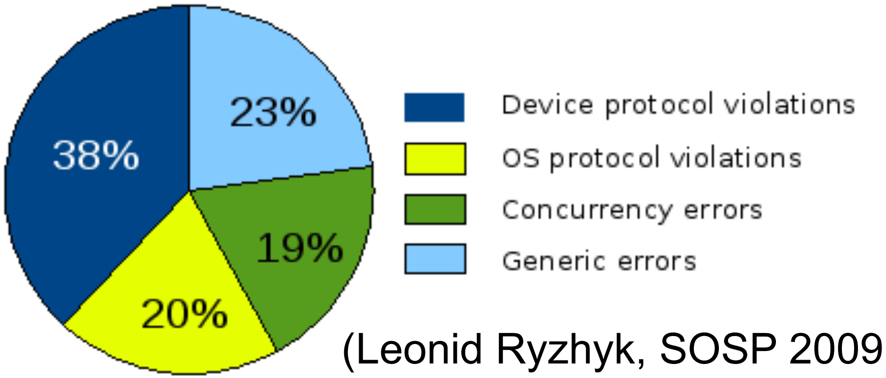
Cost Of A Module Crossing



Only 10% increase… demonstrates scalability!

Takeaways

- Significant performance boost: 150% improvement on old model
- Smaller latencies and higher throughput achieved over Linux
- Simple works!
  Eliminates concurrency bugs
  and will help verification effort.



(Leonid Ryzhyk, SOSP 2009)

Further Work

- Evaluate the passive driver model.
- Benchmark a more complex client.
- Investigate the optimal budgets and periods for different scenarios (eg. Asymmetric traffic).
- Evaluate what an optimised IP stack might look like.
- Design and build a multiplexor.
- Extend the sDDF to support other device classes.
- Evaluate multicore set up.

Code

- Current state of the code as implemented for seL4 Core Platform can be found here: https://github.com/lucypa/sDDF

- RFC: https://sel4.atlassian.net/browse/RFC-12

Questions?

Kernel Entries: Passive Model