School of Computer Science & Engineering

Trustworthy Systems Group

SMT-based Deductive Verification of Device Drivers using the Pancake-to-Viper Transpiler

Samuel Tyler, Alessandro Legnani, Junming Zhao, Zhewen Shen, Rihui Wu, Miki Tanaka, Gernot Heiser

samuel.tyler@unsw.edu.au

My Claim

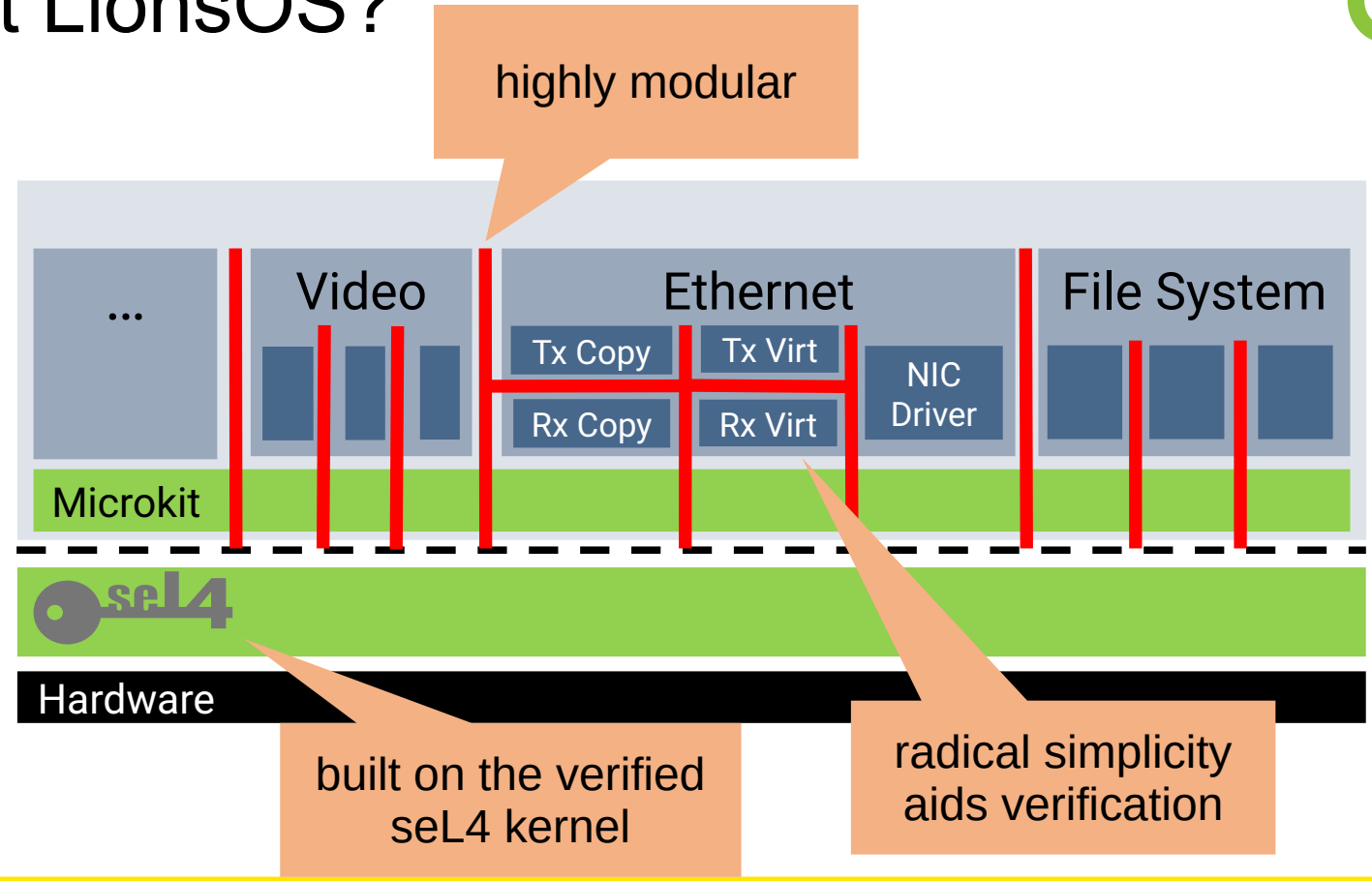
Formal verification of real-world device drivers is

necessary and **attainable**

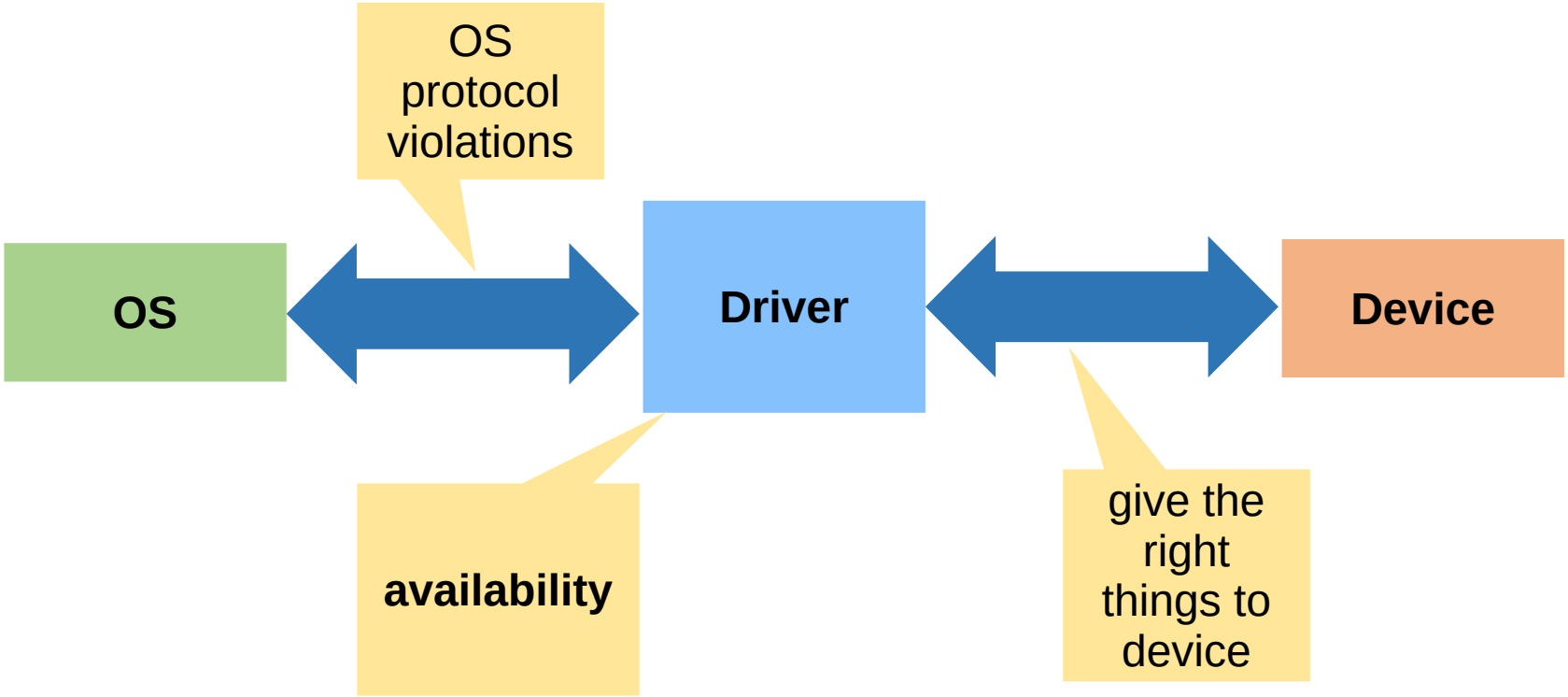
seemingly at scale

We have verified a **real-world Ethernet NIC device driver** for LionsOS using **automated** SMT-based techniques.

But LionsOS?



What goes wrong?



Systems Programmers?

```
87  typedef struct reqbk {
230     /* Add virtqueues */
231     size_t desc_off = 0;
232     size_t avail_off = ALIGN(desc_off + (16 * VIRTQ_QUEUE_SIZE), 2);
233     size_t used_off = ALIGN(avail_off + (6 + 2 * VIRTQ_QUEUE_SIZE), 4);
234     size_t size = used_off + (6 + 8 * VIRTQ_QUEUE_SIZE);
235     assert(size <= GPU_VIRTIO_METADATA_REGION_SIZE);
236
237     virtq.num = VIRTQ_QUEUE_SIZE;
238     virtq.desc = (struct virtq_desc *)(virtio_metadata + desc_off);
239     virtq.avail = (struct virtq_avail *)(virtio_metadata + avail_off);
240     virtq.used = (struct virtq_used *)(virtio_metadata + used_off);
241
242     assert(regs->QueueNumMax >= VIRTQ_QUEUE_SIZE);
243     regs->QueueSel = VIRTIO_GPU_CONTROL_QUEUE;
244     regs->QueueNum = VIRTQ_QUEUE_SIZE;
245     regs->QueueDescLow = (virtio_metadata_paddr + desc_off) & 0xFFFFFFFFUL;
246     regs->QueueDescHigh = (virtio_metadata_paddr + desc_off) >> 32;
247     regs->QueueDriverLow = (virtio_metadata_paddr + avail_off) & 0xFFFFFFFFUL
248     regs->QueueDriverHigh = (virtio_metadata_paddr + avail_off) >> 32;
249     regs->QueueDeviceLow = (virtio_metadata_paddr + used_off) & 0xFFFFFFFFUL;
250     regs->QueueDeviceHigh = (virtio_metadata_paddr + used_off) >> 32;
251     regs->QueueReady = 1;
252
253     /* Finish initialisation */
254     regs->Status |= VIRTIO_DEVICE_STATUS_DRIVER_OK;
255 }
256
257 static gpu_resp_status_t virtio_gpu_to_sddf_resp_status(enum virtio_gpu_ctrl_
258 {
259     switch (type) {
260     case VIRTIO_GPU_RESP_OK_DISPLAY_INFO:
261     /* FALLTHROUGH */
262     case VIRTIO_GPU_RESP_OK_NODATA:
```

Systems pr[illegible]amme[illegible]are the best!

- Careful e[illegible]neering!
- Strong l[illegible]ge features!

[illegible]od enou[illegible]

- Drivers are hard to get right.
- Type systems help, but are limited (especially if the system is decidable).

Formal Verification

Natural response:

- Many significant pitfalls in driver code.
- Programmers can't guarantee the absence of bugs.
- ∴ Prove the absence of bugs.

Prove **functional correctness**.

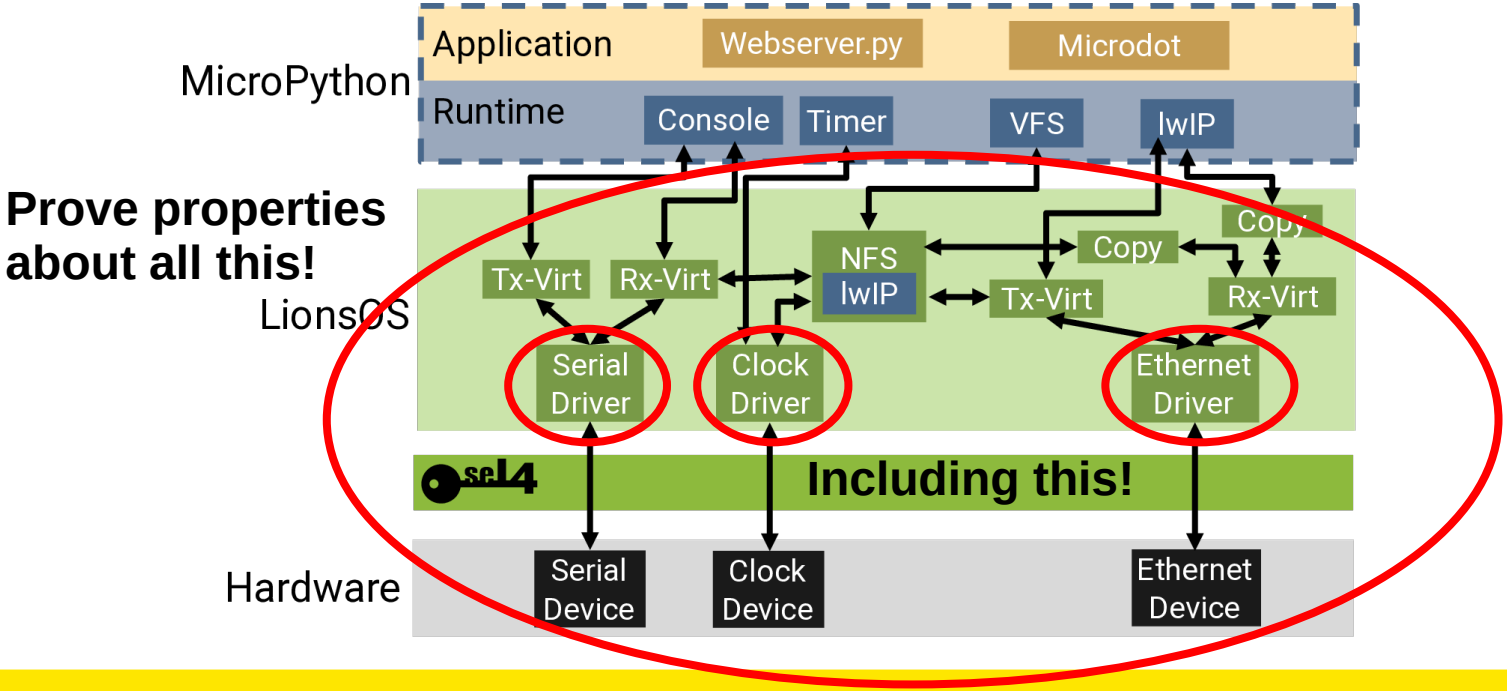
OS µkernel

First verification of functional correctness for a significant piece of systems software.

Prove a relationship between the inputs and outputs.

A bigger story

Prove **functional correctness** of the ~~driver~~ **OS**.



My Claim

Formal verification of real-world device drivers is

necessary if you...

note that things can go wrong with device drivers

observe that device drivers are difficult to write correctly

want to see a verified OS

My Claim

Formal verification of real-world device drivers is

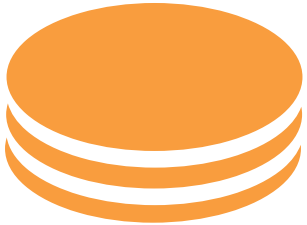
necessary and **attainable?**

We have verified a **real-world Ethernet NIC device driver!**

LionsOS **Pancake** **Pancake-to-Viper**

Pancake

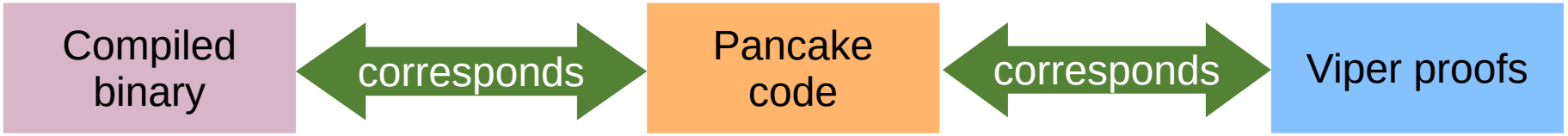
Has a **verified compiler**!



PANCAKE
A Language for Verified Systems Programming

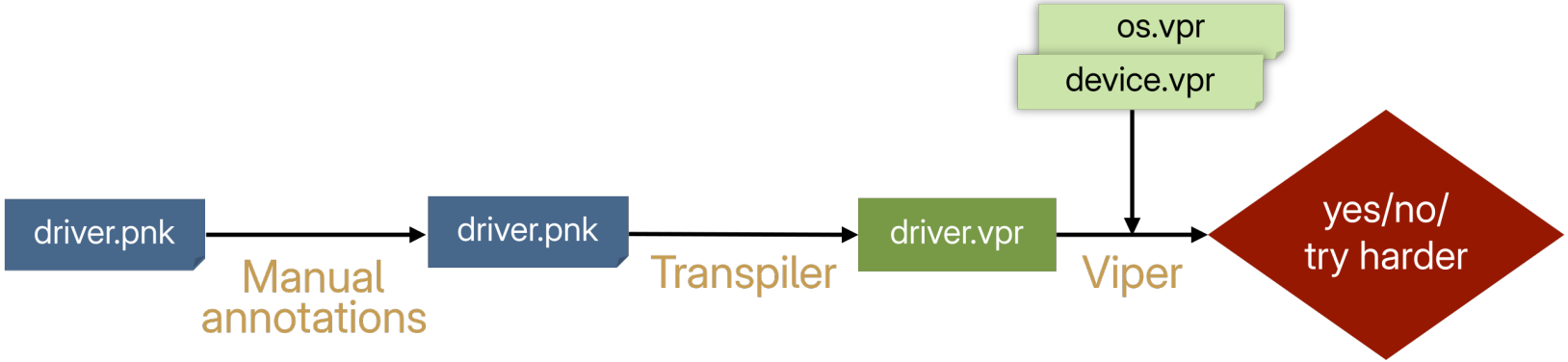
built on top of

CAKEML
A Verified Implementation of ML



Pancake-to-Viper transpiler

SMT-based automated deductive verifier
@ ETH Zürich



Why automated verification?

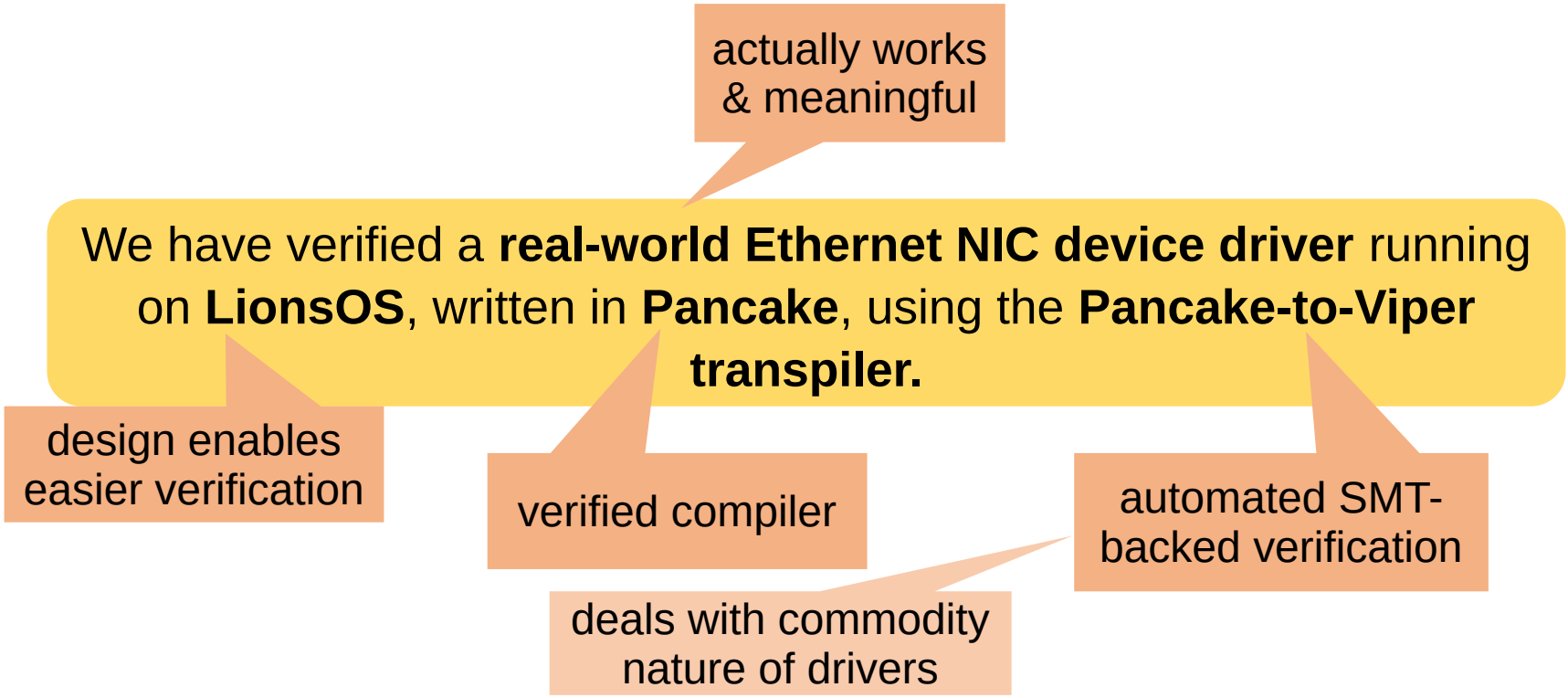
- Drivers appear regularly
  - and change from time to time
- 60% of the Linux kernel code is device drivers
  - ~25 million lines of code
- LionsOS (limited hardware support): 28 drivers
  - ~10k lines of code
  - More than seL4 kernel itself!

avoid bitrot

Automation & reusability!

Specifications alongside source code!

Summary



Summary

We have **verified** a **real-world Ethernet NIC device driver...**

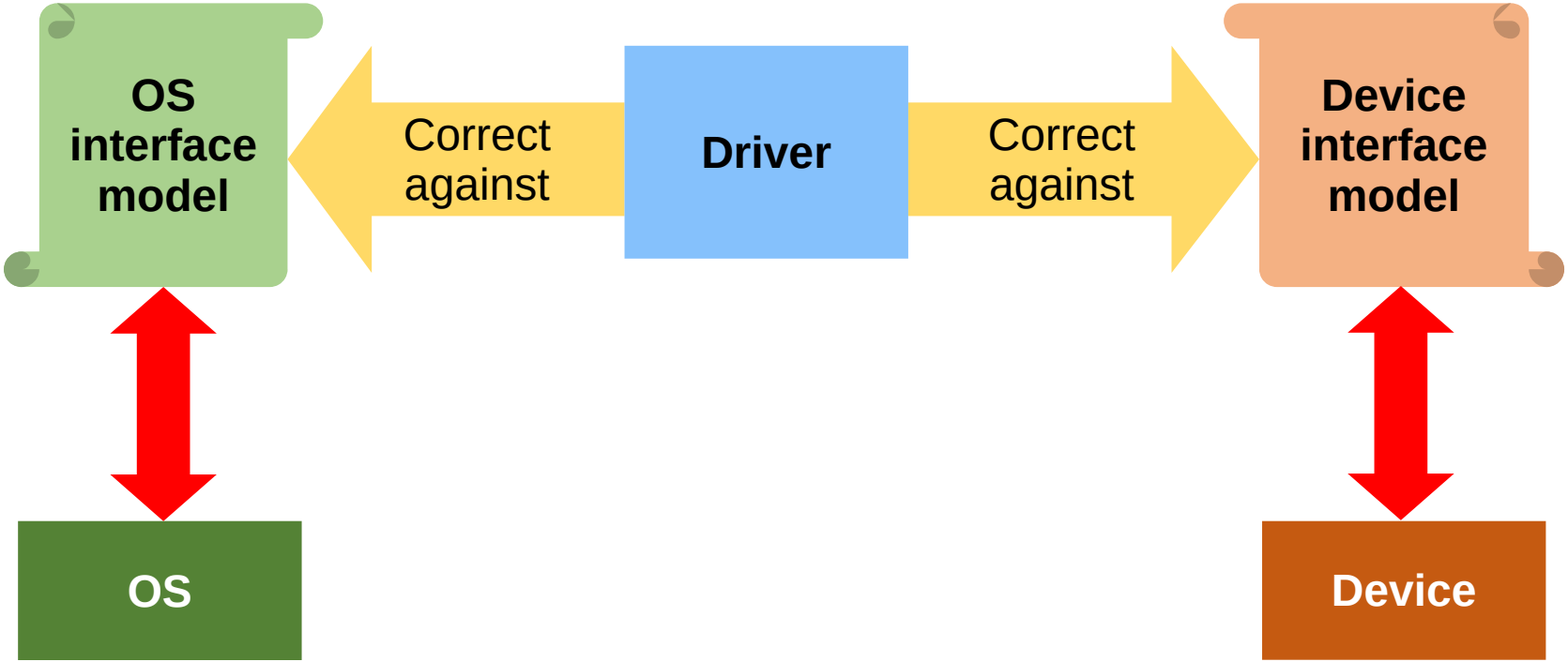
Verified what, exactly?

1. Device protocol adherence
2. Queue integrity
3. Correct buffer movement
4. Completion condition

availability

Most comprehensive driver verification to date.

Interface Model

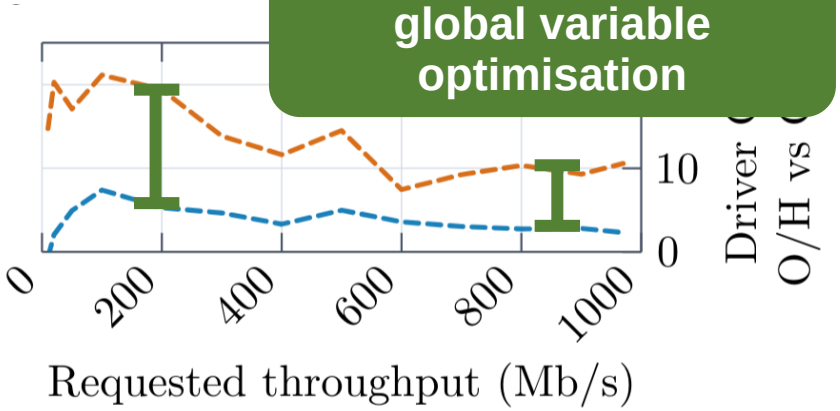


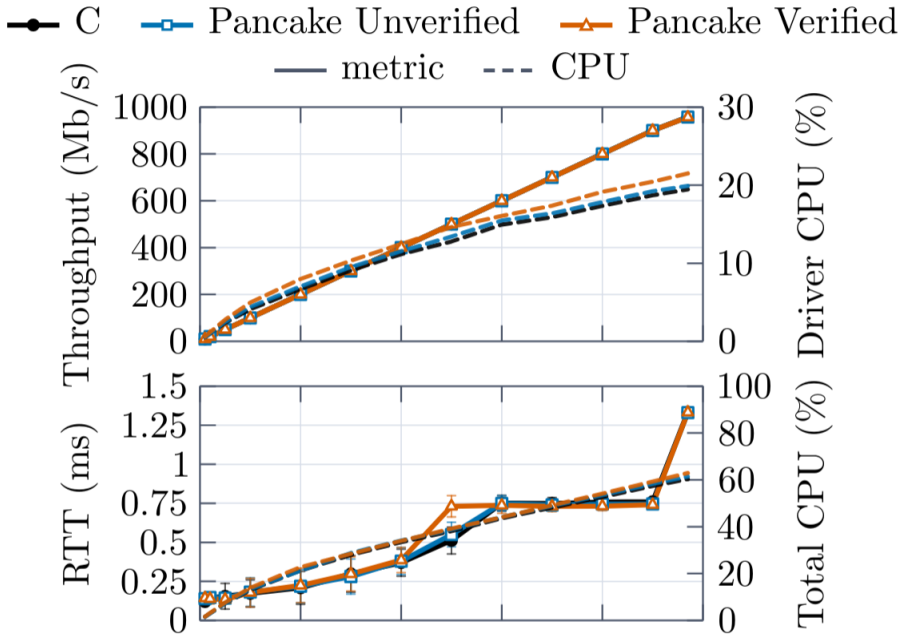
Performance

We have verified a real-world Ethernet NIC device driver...

Performance?

will be reached upon global variable optimisation





Summary

We have **comprehensively** verified a **real-world Ethernet NIC device driver** running on **LionsOS**, written in **Pancake**, using the **Pancake-to-Viper transpiler.**

Realistic? Repeatable?

Seems so,

but let's make it a nicer experience.

The Good

Time (p-wks)	**11.5**
Lines of Pancake	**426**
Lines of annotation	**1486**

annotation:Pancake ≈ 3.5

New vs paper!

Driver annot.	**1216**
Queue annot.	**270**
Device interface	**629**
OS interface	**431**

About me

- 3rd year undergraduate
- Coursework in Isabelle/HOL
- No specific skills in automated verification
- But persistent ;)

The Difficult (of SMT land)

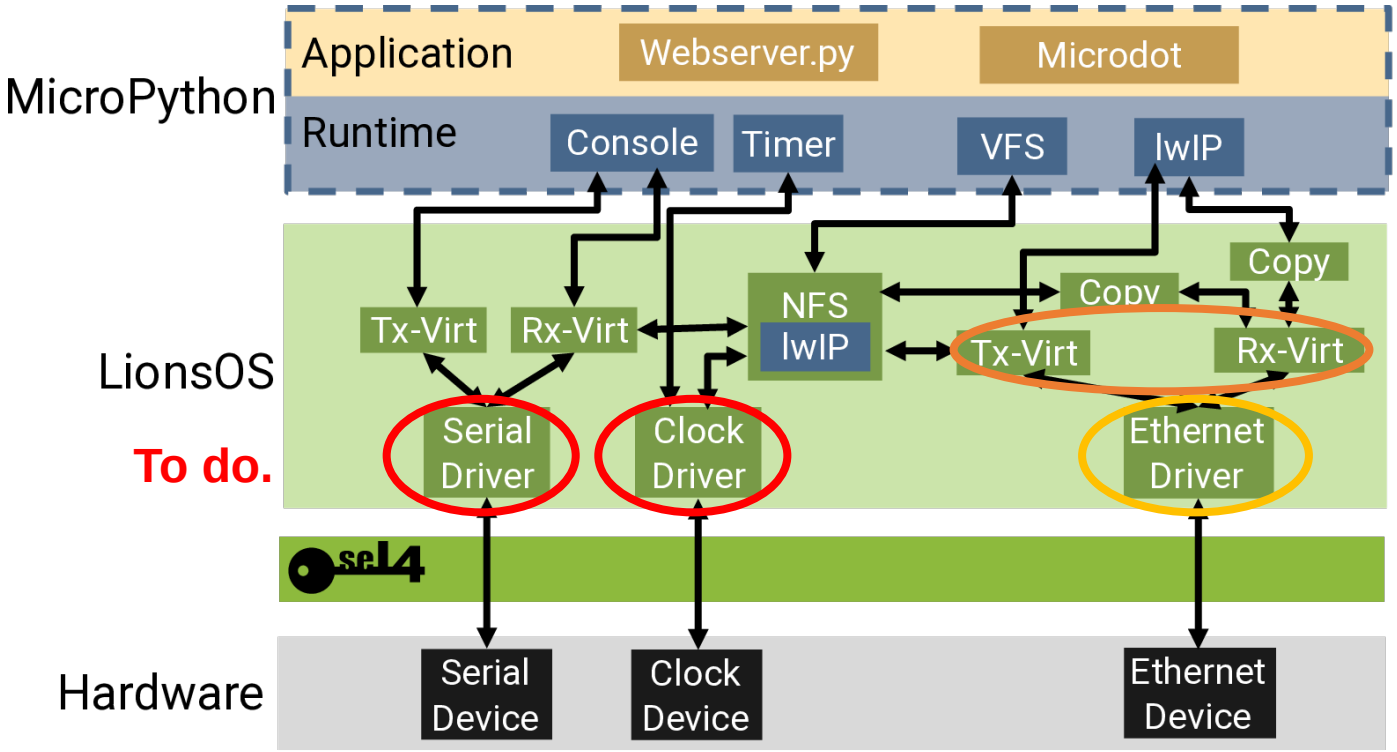
- Hand-holding the SMT solver with manual assert statements
- Mixing bitwise and integer operations
  - Structure the code to avoid mixed bitvector-integer reasoning
- **No longer verification time** **New vs paper!**
  - All functions now under 15 minutes (on a modern desktop)
  - Related to bitvector issues

+ Viper-specific issues we have ideas for how to deal with

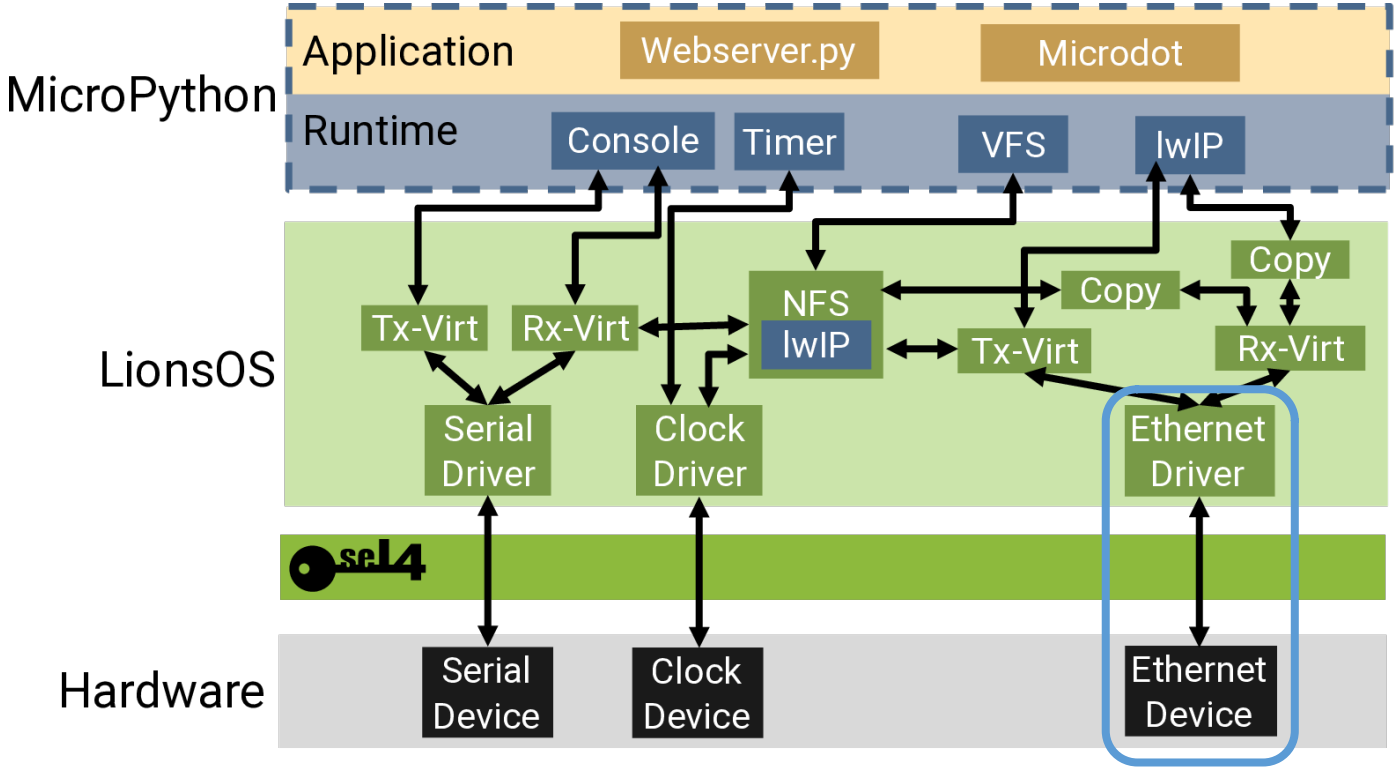
What’s next?

- bitvectors, etc
- Verification of transpiler (reduce trusted computing base)

LionsOS I



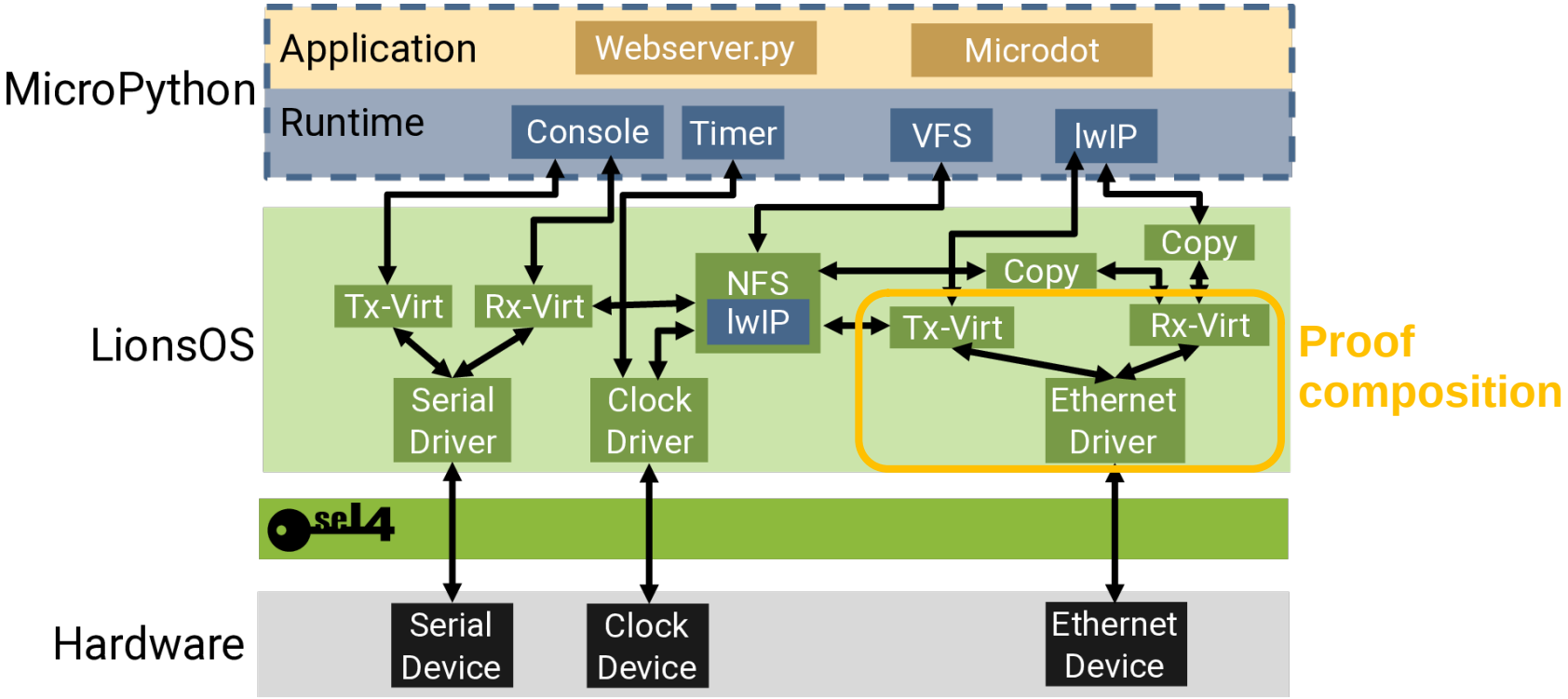
LionsOS II



High-Fidelity Specification of Real World Devices [Murphy et. al. **PLOS'25**]

Hardware-software interface formalisation

LionsOS II



So?

Verification of device driver **functional correctness** is **necessary**.

We have verified a **real-world Ethernet NIC device driver** running on **LionsOS**, written in **Pancake**, using the **Pancake-to-Viper transpiler.**

and

- after ironing out rough edges, seemingly **at scale**
- more work will **minimise trusted computing base**
- towards a **verified OS**.

Questions?