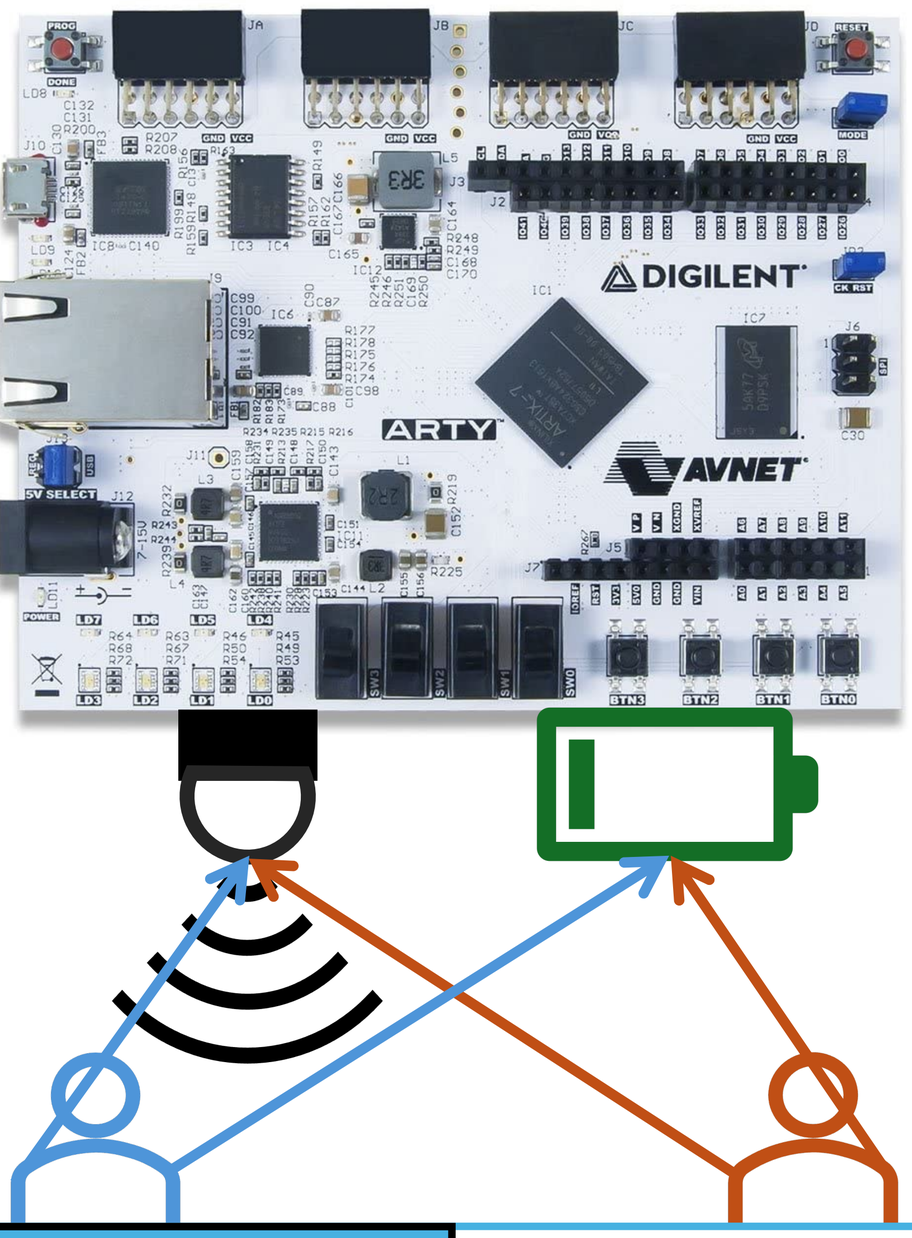
Kernel Functions: An Operating System for Memory-Constrained Devices

Yacqub Mohamed, Robbert van Renesse

Motivation

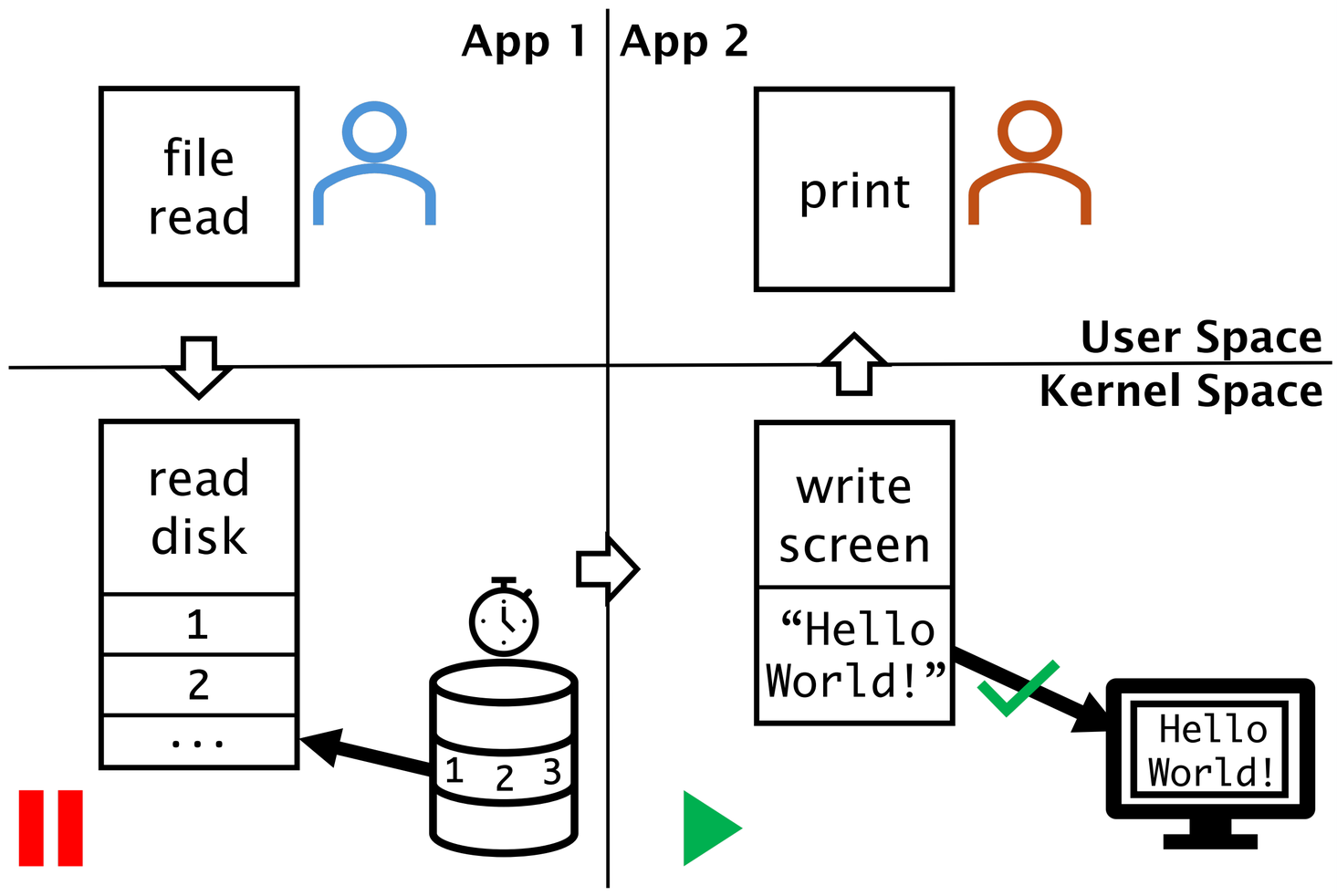
Want apps to share constrained devices

- Minimizes costs and area usage
- Maximizes device utilization



Traditional kernels multiplex device resources in a memory-wasteful way

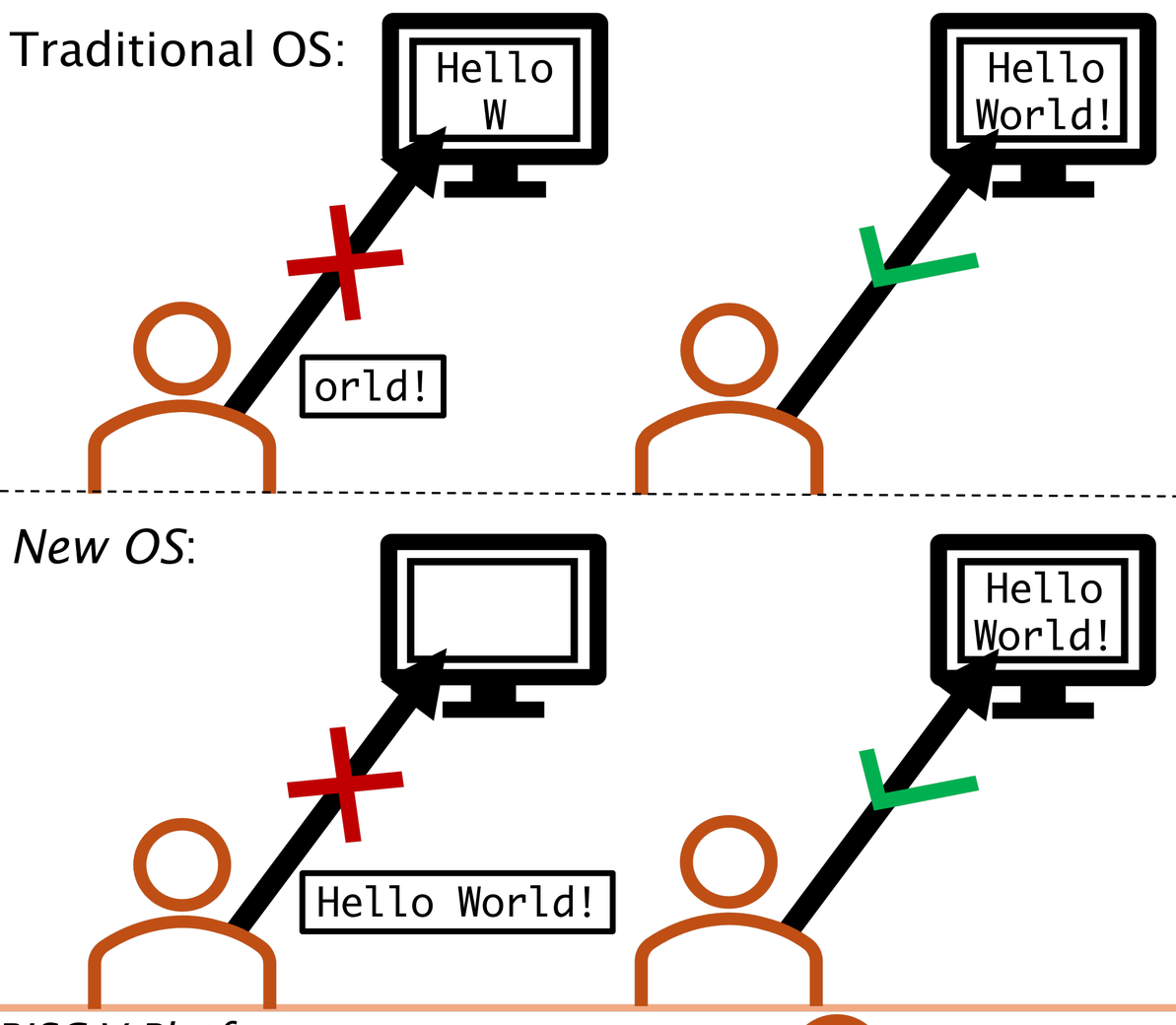
- **Apps invoke kernel routines to access devices**
  - Kernel stack per process enables other apps to run while one app is waiting for device
- **Kernel stacks must be generously sized and cannot be paged out**



Design

All processes share a single kernel stack, and access devices using **kernel functions**

- Kernel stack is **reused** and **volatile**
  - **Single Kernel Lock:** Only one process inside the kernel at a time
- Kernel functions are **atomic** and **idempotent**
  - **Atomic:** Run to completion with interrupts disabled in supervisor mode
  - **Idempotent:** Retried from beginning until successful



Implementation

Implemented in the EGOS-2000 OS on a RISC-V Platform

Every device has a single device buffer, and kernel functions only interact with device buffers

- Keyboard/Screen buffer
- Message buffer

Kernel Function Example: Reading a disk block

- **Kernel Buffer Empty:** Send request to device buffer
  - **OS** puts request from kernel buffer in disk controller
  - **Interrupt handler** buffers disk block in kernel buffer
- **Kernel Buffer Full:** Read block into user space
  - Success is independent of initial requestor