barkhausen institut

M³: Taking Microkernels to the Next Level

Nils Asmussen

FOSDEM, 02/02/2020, Brussels

About Me

- Nils Asmussen
- PhD last year at the OS chair of the TU Dresden
- Low-level system programming and microkernels
- Worked on several microkernel-based OSes in the past
  - Escape, own hobby OS (presented here in 2013): `https://github.com/Nils-TUD/Escape`
  - NRE, userland for NOVA: `https://github.com/TUD-OS/NRE`
  - M^3, presented today: `https://github.com/TUD-OS/M3`
- Since 2019 at the Barkhausen Institut

Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 30 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators

Barkhausen Institut

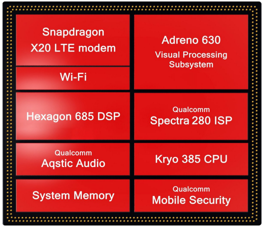
- Research institute in Dresden, founded end of 2017
- Currently about 30 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators

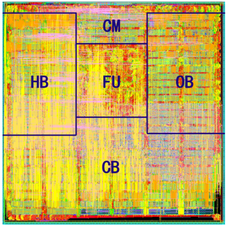
Wireless	RF Design	Privacy
Lab	MPSoC	OS

Motivation

- Microkernel-based systems have proven valuable for several objectives
    - Security
    - Robustness
    - Real time
    - Flexibility
- Recently, new challenges are coming from the hardware side
    - Heterogeneous systems
    - Third-party components
    - Security issues of complex general-purpose cores

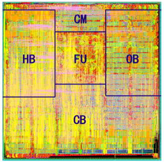
Heterogeneous Systems

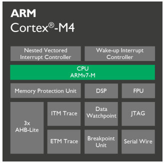




- Demanded by performance and energy requirements
- Big challenge for OSes: single shared kernel on all cores does no longer work
- OSes need to be prepared for processing elements with different feature sets

Third-party Components



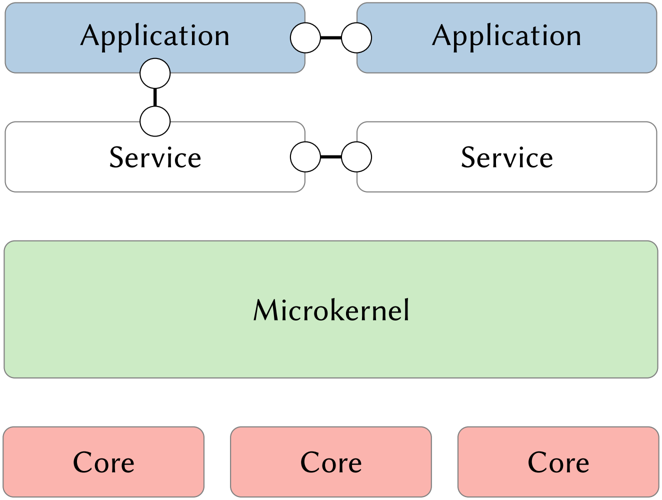


- Market pressure forces us to integrate third-party components
- We should not trust these components
- Currently, often no isolation between them
- Bug in such a component can compromise whole system (see Broadcom incident)

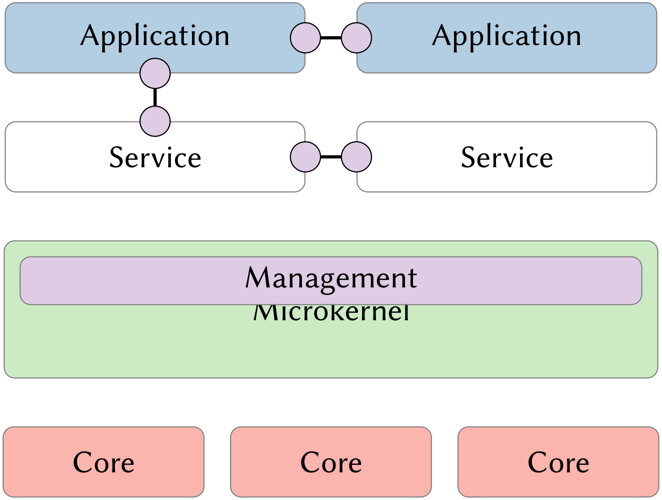
Security Issues of Complex General-purpose Cores

- 20 known attacks (and counting . . .)
- Allow to leak private data, sometimes bypassing all security measures of the core
- Mitigations exist, but these are complex and costly
- These security holes have been lurking in CPUs for many years
- Should we still trust these complex cores to properly enforce the isolation between different software components?

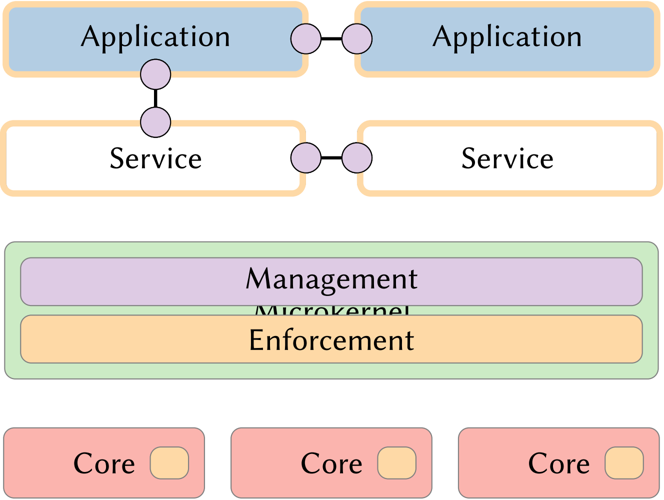
Microkernel-based System as Foundation



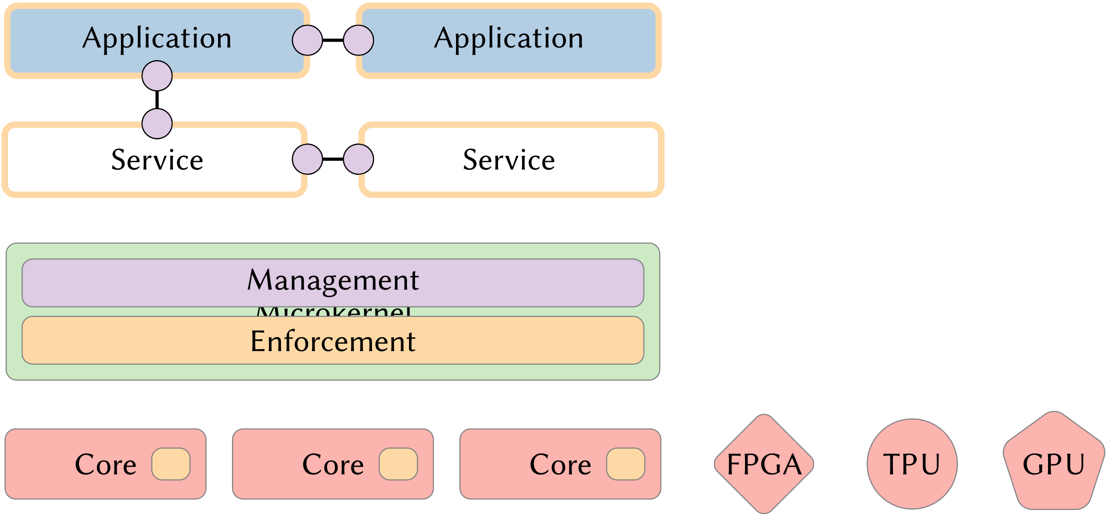
Microkernel-based System as Foundation



Microkernel-based System as Foundation



Microkernel-based System as Foundation

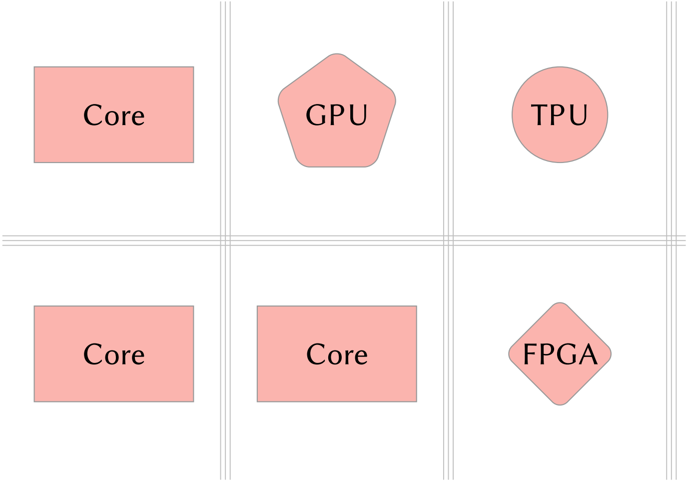


Outline

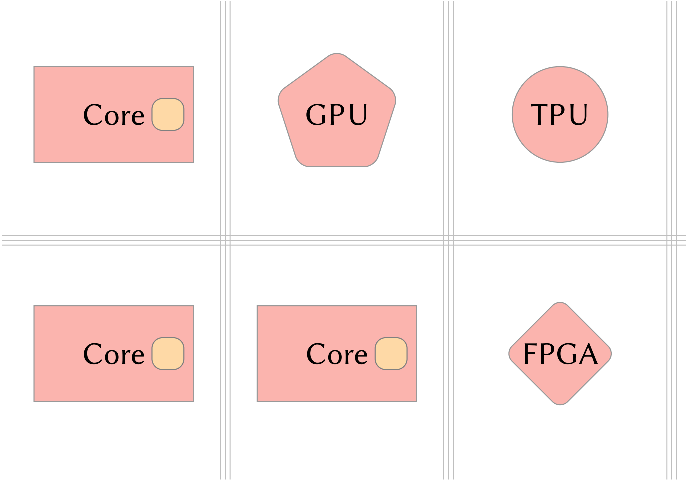
Outline

Hardware/Operating System Co-Design

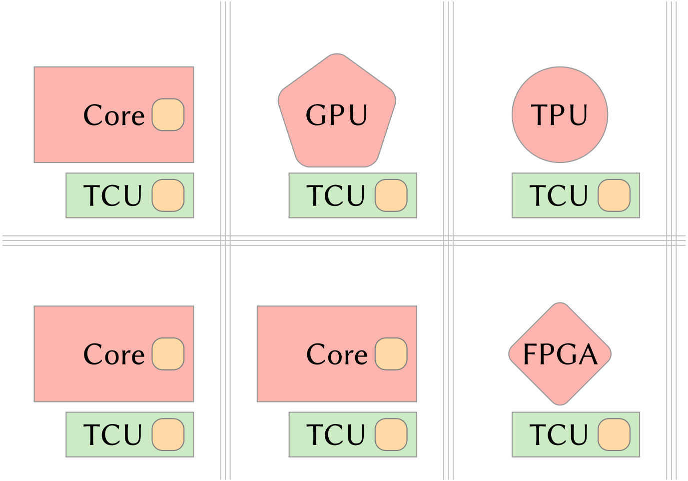
Hardware/Operating System Co-Design



Hardware/Operating System Co-Design



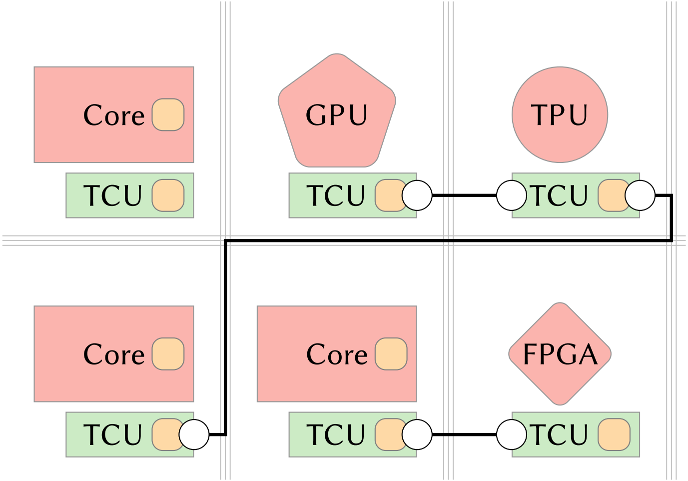
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component

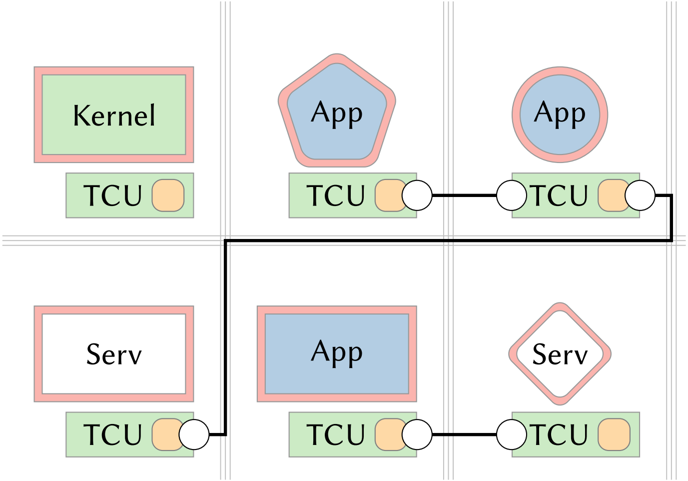
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component

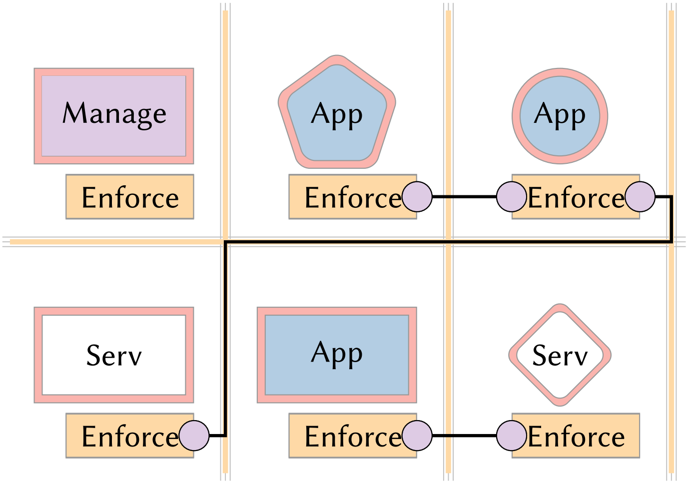
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component
- Kernel on dedicated PE

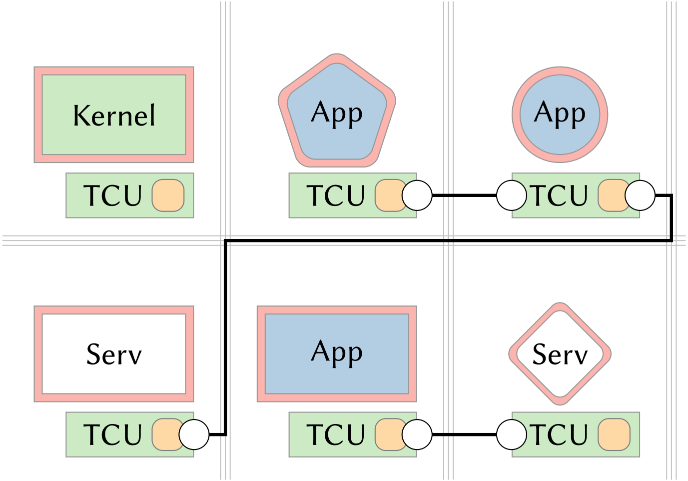
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component
- Kernel on dedicated PE
- Kernel manages, TCU enforces

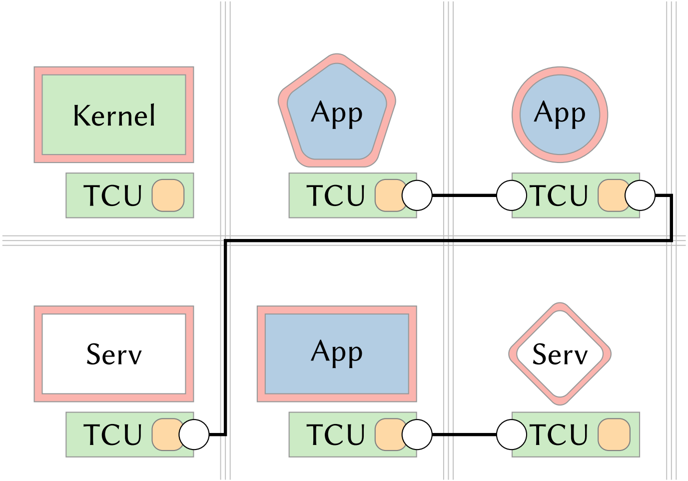
Hardware/Operating System Co-Design



Takes μ-kernels to the next level:

- TCU as secure foundation

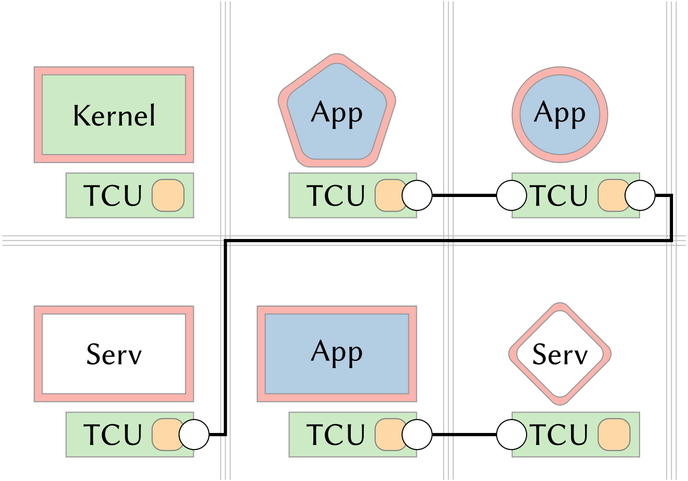
Hardware/Operating System Co-Design



Takes μ-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface

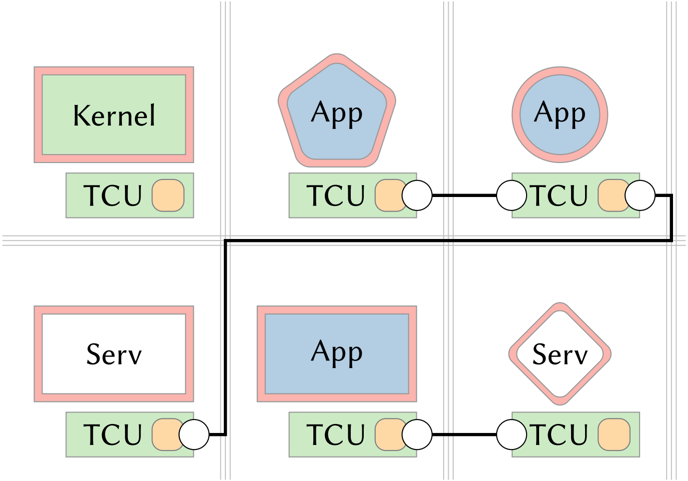
Hardware/Operating System Co-Design



Takes μ-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface
- Untrusted HW comp.:
  Protected by TCU

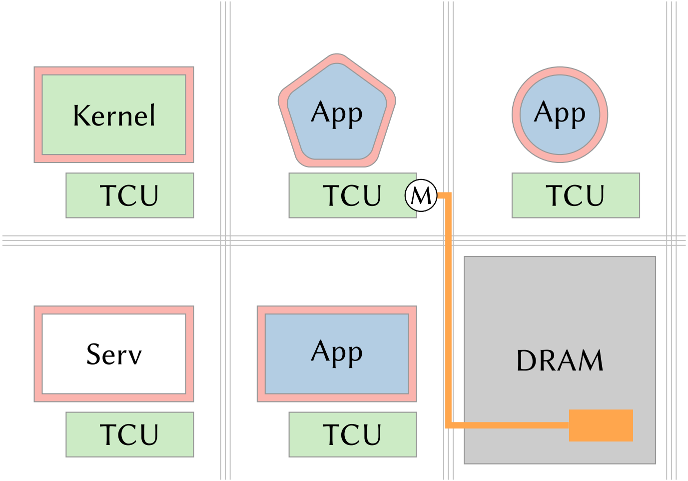
Hardware/Operating System Co-Design



Takes μ-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface
- Untrusted HW comp.:
  Protected by TCU
- Side channels:
  Physical isolation

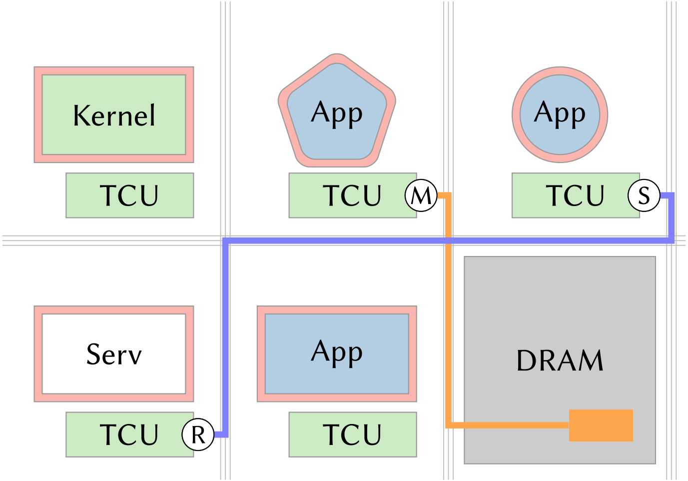
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)

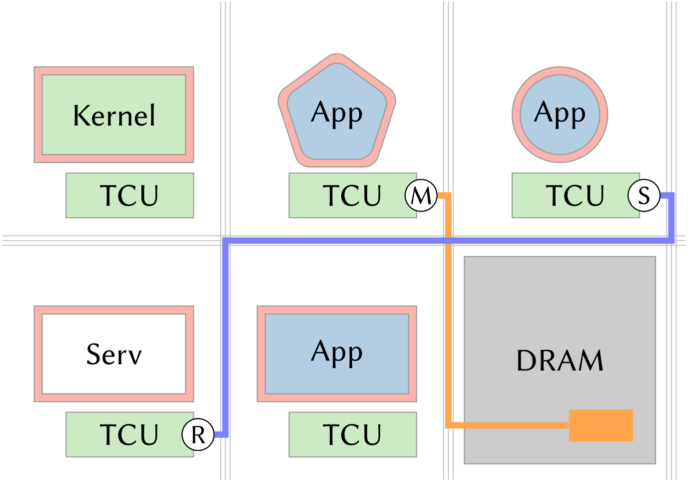
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint

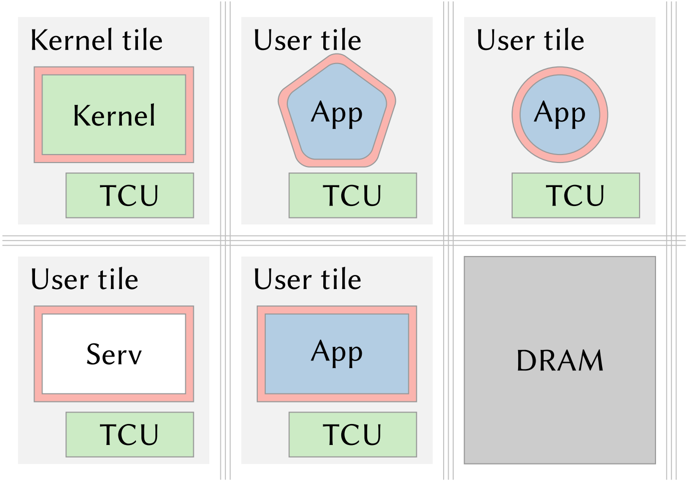
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint
- Replies for RPC

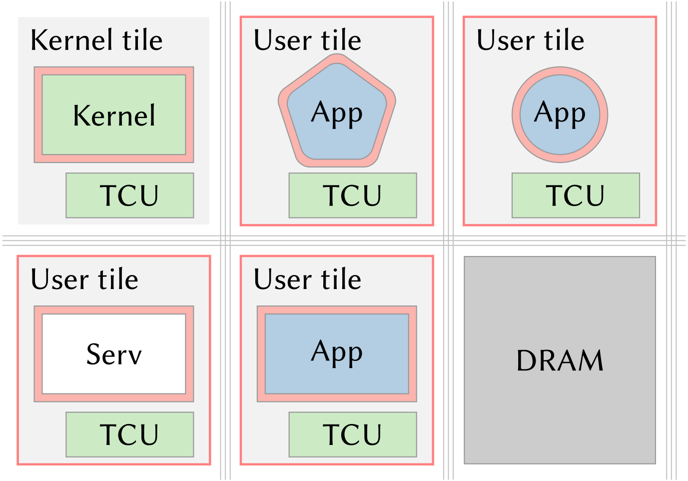
Isolation



TCU-based isolation:

- Additional protection layer

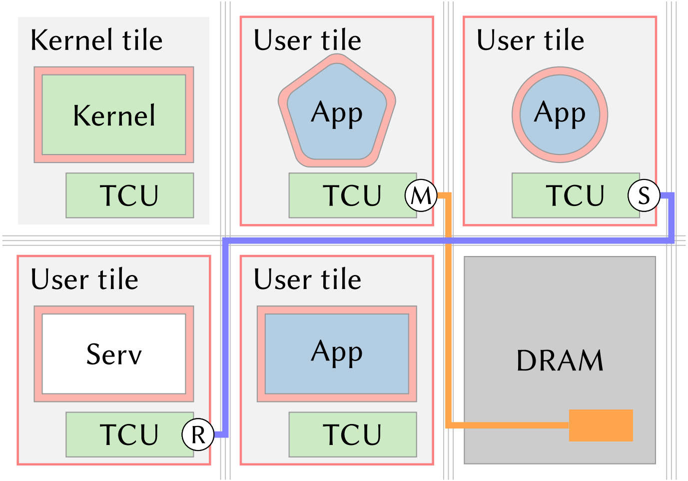
Isolation



TCU-based isolation:

- Additional protection layer

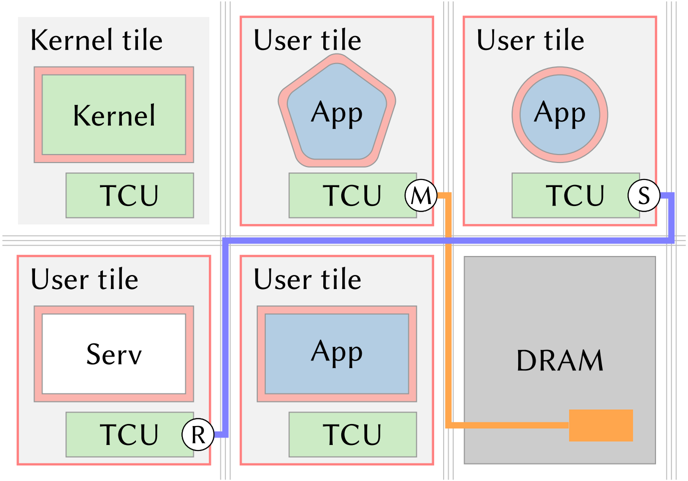
Isolation



TCU-based isolation:

- Additional protection layer
- Only kernel tile can establish communication channels

Isolation



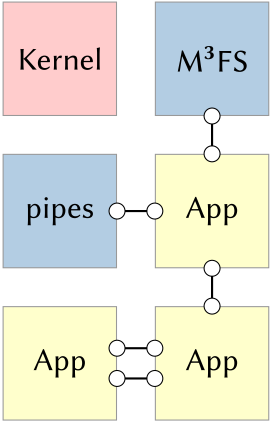
TCU-based isolation:

- Additional protection layer
- Only kernel tile can establish communication channels
- User tiles can only use established channels

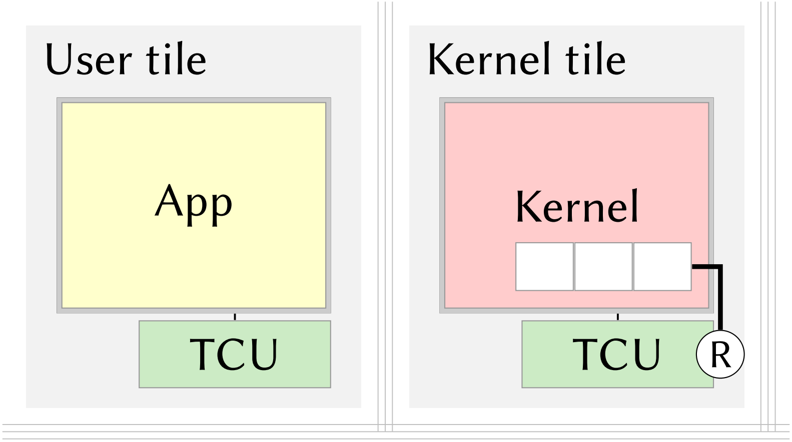
Outline

OS Design

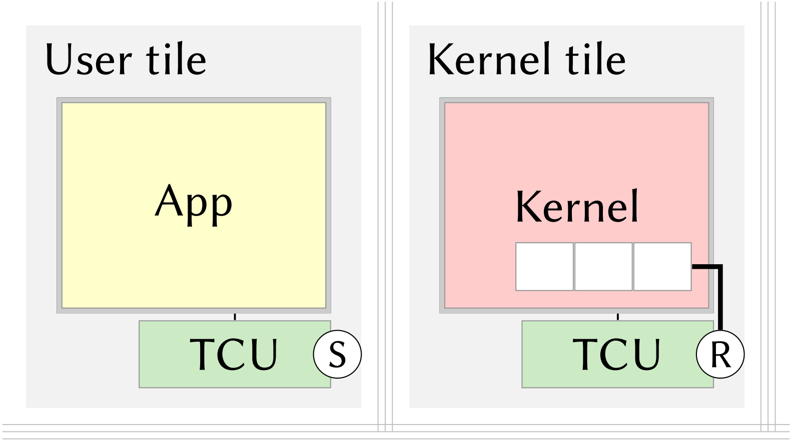
- M^3: **M**icrokernel-based syste**m** for het. **m**anycores (or L4 $\pm$ 1)
- Implemented from scratch in Rust and C++
- Drivers, filesystems, etc. implemented on user tiles
- Kernel manages permissions, using capabilities
- TCU enforces permissions (communication, memory access)
- Kernel is independent of other tiles



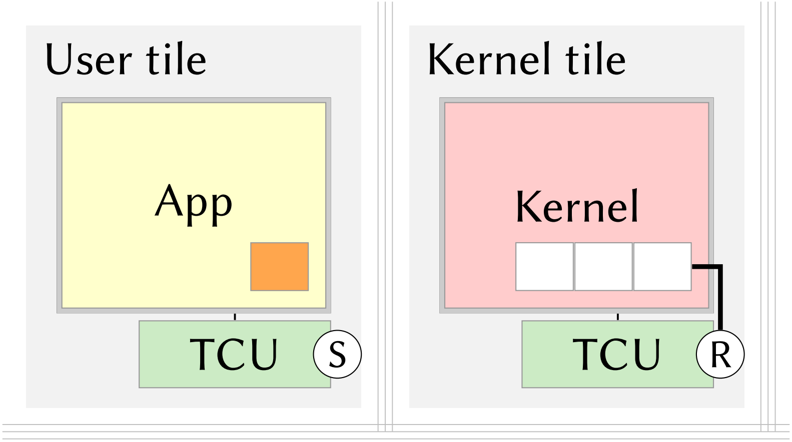
M³ System Call



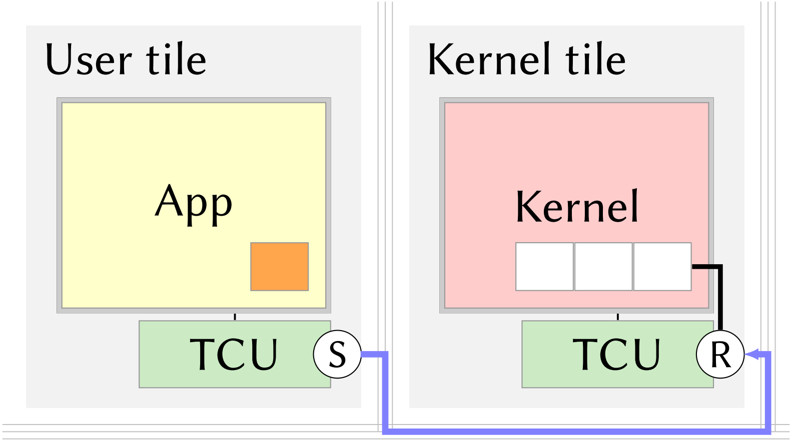
M³ System Call



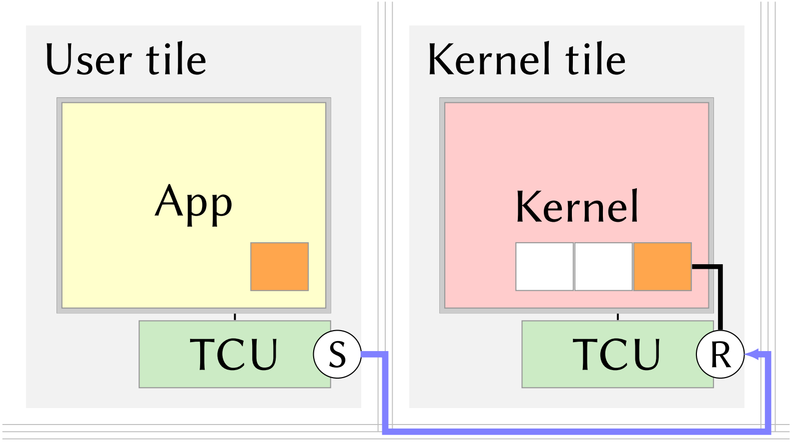
M³ System Call



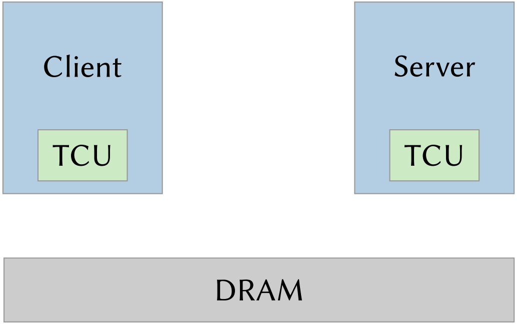
M³ System Call



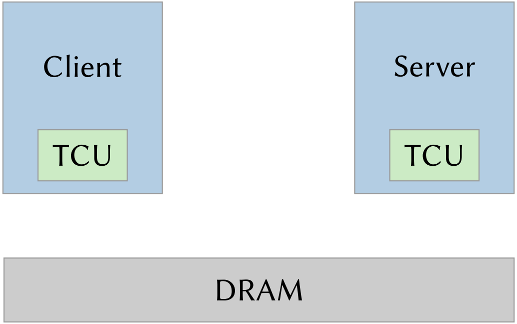
M^3 System Call



OS Service Access



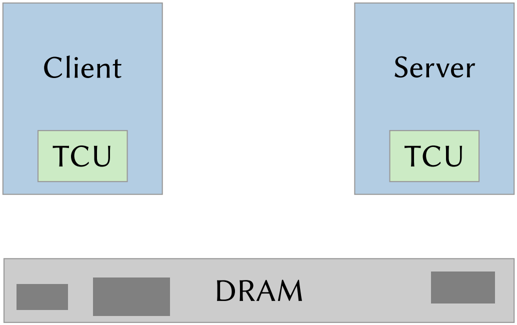
OS Service Access



File Protocol:

- Used for: files, pipes, …

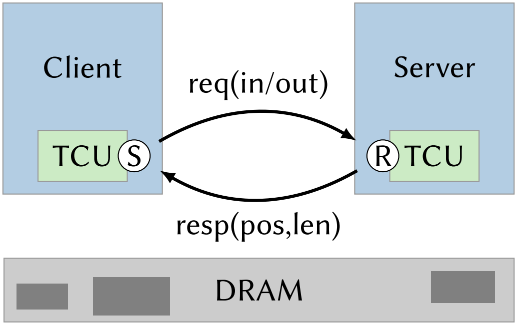
OS Service Access



File Protocol:

- Used for: files, pipes, ...
- Data in memory

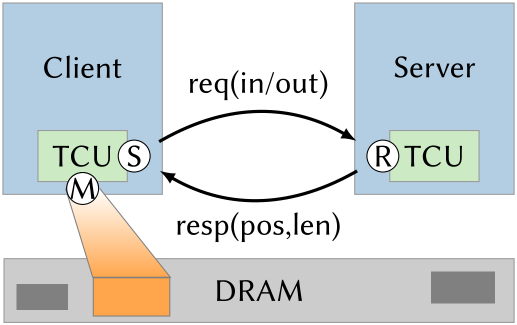
OS Service Access



File Protocol:

- Used for: files, pipes, ...
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece

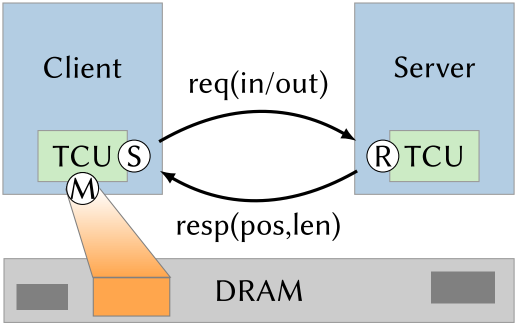
OS Service Access



File Protocol:

- Used for: files, pipes, ...
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP

OS Service Access



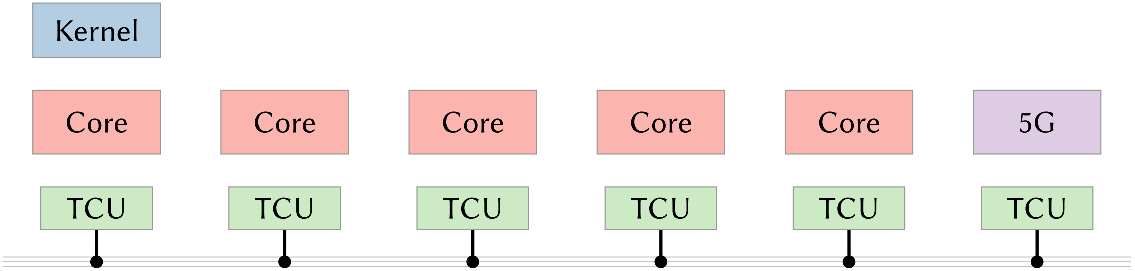
File Protocol:

- Used for: files, pipes, …
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP
- Client accesses data via TCU

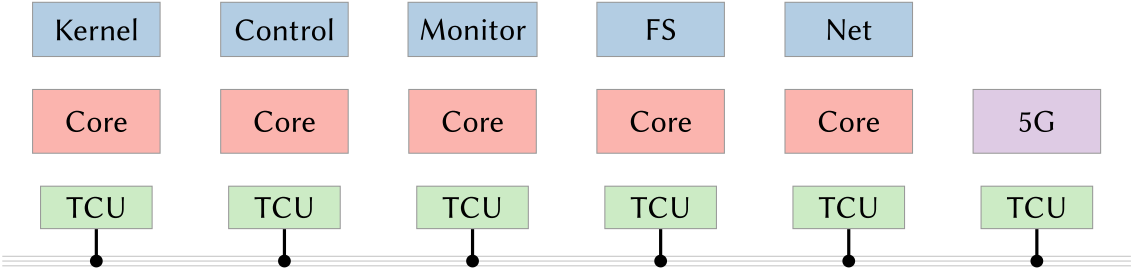
Outline

1. The New System Architecture
2. M^3: The Operating System
3. What are the Benefits?

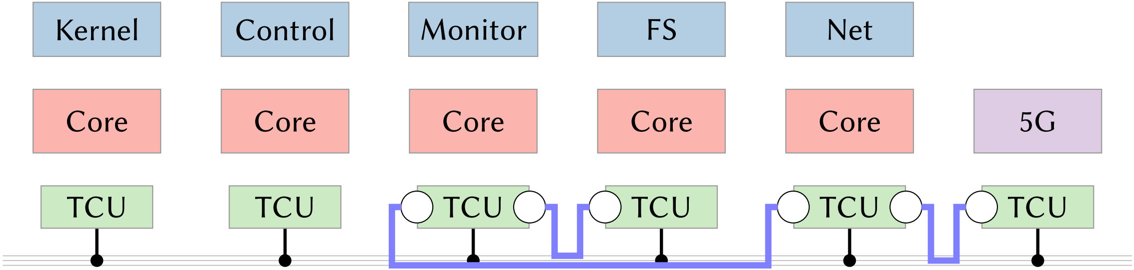
Example System



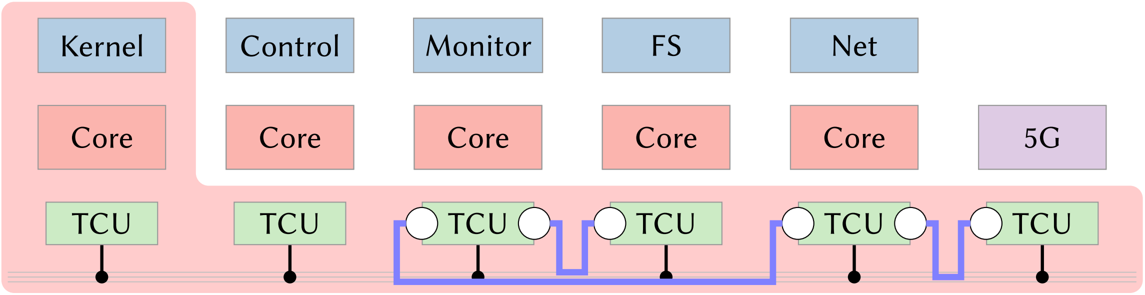
Example System



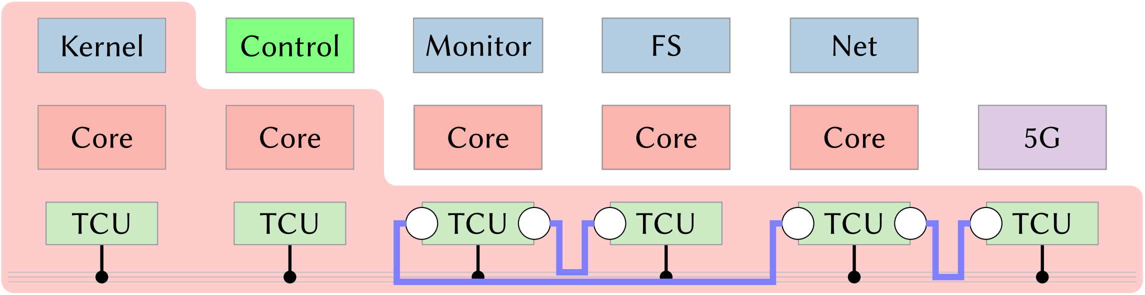
Example System



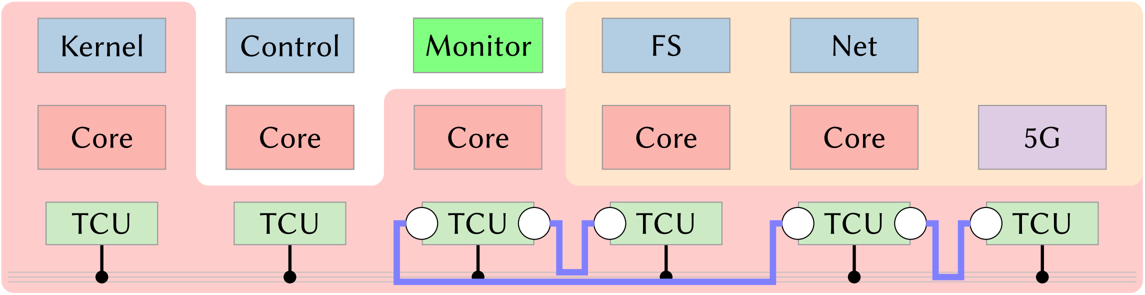
Example System – TCB



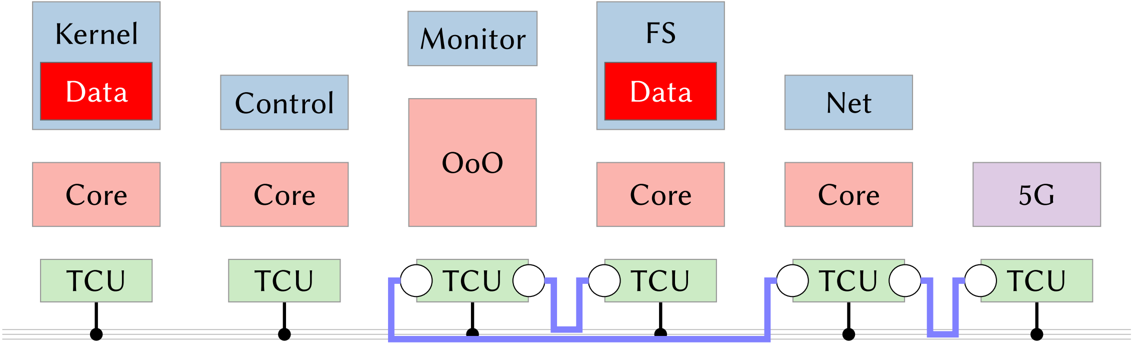
Example System – TCB



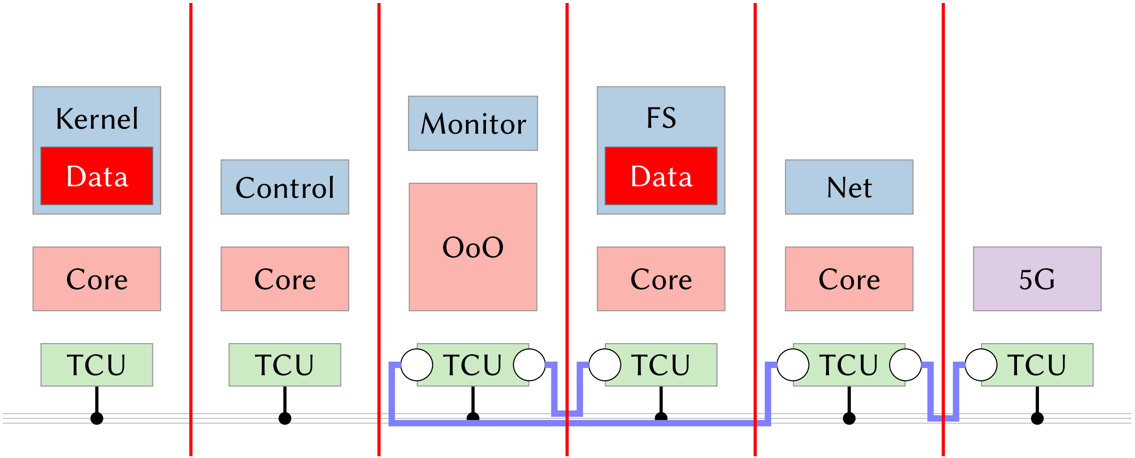
Example System – TCB



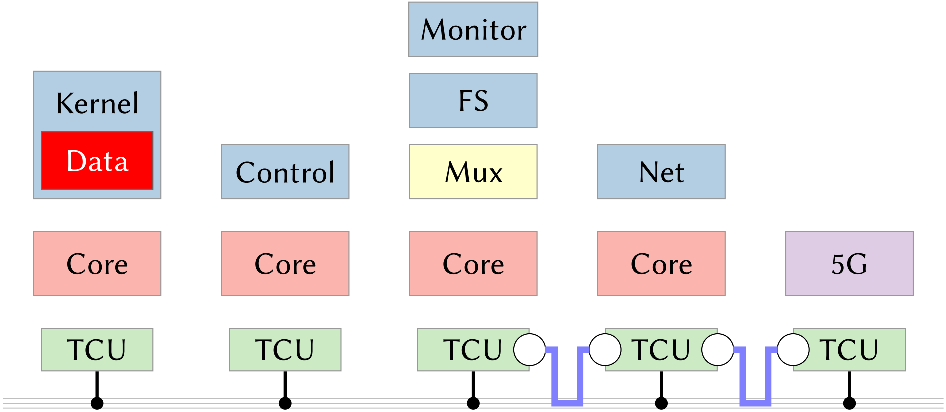
Example System – Untrusted Core



Example System – Untrusted Core



Example System – Sharing (WIP)

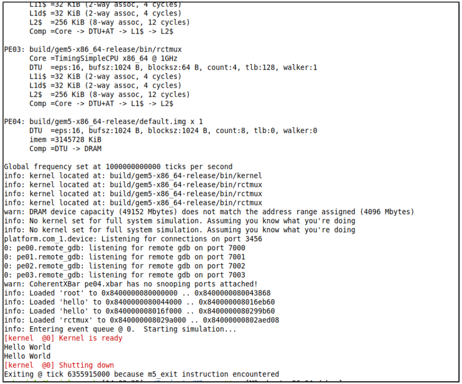


Prototype Platforms

```
      L1i$ =32 KiB (2-way assoc, 4 cycles)
      L1d$ =32 KiB (2-way assoc, 4 cycles)
      L2$  =256 KiB (8-way assoc, 12 cycles)
      Comp =Core -> DTU+AT -> L1$ -> L2$

PE03: build/gem5-x86_64-release/bin/rctmux
      Core =TimingSimpleCPU x86_64 @ 1GHz
      DTU  =eps:16, bufsz:1024 B, blocksz:64 B, count:4, tlb:128, walker:1
      L1i$ =32 KiB (2-way assoc, 4 cycles)
      L1d$ =32 KiB (2-way assoc, 4 cycles)
      L2$  =256 KiB (8-way assoc, 12 cycles)
      Comp =Core -> DTU+AT -> L1$ -> L2$

PE04: build/gem5-x86_64-release/default.img x 1
      DTU  =eps:16, bufsz:1024 B, blocksz:1024 B, count:8, tlb:0, walker:0
      imem =3145728 KiB
      Comp =DTU -> DRAM

Global frequency set at 1000000000000 ticks per second
info: kernel located at: build/gem5-x86_64-release/bin/kernel
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
warn: DRAM device capacity (49152 Mbytes) does not match the address range assigned (4096 Mbytes)
info: No kernel set for full system simulation. Assuming you know what you're doing
info: No kernel set for full system simulation. Assuming you know what you're doing
platform.com_1.device: Listening for connections on port 3456
0: pe00.remote_gdb: listening for remote gdb on port 7000
0: pe01.remote_gdb: listening for remote gdb on port 7001
0: pe02.remote_gdb: listening for remote gdb on port 7002
0: pe03.remote_gdb: listening for remote gdb on port 7003
warn: CoherentXBar pe04.xbar has no snooping ports attached!
info: Loaded 'root' to 0x8400000080000000 .. 0x8400000080043868
info: Loaded 'hello' to 0x8400000080044000 .. 0x840000008016eb60
info: Loaded 'hello' to 0x840000008016f000 .. 0x8400000080299b60
info: Loaded 'rctmux' to 0x840000008029a000 .. 0x84000000802aed08
info: Entering event queue @ 0.  Starting simulation...
[kernel  @0] Kernel is ready
Hello World
Hello World
[kernel  @0] Shutting down
Exiting @ tick 6355915000 because m5_exit instruction encountered
```

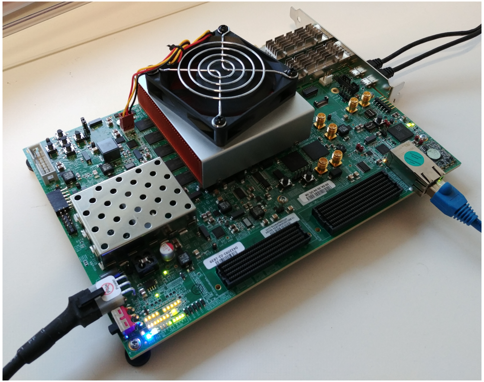
gem5 simulator

Prototype Platforms



```
      L1i$ =32 KiB (2-way assoc, 4 cycles)
      L1d$ =32 KiB (2-way assoc, 4 cycles)
      L2$  =256 KiB (8-way assoc, 12 cycles)
      Comp =Core -> DTU+AT -> L1$ -> L2$

PE03: build/gem5-x86_64-release/bin/rctmux
      Core =TimingSimpleCPU x86_64 @ 1GHz
      DTU  =eps:16, bufsz:1024 B, blocksz:64 B, count:4, tlb:128, walker:1
      L1i$ =32 KiB (2-way assoc, 4 cycles)
      L1d$ =32 KiB (2-way assoc, 4 cycles)
      L2$  =256 KiB (8-way assoc, 12 cycles)
      Comp =Core -> DTU+AT -> L1$ -> L2$

PE04: build/gem5-x86_64-release/default.img x 1
      DTU  =eps:16, bufsz:1024 B, blocksz:1024 B, count:8, tlb:0, walker:0
      imem =3145728 KiB
      Comp =DTU -> DRAM

Global frequency set at 1000000000000 ticks per second
info: kernel located at: build/gem5-x86_64-release/bin/kernel
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
info: kernel located at: build/gem5-x86_64-release/bin/rctmux
warn: DRAM device capacity (49152 Mbytes) does not match the address range assigned (4096 Mbytes)
info: No kernel set for full system simulation. Assuming you know what you're doing
info: No kernel set for full system simulation. Assuming you know what you're doing
platform.com_1.device: Listening for connections on port 3456
0: pe00.remote_gdb: listening for remote gdb on port 7000
0: pe01.remote_gdb: listening for remote gdb on port 7001
0: pe02.remote_gdb: listening for remote gdb on port 7002
0: pe03.remote_gdb: listening for remote gdb on port 7003
warn: CoherentXBar pe04.xbar has no snooping ports attached!
info: Loaded 'root' to 0x8400000080000000 .. 0x8400000080043868
info: Loaded 'hello' to 0x8400000080044000 .. 0x840000008016eb60
info: Loaded 'hello' to 0x840000008016f000 .. 0x8400000080299b60
info: Loaded 'rctmux' to 0x840000008029a000 .. 0x84000000802aed08
info: Entering event queue @ 0.  Starting simulation...
[kernel  @0] Kernel is ready
Hello World
Hello World
[kernel  @0] Shutting down
Exiting @ tick 6355915000 because m5_exit instruction encountered
```

gem5 simulator



FPGA

Demo

Summary

- Microkernels are great!
- Their ideas can also be applied to hardware:
  - Trusted communication unit per tile
  - Isolated software *and* hardware components on top
- Has several additional benefits:
  - Allows to securely integrate untrusted third-party components
  - Prevents (known) side-channel attacks by physical isolation
  - Simplifies heterogeneous systems by uniform interface
- M^3 is available at `https://github.com/TUD-OS/M3`,
  gem5 extensions at `https://github.com/TUD-OS/gem5-dtu`