Termite: Driver Synthesis

Drivers!

- Historically buggy, widely varying in quality
- BSOD was historically caused by bad drivers
- Drivers are untrusted 3rd party code running with kernel privileges

Imagining a better driver

- Verification? Bounded model checking? Better testing?

- Intuition: most drivers in the same class are (basically) the same
  - All hardware in the same class act more or less the same
  - All drivers in the same class will use the same OS interface

Termite: at a high level

1. Specify OS interface

2. Specify the device-class behavior

3. Specify the device hardware behavior

Gluing it all together

- The three specifications “communicate” via messages

- Behavior of each part is specified as a state machine
  - Specified in a sort of process logic

OS interface

1. Specify OS interface

2. Specify the device-class behavior

3. Specify the hardware behavior

OS behavior

- OS requests are modeled as incoming messages to driver

- This state machine specifies the possible requests sent to the driver (and expected response)

OS behavior

- Unclear where this is derived from

- Likely from existing code

Device-class behavior

1. Specify OS interface

2. Specify the device-class behavior

3. Specify the hardware behavior

Device-class specifications

- High-level description of what the device can do
  - what “kind” of devices is it, e.g. ethernet adapter, SD card reader, etc.

- Agreed upon by regulatory body like IEEE
  - possibly extended by specific device manufacturers

Hardware behavior

1. Specify OS interface

2. Specify the device-class behavior

3. Specify the hardware behavior

Hardware specification

- Maps high-level device-class messages into low level hardware actions
  - set this register to X value
  - wait for Y interrupt…
- Hardware specific
  - but the device-class “interface” makes it reusable across OS’s

Hardware specification

- Informal, plain text documentation
  - i.e., manufacturer data sheets
  - But… incomplete, possibly out-of-sync with device
- Existing reference implementation
  - Exact, unambiguous spec
  - But… bugs in existing implementation will carry over
- Hardware RTL
  - Exact, 100% complete and in-sync with H/W
  - But… usually proprietary, not easy to get to

Synthesis algorithm

- There are two state machines (OS/Device)
- They are merged into a big state machine encoding all possible behaviors
  - This isn’t necessarily the behavior we WANT
- Synthesis as two-player reachability game:
  - Device driver is winning strategy in the game

Generating C code

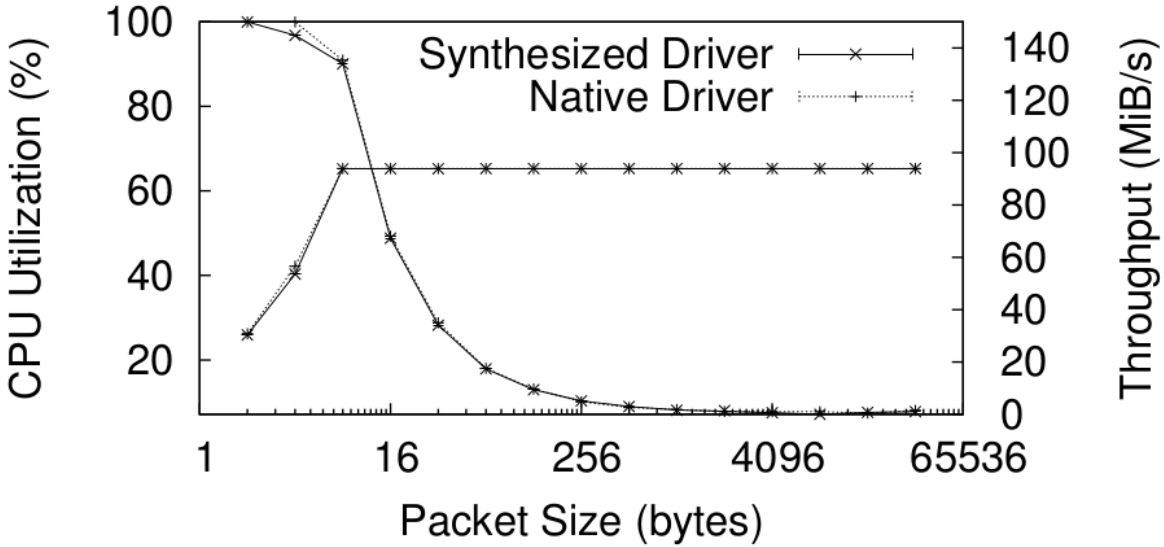
- Driver is output as a big mess of C code

	R5C822	AX88772
Native Linux driver	1174	1200
Device interface	653	463
OS interface (SD/Ethernet)	378	213
Bus interface (PCI/USB)	263	96
Synthesised driver	4667	2620

- Refer to example; they don't really explain the process

Performance

Nearly identical:



Limitations

- Synthesized drivers are single threaded
- Some manual hacking is required
  - Synthesis cannot handle editing buffers
- Drivers are required to look like state machines
  - Moving away from this in Termite2?

Discussion

- Is writing their spec easier that writing the code?
- Can we autogenerate from RTL? to RTL?
- They rely on separation of concerns: OS side vs. device side. Where else could this be leveraged?
- Could a system like this be widely adopted?