The Case for Cooperative Kernel Threads

Yanjin Zhu (student)
Leonid Ryzhyk (student)
Peter Chubb
Ihor Kuz
Gernot Heiser

NICTA, University of New South Wales, Open Kernel Labs
Preemptive Kernel Threads

• Most OS kernels are multithreaded
• Drivers run as part of the kernel
  – Need to deal with concurrent invocations
  – Shared state must be maintained
• Synchronisation is hard to get right!
  – Race conditions and deadlocks
  – 20%\(^1\) of bugs in device drivers

Event-based Device Drivers

```c
probe() {
    ...\n    write();
    msleep(10);
    read();
    ...\n}
```

```c
probe() {
    ...\n    write();
    drv->state = 1;
    schedule_timeout(10, drv, timeout);
    return;
}
```

```c
timeout(drv) {
    switch(drv->state) {
        case 1:
            read();
            break;
        ...
    }
}
```
Cooperative Threading

- The best of the two worlds
  - Atomic execution
  - Blocking Allowed
- Not supported in the Linux kernel
- Runtime support for cooperative threads
  - Create cooperative scheduling domains inside the kernel
  - High-performance message passing between preemptive and cooperative threads
Conclusion

- **Research Aim:**
  - Dealing with concurrency in device drivers

- **Problems:**
  - Pre-emptive threading
    - Synchronisation
  - Event-based threading
    - Stack ripping

- **Proposed Solution:**
  - Cooperative threading
    - Best of both worlds
    - Run-time support in Linux kernel