School of Computer Science & Engineering

Trustworthy Systems Group

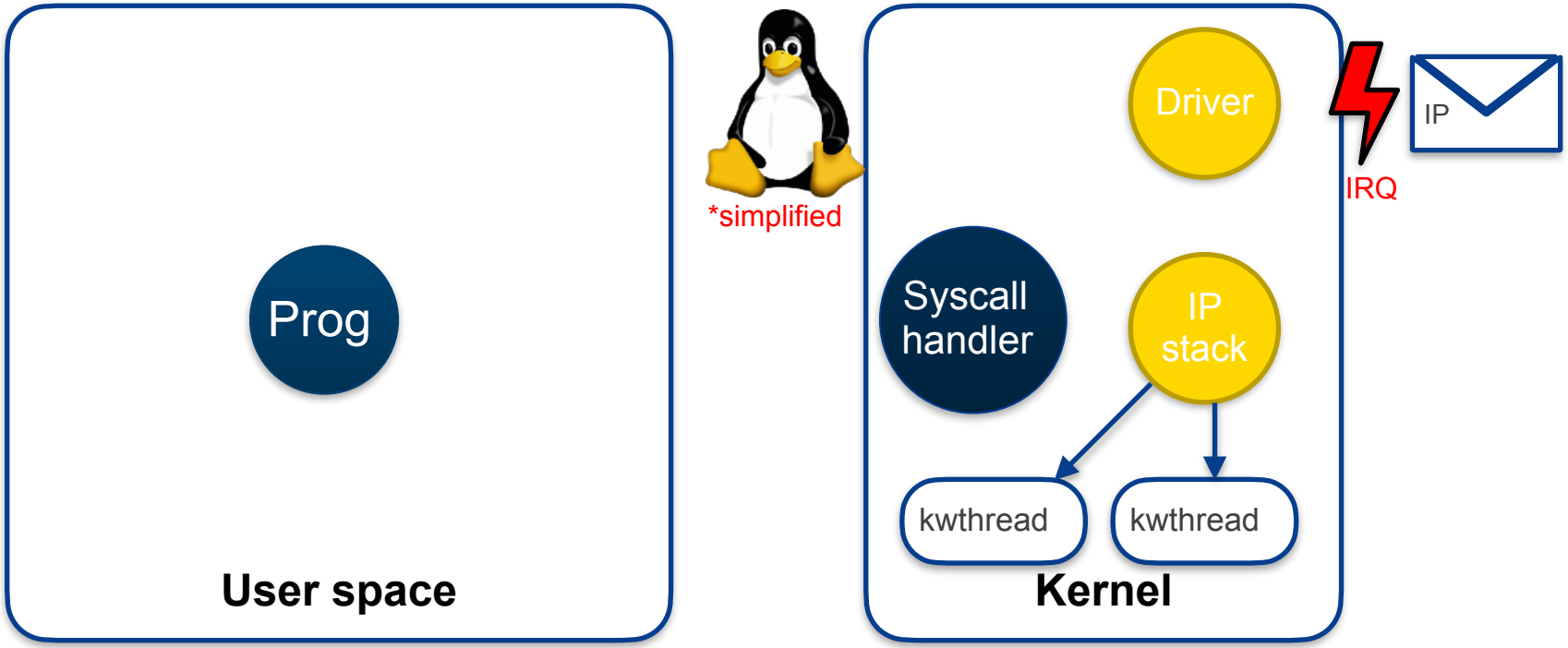
Performance analysis and modelling of Ethernet on LionsOS

seL4 Summit 2026

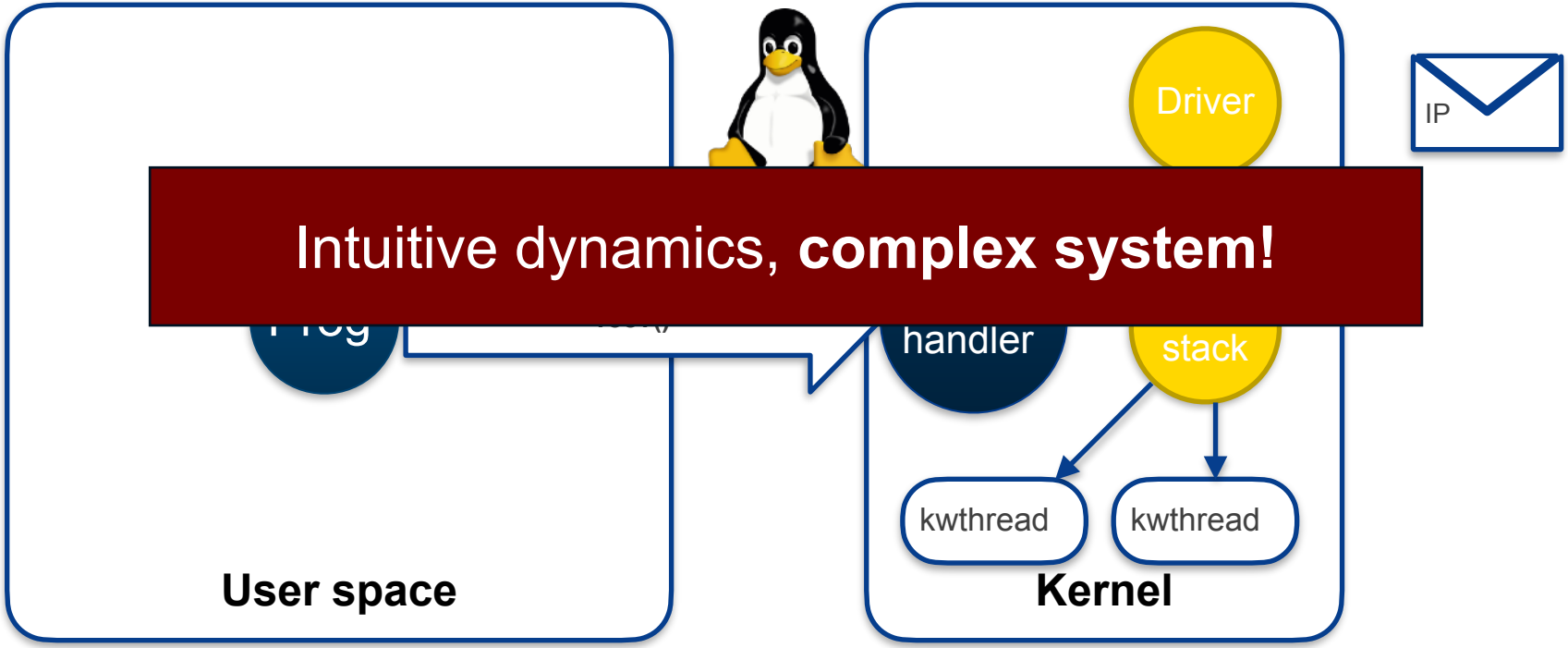
Lesley Rossouw

Normally driver-client dataflow is *relatively* predictable

Some in-kernel infrastructure does some work, delivers a packet to user space



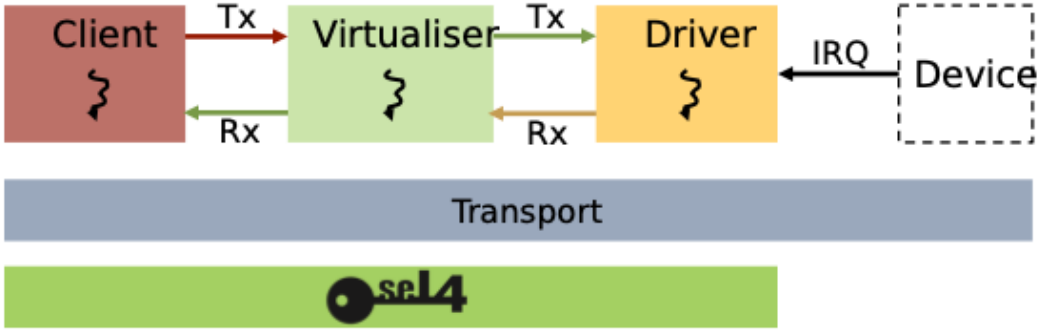
Some in-kernel infrastructure does some work, delivers a packet to user space



The seL4 Device Driver Framework

The seL4 Device Driver Framework (sDDF) is a high performance framework for implementing drivers.

- *"Radically simple" design*
- PDs interconnected with **queues** to move data
- **Notifications** to indicate queue changes
- seL4 Microkit - "abstract interface"



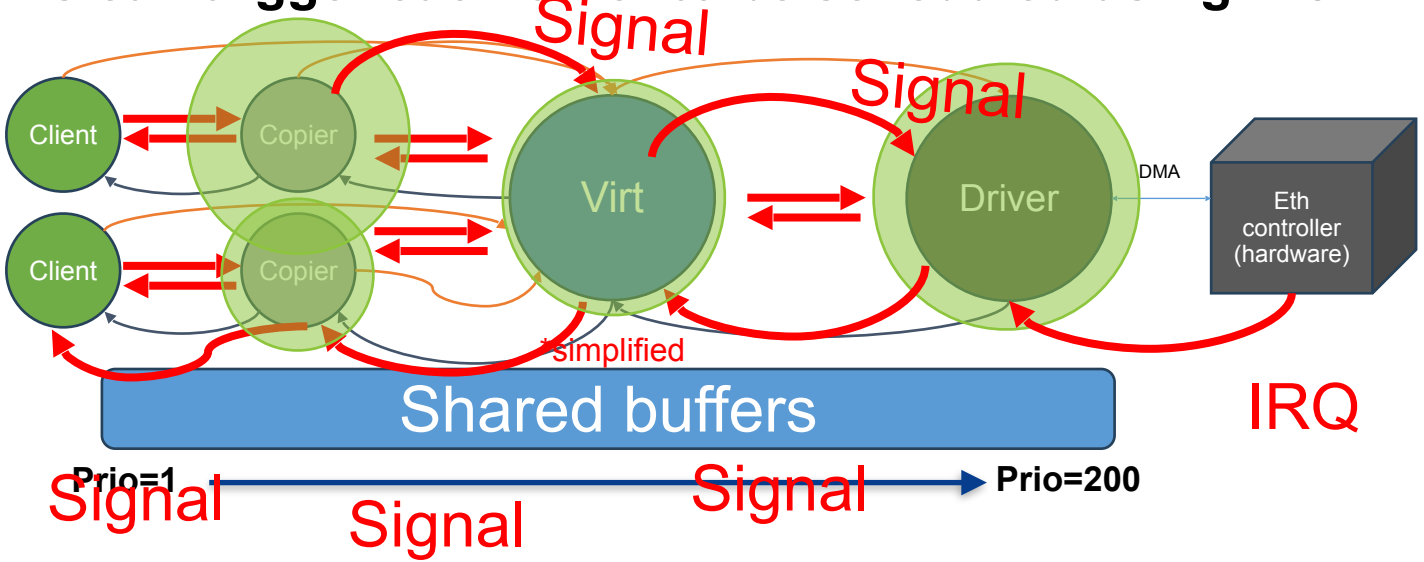
Provides drivers to **LionsOS**

**software interface perspective*

Dataflow is complex

Data moves from one PD to another when scheduled

… and PDs can trigger each other to be scheduled using IPC!



Scheduling priority: lowest on LHS, highest RHS

Dataflow is complex

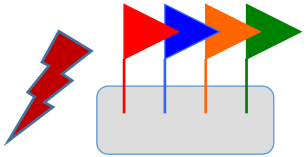
Data moves from one PD to another when scheduled

… and PDs can trigger each other to be scheduled using IPC!

Scheduling priority: lowest on LHS, highest RHS

Radically simple design

...complex dynamics!



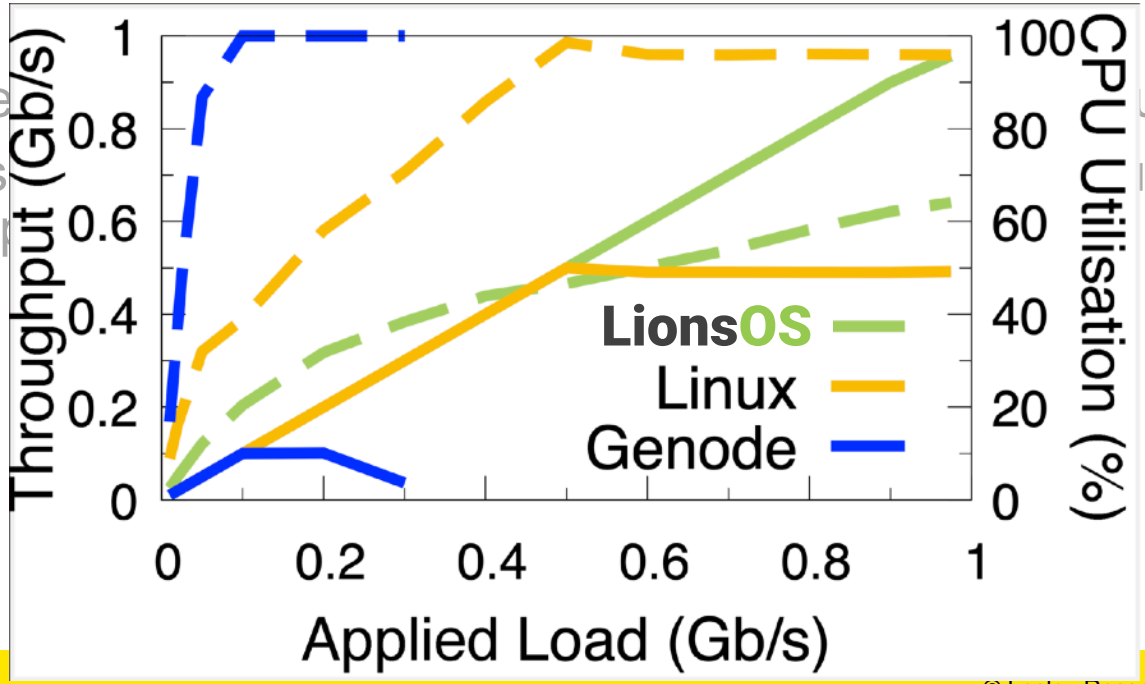
Observations about the sDDF

1. Performance can be characterised by **queue operations** and **IPC**
2. Scheduling has a **massive** impact on behaviour
3. Data flow is **oscillatory**
4. sDDF doesn't converge on an efficient operating point naturally
5. All of these properties will be true of any Microkit system using a similar separation of concerns

Most importantly: even without adapting to this, the sDDF performs excellently!

1. Perfo
2. Scheduling has a massive impact on behaviour
3. Data flow
4. sDDF doe urally
5. All of thes sing a similar sep

UDP Echo test, single-core i.MX8MM



Point of this talk

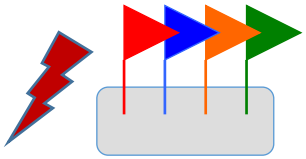
Explore the dynamics of queueing Microkit systems!

Observations

1. Performance can be characterised by **queue operations** and **IPC**
2. Scheduling has a **massive** impact on behaviour
3. Data flow is **oscillatory**

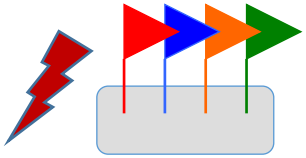
We need some tools to reason about this!

Queueing theory



Queues

- **What is a queue?**
  - Conceptually: some number of “servers” who provide a service to some number of “customers”
  - … with some **rate of customer arrivals** = λ
  - … and some **rate of customer processing** = μ
- If we have λ and μ, we can **predict behaviour of a whole queueing network!**



Queues (in human terms)

From the **perspective of one queue**, sDDF queues are:

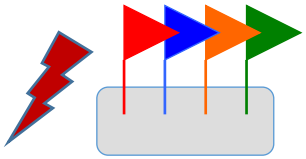
Kendall's notation

$$Q = M^{\lambda}/G^{\mu}(\lambda)/1$$

Markovian packet arrival process

Service time depends on how many buffers arrive at once

One server



Queues (in human terms)

From the **perspective of one queue**, sDDF queues are:

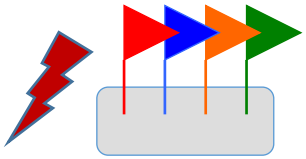
Kendall's notation

$$Q = M/G(\lambda)/1 \quad (\lambda, \mu)$$

Basically the simplest queue we can hope for!

rver

Service time depends on how many buffers arrive at once



Queueing networks

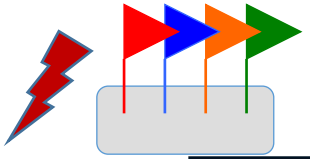
From the **perspective of the queueing network**, sDDF queues are:

Kendall's notation

$$Q = M/G(\lambda, S)/1$$

Extra consideration … **scheduling!**

Queues only serviced when PD is runnable and running!



Queueing networks

Scheduling depends on behaviour of programs!

Behaviour of programs depends on data entering the system!

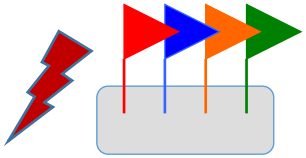
We can't know "true" μ, λ, values - depend on steady-state of system.
... **non-stationary queueing network**

But we can approximate the mean μ, λ with an execution trace!

We can’t easily collect full execution traces without affecting results however …

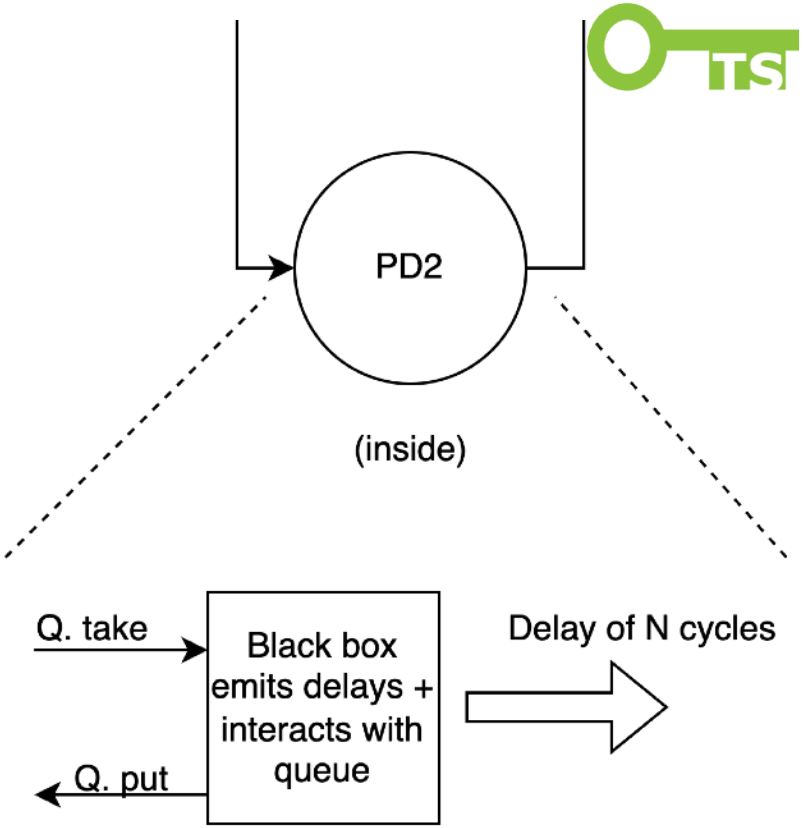
Simulation

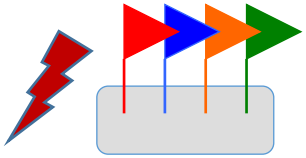
What if we throw out everything that doesn't affect scheduling or queue ops?



Simulation

- **Discrete event simulation**: don't simulate cycles, keep a queue of events instead
- We reduce PDs to be black boxes w/ **2 event types**
  - Queue operations + IPC
  - Delays modelling program work
- **CPU cores** and **devices** run in "parallel" - each has own event queue.

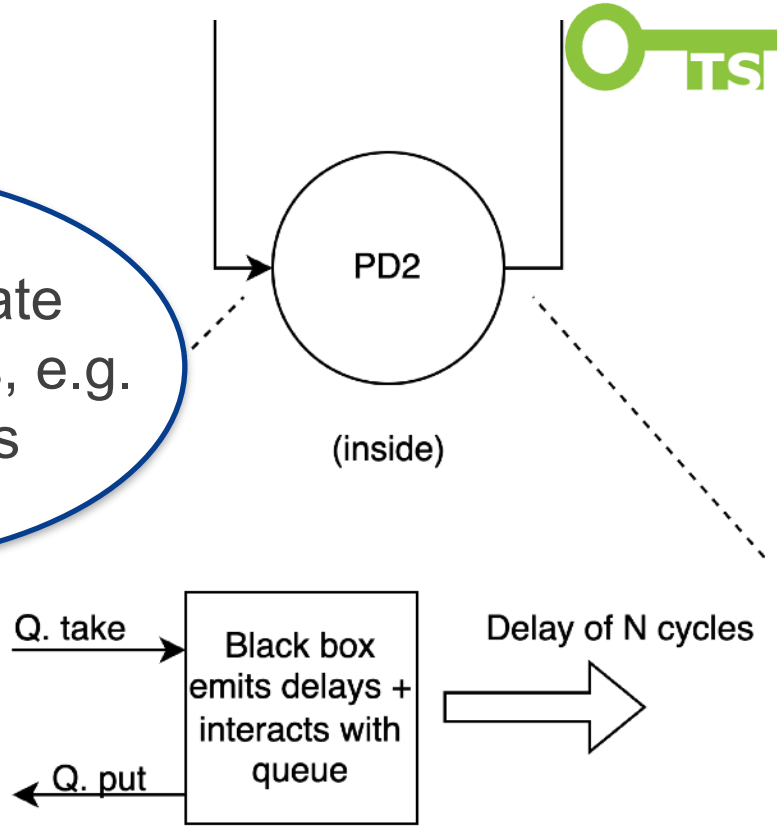


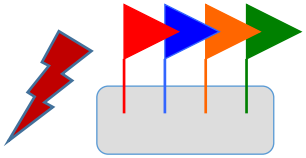


Simulation

- **Discrete event simulation**: don't simulate cycles, keep a… events instead
- We reduce PDs to … **event types**
  - Queue operations + …
  - Delays modelling … ogram work
- **CPU cores** and **devices** run in "parallel" - each has own event queue.

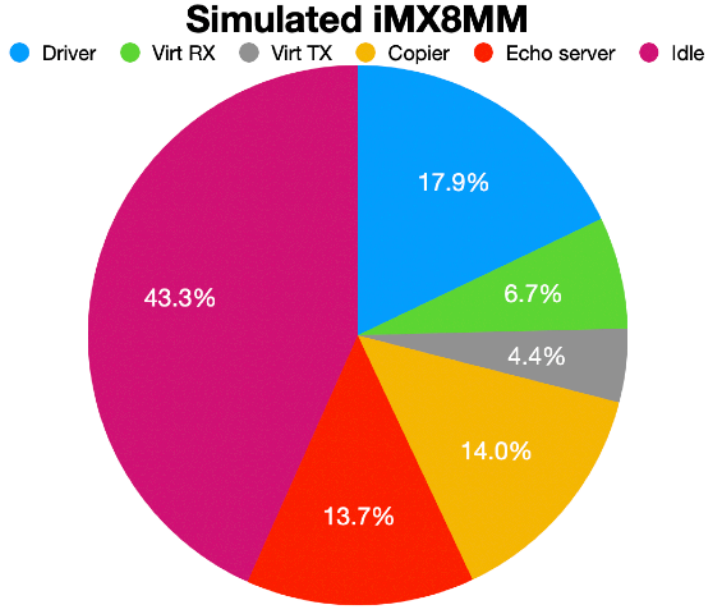
Devices generate random variations, e.g. traffic patterns

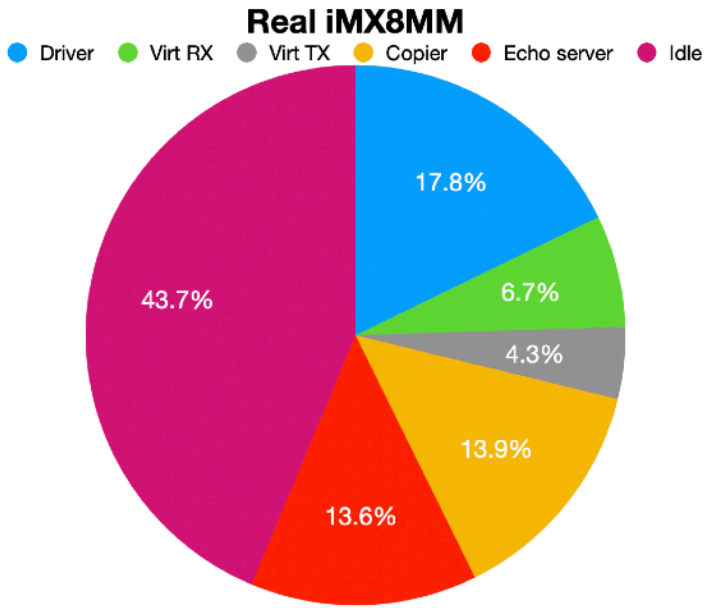


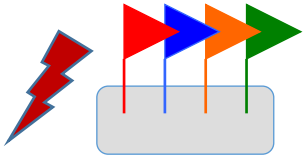


Simulation engine

- Uses seL4bench output to approximate system performance
- Program delays approximated from compiled code



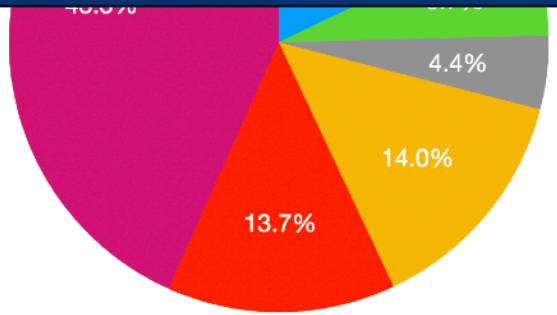


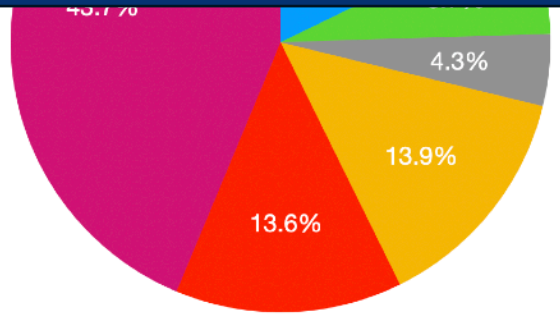


Simulation engine

- Uses seL4bench output to approximate system performance
- Program delays approximated from compiled code

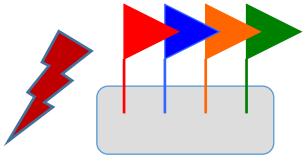
We can predict performance relatively accurately with queueing ops (inc. scheduling) **alone!**



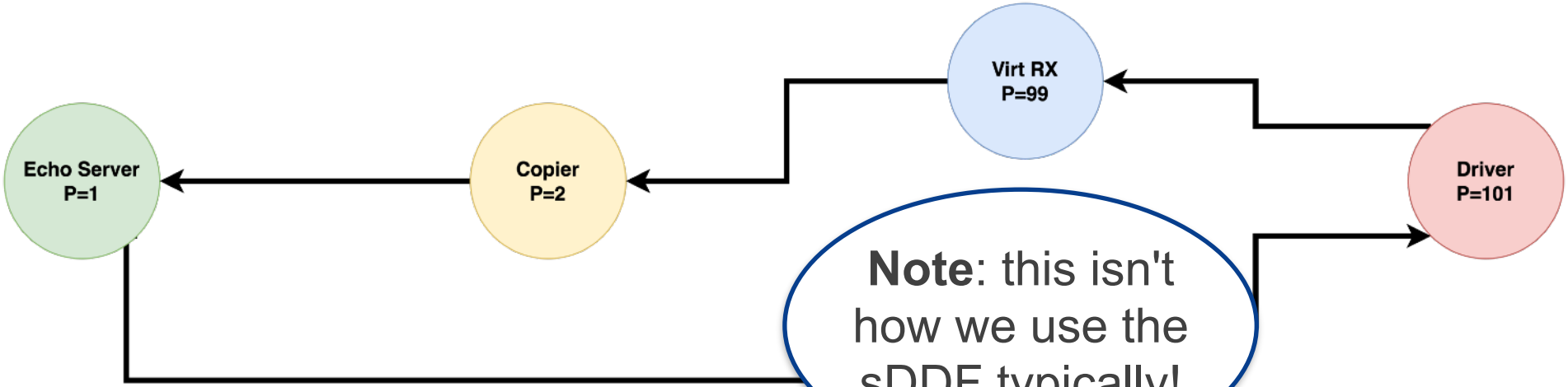


Observations

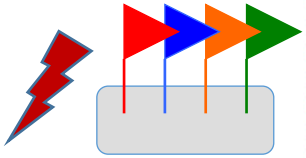
1. Performance can be characterised by queue operations and IPC
2. Scheduling has a **massive** impact on behaviour
3. Data flow is **oscillatory**



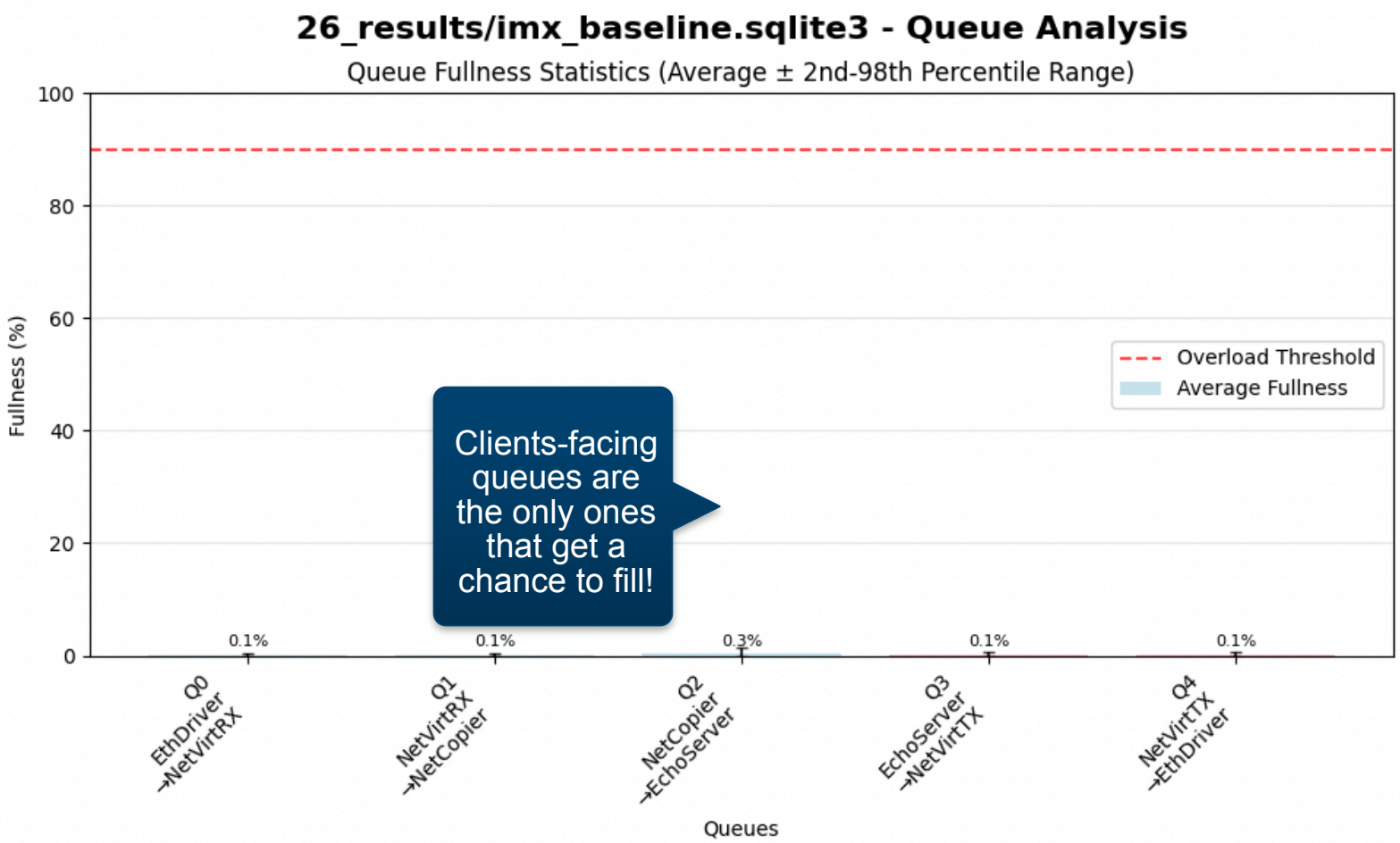
Demo system

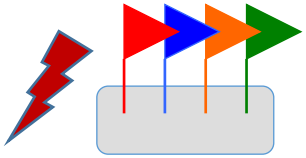


Simulating: single-core iMX8[illegible] mini, 1472 byte packets, echo client, **unlimited periods/budgets**

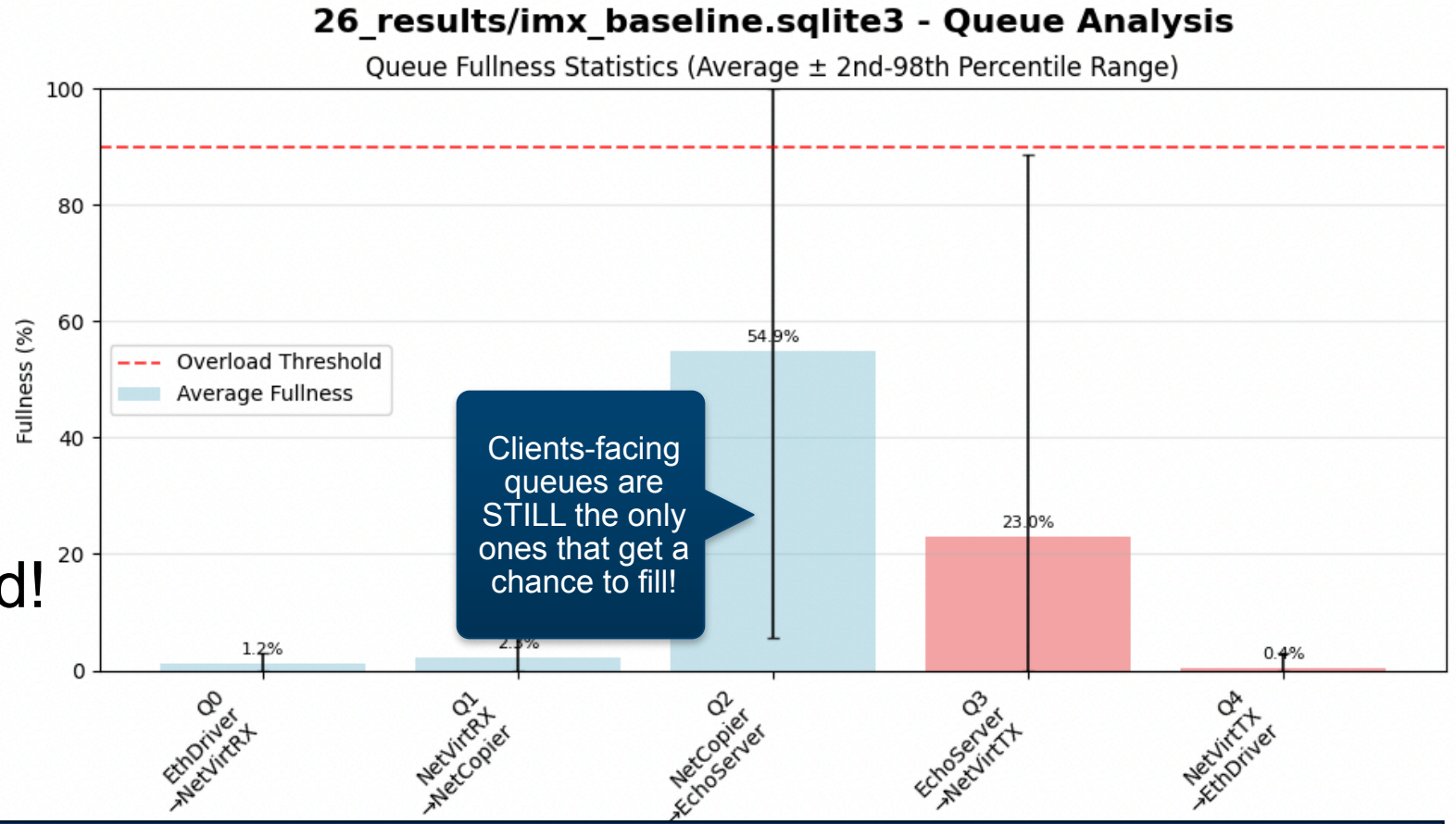


Queues don't do much queueing!

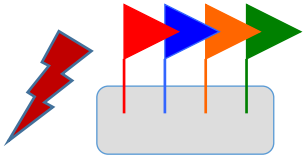




... unless we're overloaded!

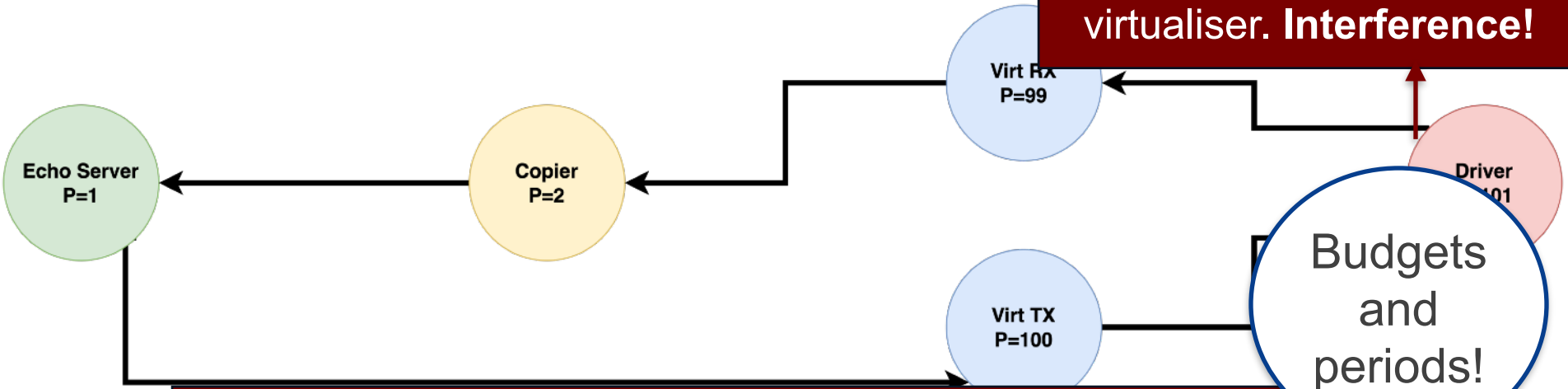


Simulating: single-core iMX8M mini, ~~1472~~ 200 byte packets



... unless we're overloaded!

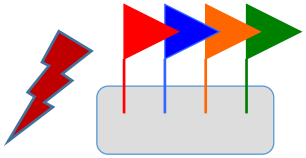
Driver is constantly awoken by IRQs, notifications from virtualiser. **Interference!**



Higher priority PDs always get a chance to run first, **so they rarely fill their queues AND starve clients unless constrained!**

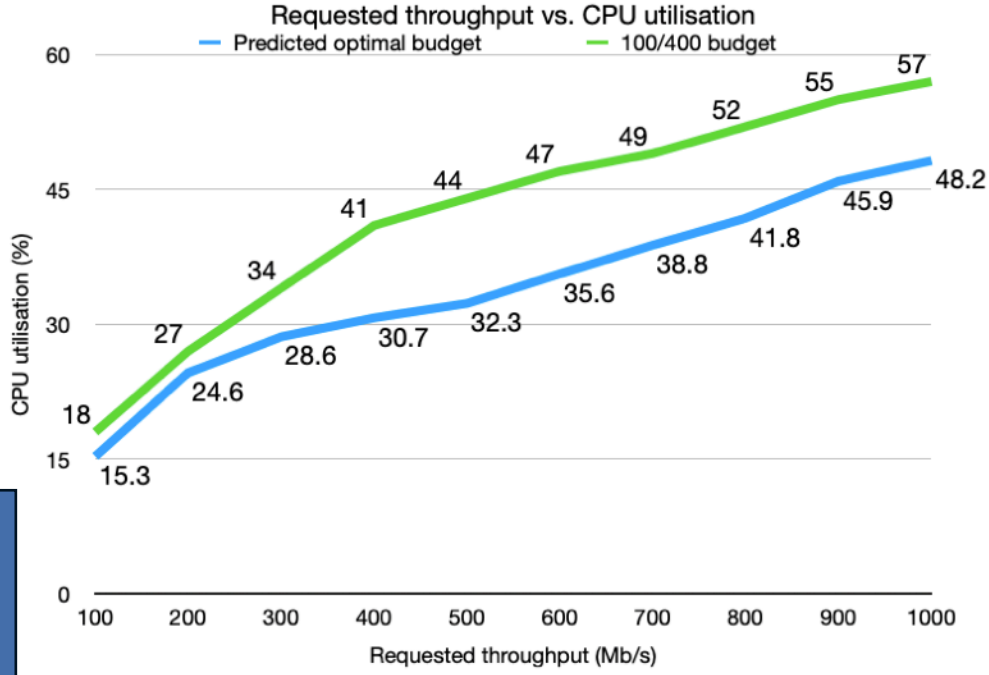
... but how should we budget things optimally?

Simulator approximates queueing constants, we can use that!



Optimal drtiver budgeting

- Let's use $400\mu s$ period, reduce latency
- $\text{budget} \approx \frac{\text{period}}{T_{\text{packet}}} \times \mu_{\text{avg}} + \text{T}_{\text{packet}} = 63\mu s$
- Testing on iMX8MM EVK, corresponding sim model



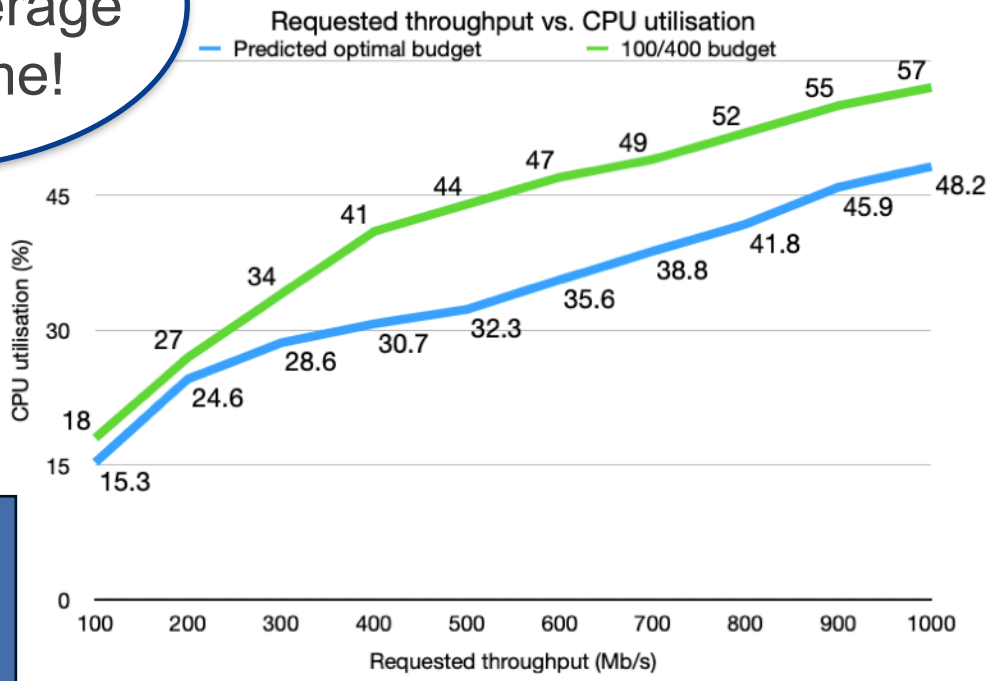
Free performance!
Same work done, more efficiently!
(… for an expected sustained load)

udgeting

"Driver only wakes long enough to enqueue packets that arrived while sleeping"

Simulator
s us average
service time!

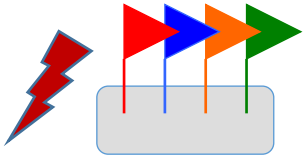
- Let's use $400\mu s$ to reduce latency
- $\text{budget} \approx \frac{\text{period}}{T_{\text{packet}}} \times \mu_{\text{avg}} + \text{T}_{\text{packet}} = 63\mu s$
- Testing on iMX8MM EVK, corresponding sim model



Free performance!
Same work done, more efficiently!
(… for an expected sustained load)

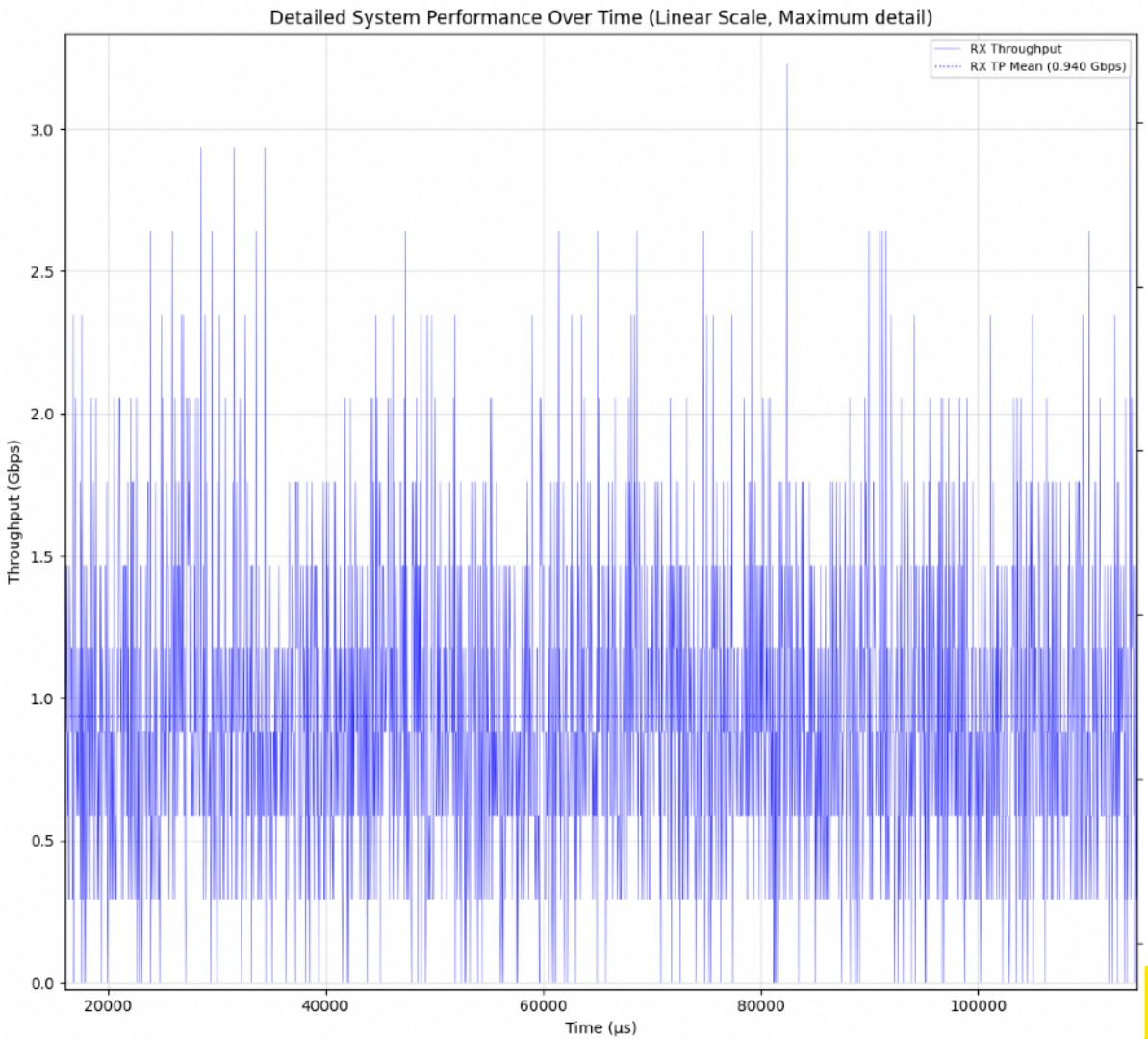
Observations

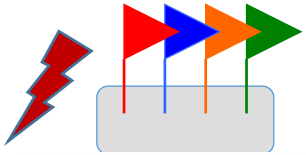
1. Performance can be characterised by queue operations and IPC
2. Scheduling has a **massive** impact on behaviour
3. Data flow is **oscillatory**



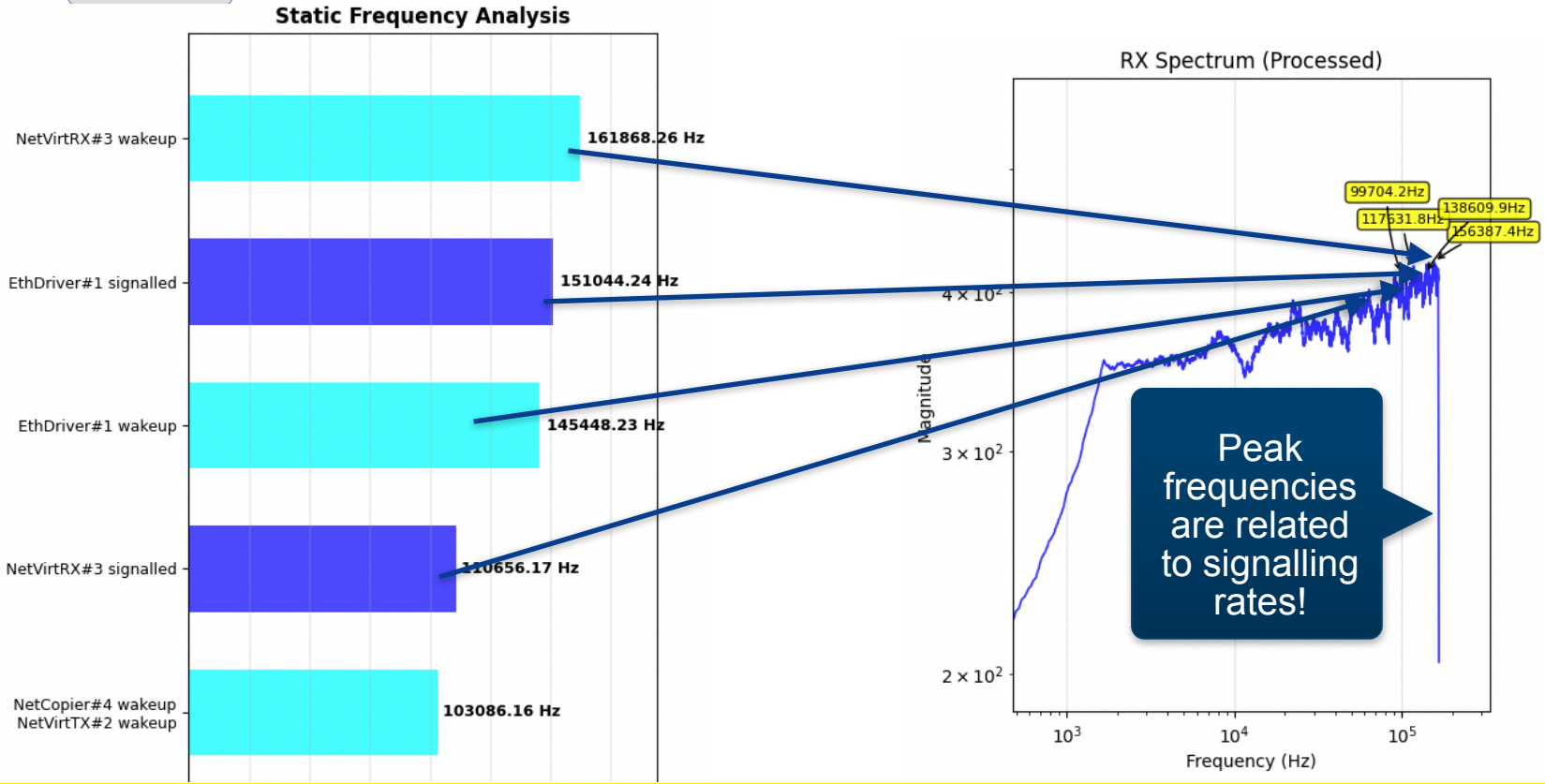
Client sees a **noisy** stream arriving

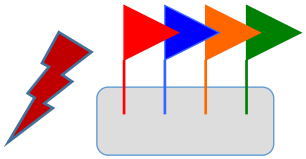
Shown: RX throughput seen by client vs. time



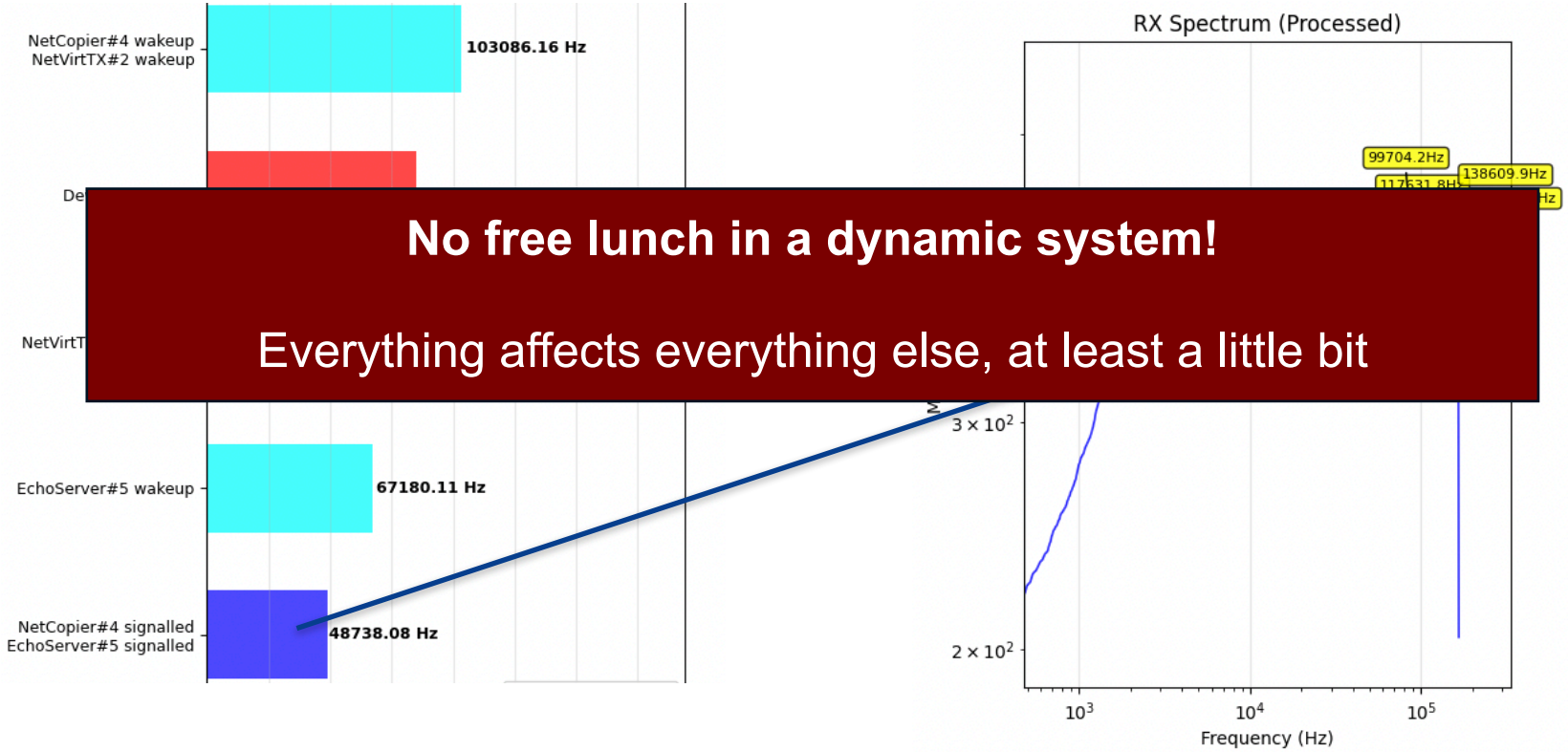


Client sees a **noisy** stream arriving



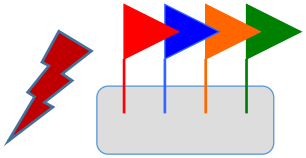


Client sees a **noisy** stream arriving



No free lunch in a dynamic system!

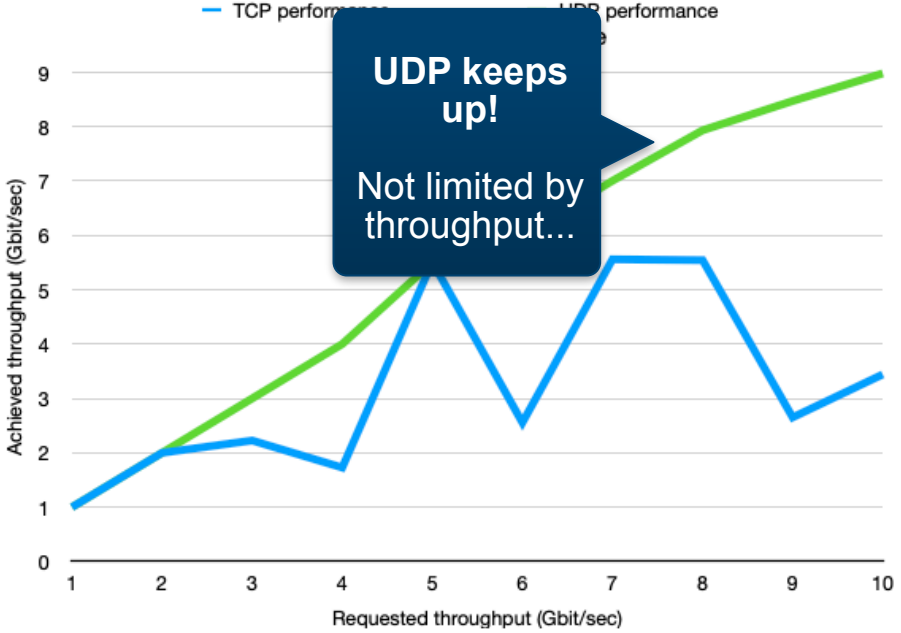
Everything affects everything else, at least a little bit

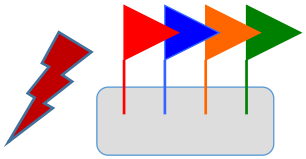


Resonance

Case: terrible TCP performance, good UDP performance

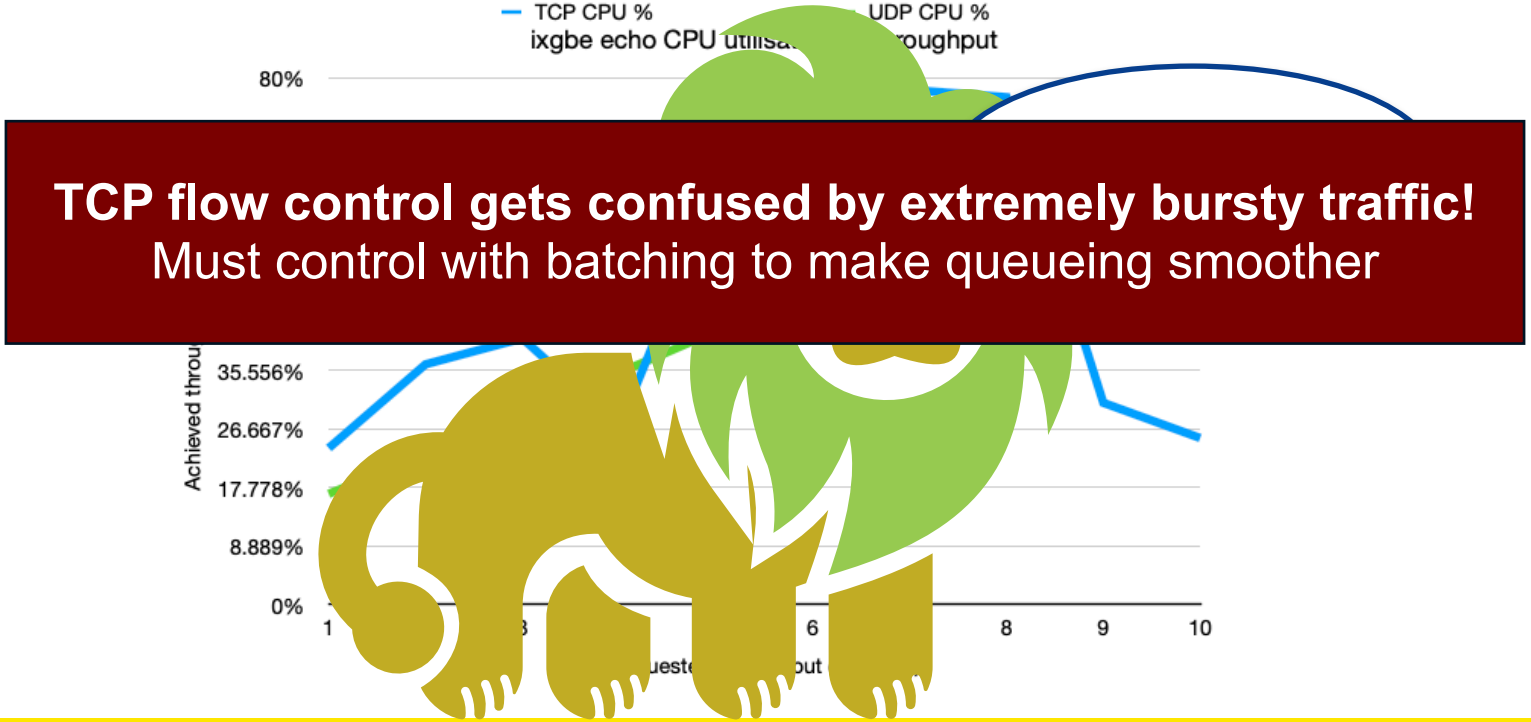
- **10GBe networking tests**
- 9-10Gbit/sec UDP
- … but 3-5Gbit/sec on TCP



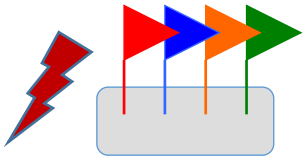


Resonance

Case: terrible TCP performance, good UDP performance

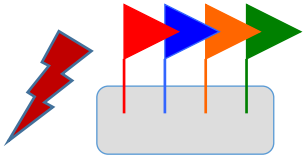


TCP flow control gets confused by extremely bursty traffic!
Must control with batching to make queueing smoother



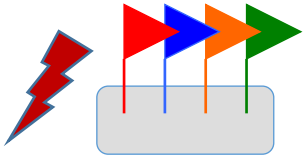
Lessons learned

- Main lesson: **Microkit systems are dynamic systems!**
- **Think like a non-software engineer!**
- Simple program behaviour ≠ simple interactions between programs
- **Really think about the impact of scheduling!**
  - Reason about your systems in terms of steady states
  - ... and make sure your system will actually converge on an acceptable one



Lessons learned

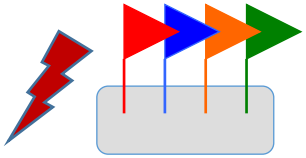
- **Number of buffers moved per context switch should be as high as possible!**
  - Maximises work per syscall
  - Minimises chances for interference
- CPU time to spare ≠ "not overloaded"
- **Analytical models of our systems are crucial to understand and optimise them!**



Future

- Wouldn't it be nice if our queues could adapt to a new steady state, rather than being manually tuned for just one?
- ... Working on 100Gbit/sec ethernet soon, **a fixed tuning isn't good enough...**

Something cool coming soon
(hopefully!)



More info?

https://trustworthy.systems/publications/theses_public/25/Rossouw%3Abe.pdf

```
  @masterthesis{Rossouw:be,
    address           = {Sydney, Australia},
    author            = {Lesley Rossouw},
    keywords          = {LionsOS, performance,
networking},
    month             = sep,
    paperUrl          = {https://trustworthy.systems/
publications/theses_public/25/Rossouw%3Abe.pdf},
    school            = {School of Computer Science
and Engineering},
    title             = {{sDDF} Queueing Performance},
    type              = {{BE} Thesis},
    year              = {2025}
  }
```

Questions?

More info

www.trustworthy.systems