CelluOS: An OS for Comparing Isolation Mechanisms

seL4 Summit 2025, Prague, Sept 3-5, 2025

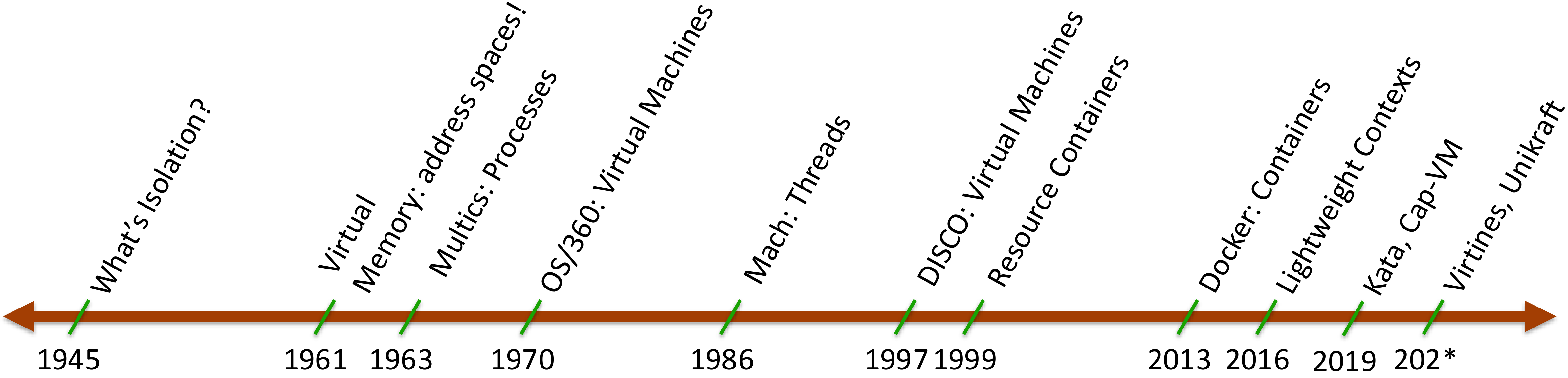
Presenter: Sid Agrawal

Students: Sid Agrawal, Arya Stevinson, Linh Pham, Ethan Xu, Shaurya Patel

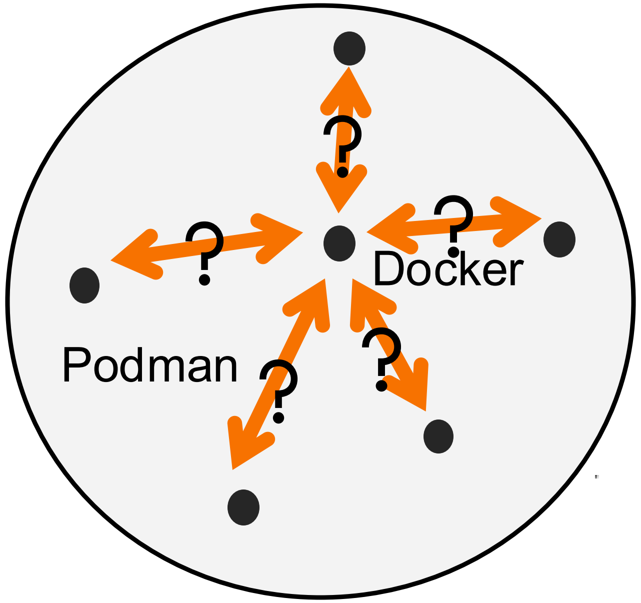
Advisors: Margo Seltzer, Reto Achermann, Aastha Mehta, Hugo Lefeuvre

Systopia Lab, University of British Columbia, Vancouver, Canada

Isolation over Time



Problems With the Current Landscape



- Hard to compare similar mechanisms in a principled way
- Too little or too much isolation
- Unintended sharing of
  - Hardware
  - Software
- Easy to misconfigure

What do we need from a model?

- Describe a system
  - Tasks depend on resources for execution.
  - Resources, in turn, depend on other resources and tasks.

- Answer some questions about tasks.
  - How hard is it for one task to affect another task?
    - Sharing resources, or pools of resources
  - How do tasks depend on each other?
    - Providing services to each other

Agenda

Goal

Enable developers and infrastructure providers to compare isolation mechanisms precisely

→

Model

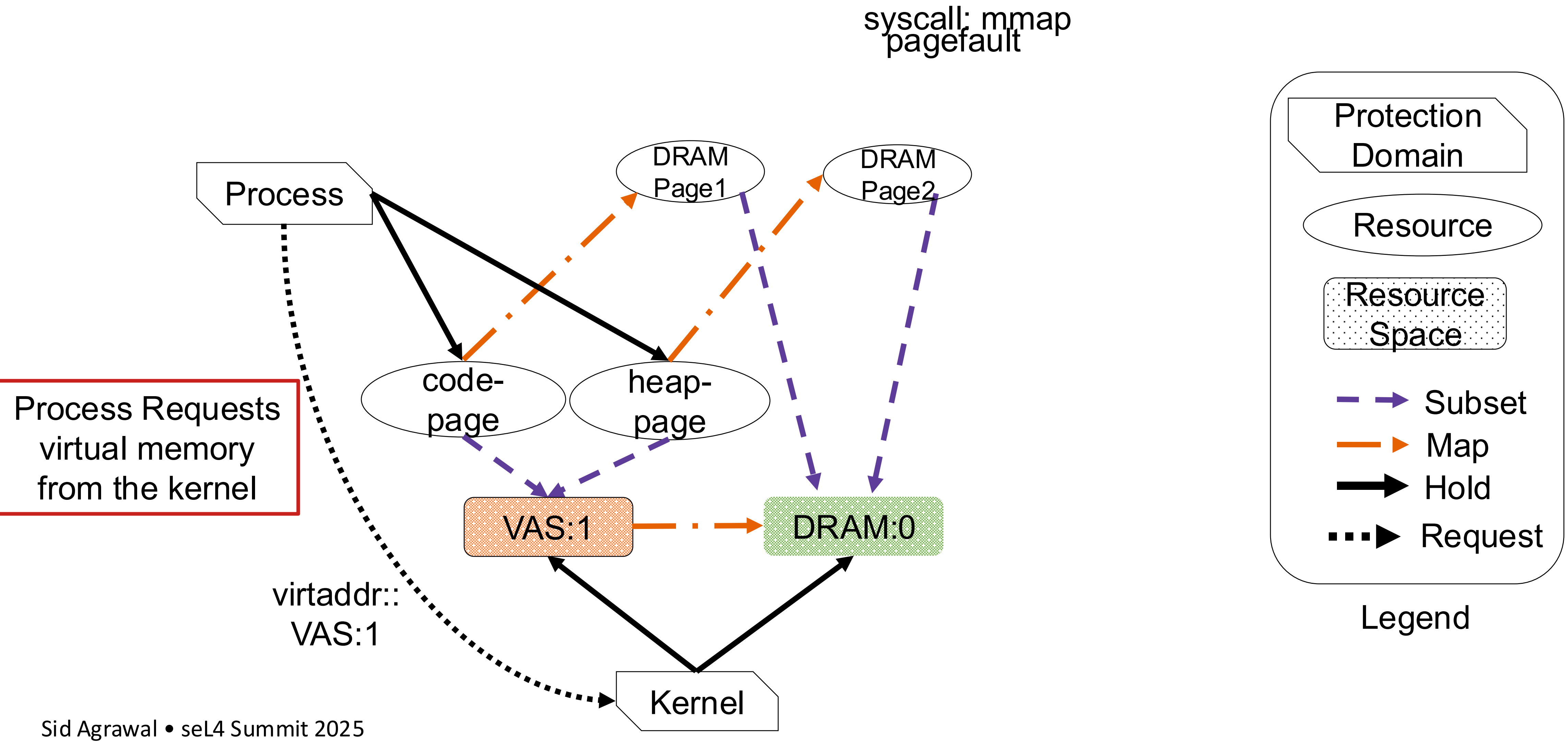
OSmosis: Formal model for reasoning about shared resources between protection domains

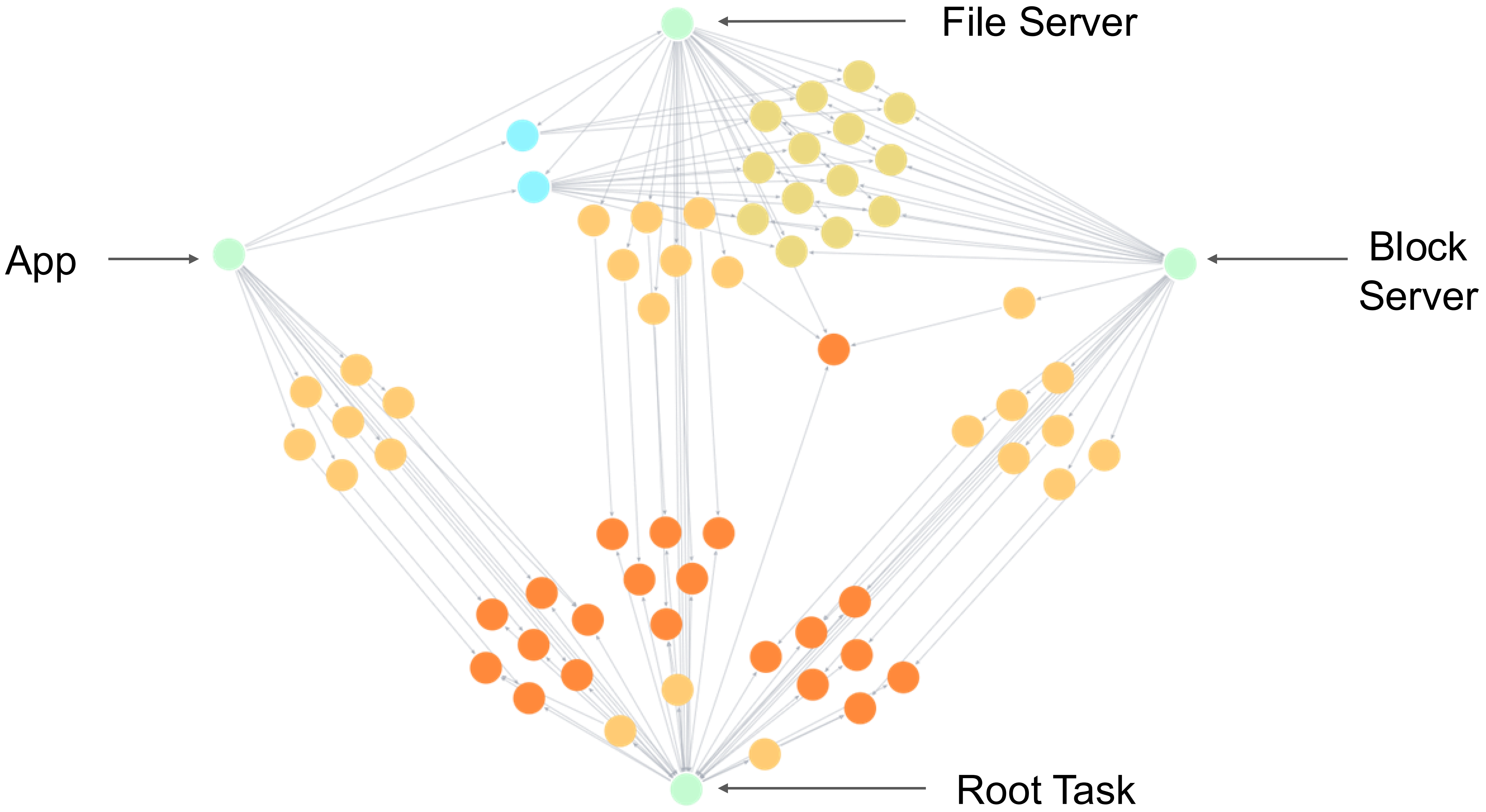
→

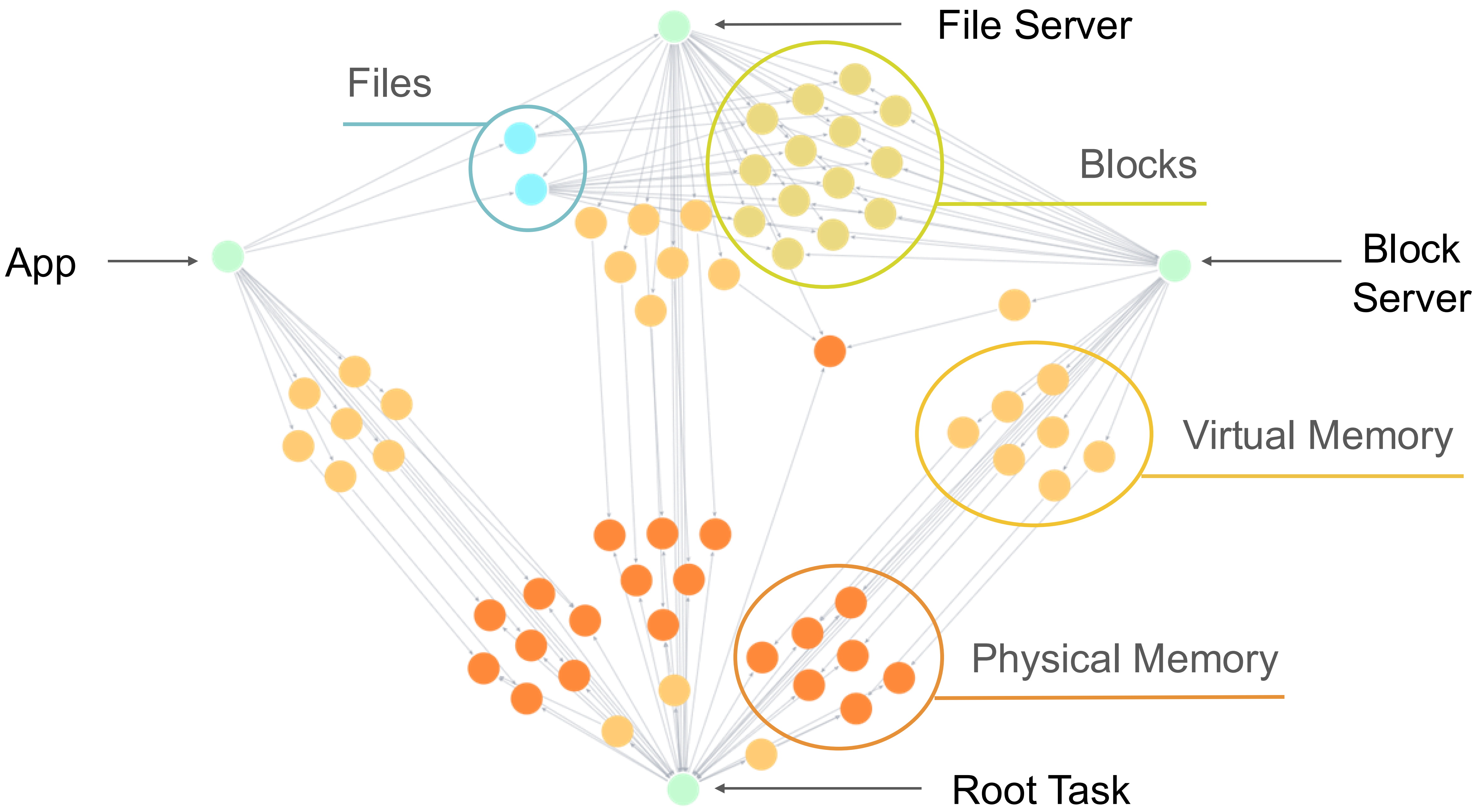
seL4 based OS

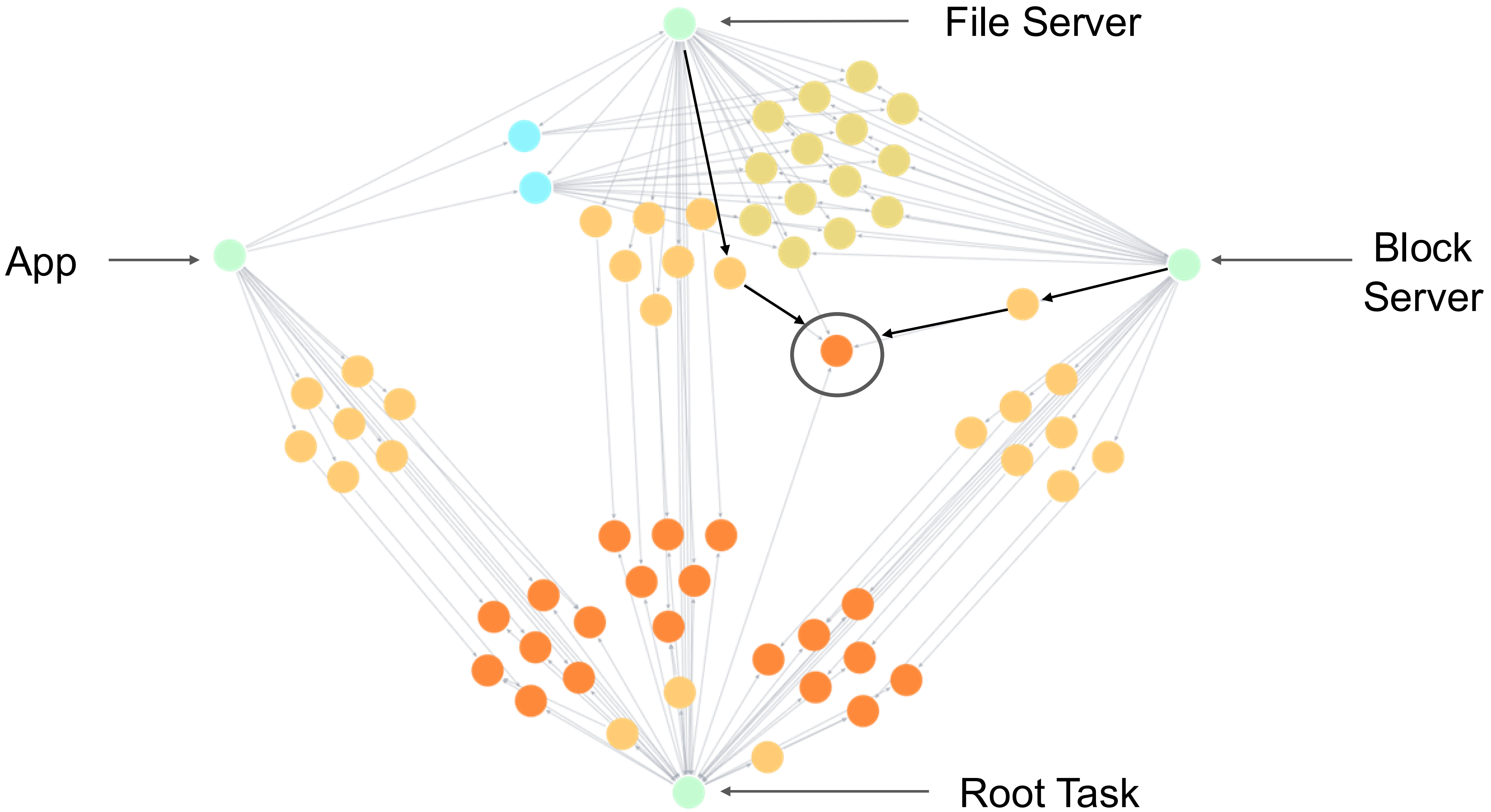
CellulOS: Custom OS that exposes protection domain state

OSmosis Model




File Server
App
Block
Server
Root Task


File Server
Files
Blocks
App
Block Server
Virtual Memory
Physical Memory
Root Task


File Server
App
Block
Server
Root Task

Understanding Domain Dependencies

Trusted Computing Base (TCB)

- A set of domains upon which a protection domain relies
- Critical for security analysis

Impact Boundary (IB)

- A set of domains affected by a faulty or malicious domain
- Essential for fault isolation assessment

TCB Definition in as a Graph Query

$$\begin{aligned}\texttt{TCB}(PD_x, types, mode) = &\ \texttt{SharedResources}(PD_x, types, mode)\\ \cup &\ \texttt{SharedResourceSpaces}(PD_x, types)\\ \cup &\ \texttt{ResourceSpaceServers}(PD_x)\\ \cup &\ \texttt{SharedResourceSpaceServers}(PD_x)\\ \cup &\ \texttt{CanControl}(PD_x)\end{aligned}$$

Shared Resources can affect ***Confidentiality***, ***Integrity***, and ***Availability***. E.g., shared memory, file.

Shared Resources Spaces can affect ***Availability***. E.g., cgroups

Shared Resources Servers can affect ***Availability***. E.g., file-server

PD that control a given PD can affect its ***Availability***

Resource Space Servers provide Resources and can affect ***Confidentiality***, ***Integrity***, and ***Availability***. E.g., file-server

Finding Dependencies: Shared Resources

$\texttt{SharedResources}(PD_x,\ \texttt{ResourceTypes},\ \texttt{AccessMode}) =$

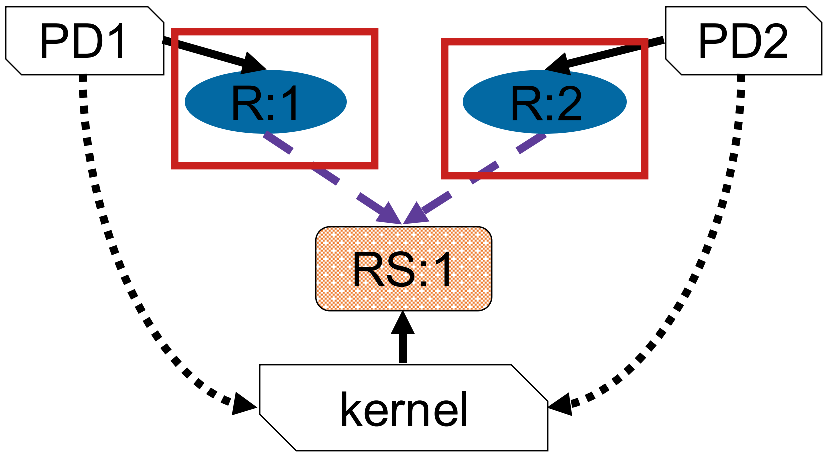
$\{\ PD_i\ \mid\ PD_i \neq PD_x \wedge ($

$BFS(PD_x,\ \{\texttt{hold},\ \texttt{map}\},\ \texttt{fwd},\ \texttt{ANY},\ \infty,\ \{\texttt{Resource}\},\ \texttt{ResourceTypes})$

$\cap$

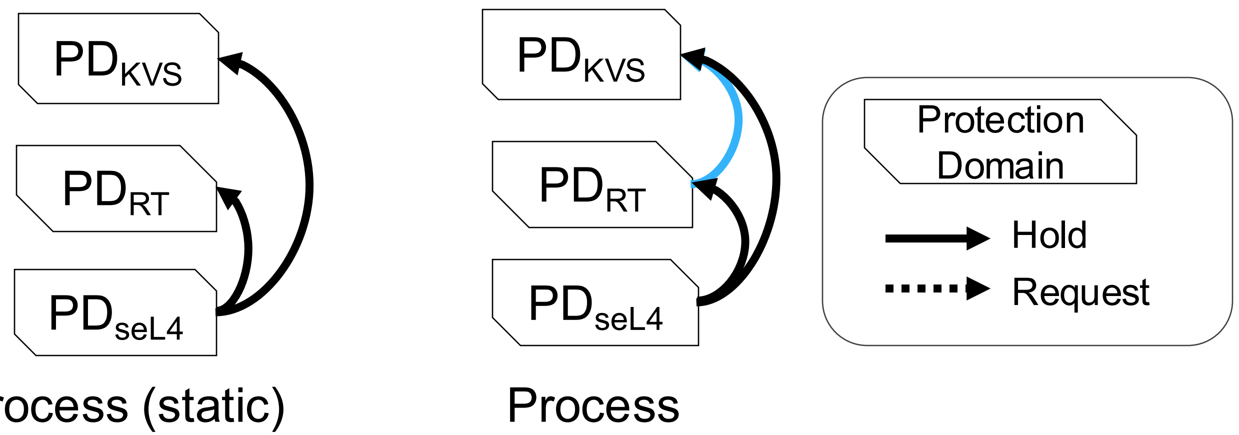
$BFS(PD_i,\ \{\texttt{hold},\ \texttt{map}\},\ \texttt{fwd},\ \texttt{AccessMode},\ \infty,\ \{\texttt{Resource}\},\ \texttt{ResourceTypes})$

$\neq \emptyset\ )\ \}$



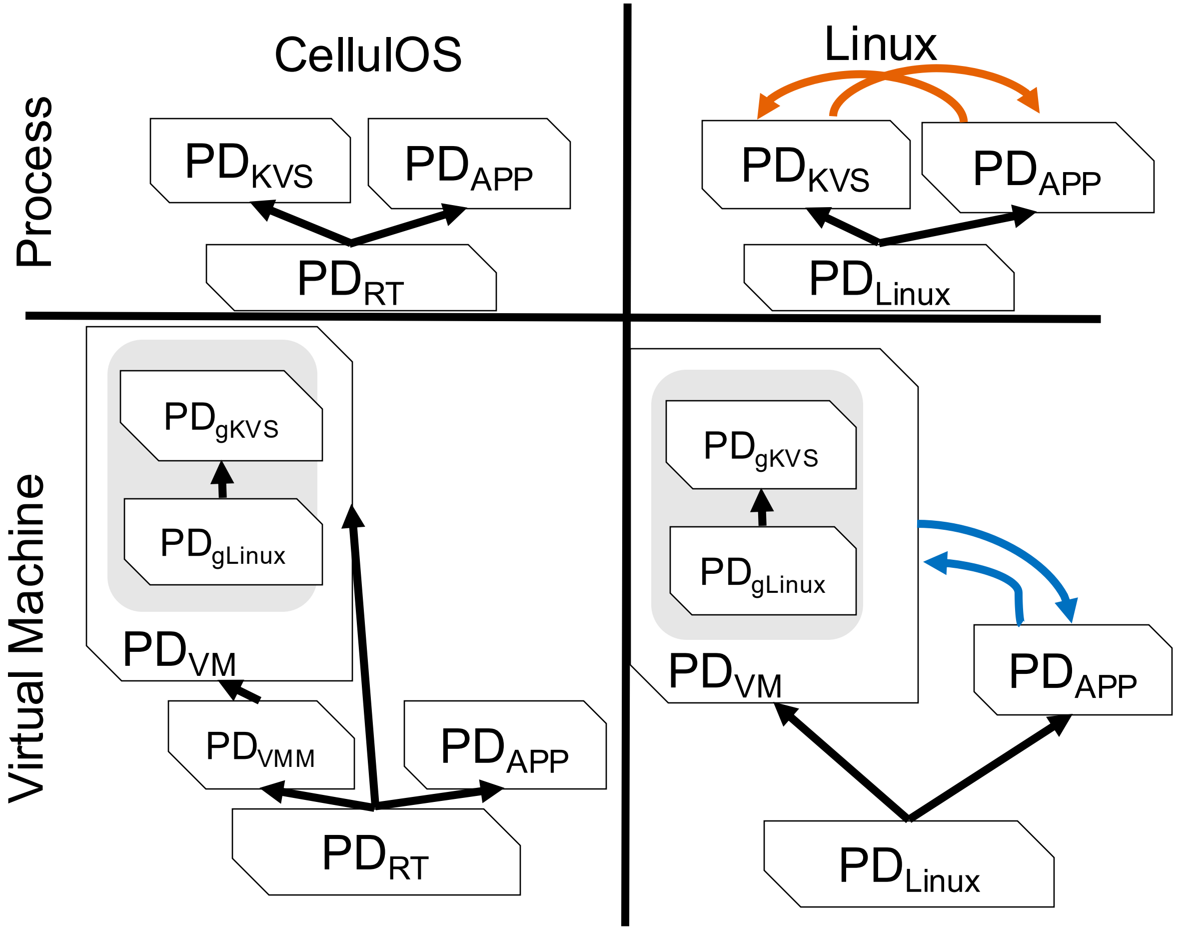
Visualizing Domain Dependencies (2/3)

Which protection domains are in the TCB?



Query	Process (static)	Process
CanControl(PD_{KVS})	PD_{seL4}	PD_{RS} PD_{seL4}

Visualizing Domain Dependencies (3/3)



Which protection domains are in the IB?

Query	CellulOS	Linux
CanControl(PD_{KVS})	PD_{RT}	PD_{App}, PD_{Linux}
ControlBy(PD_{KVS})	∅	PD_{App}
CanControl(PD_{VM})	$PD_{RT}PD_{VMM}$	PD_{App}, PD_{Linux}
ControlBy(PD_{VM})	∅	PD_{App}
CanControl(PD_{gKVS})	PD_{gLinux}	PD_{gLinux}
ControlBy(PD_{gKVS})	∅	∅

CellulOS

Goals

- Build an OS that makes it easy to track and extract the model state

Why a capability-based microkernel?

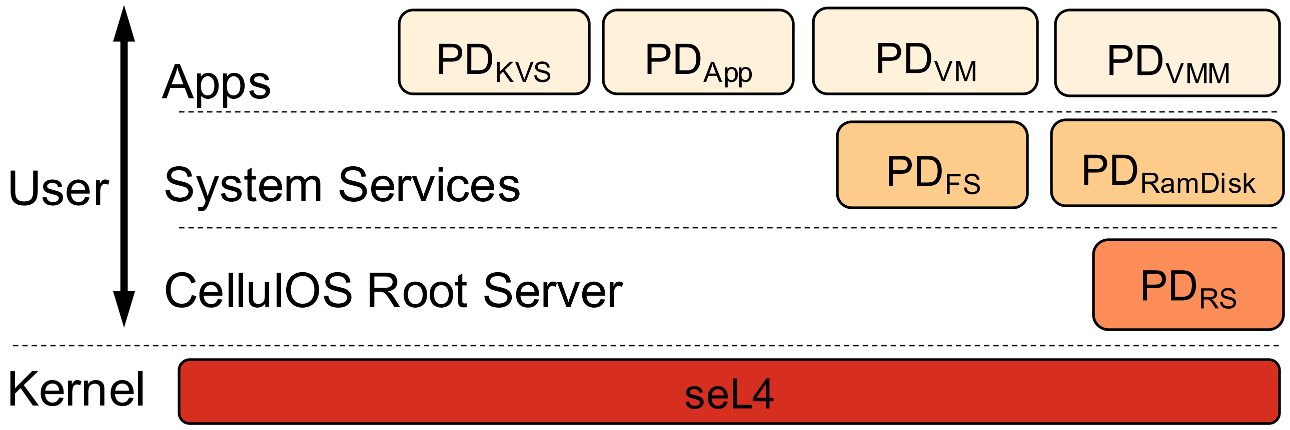
- **Explicit Resource Management:** All resources represented as capabilities
- **Fine-grained Control:** Precise specification of what each component can access
- **Best documented µKernel:** Only getting better

CellulOS Architecture

Root Task
- Starts System Services
- Based on sel4test
- Tracks every capability

System Services
- Provide Resources: file, blocks
- Export internal model state

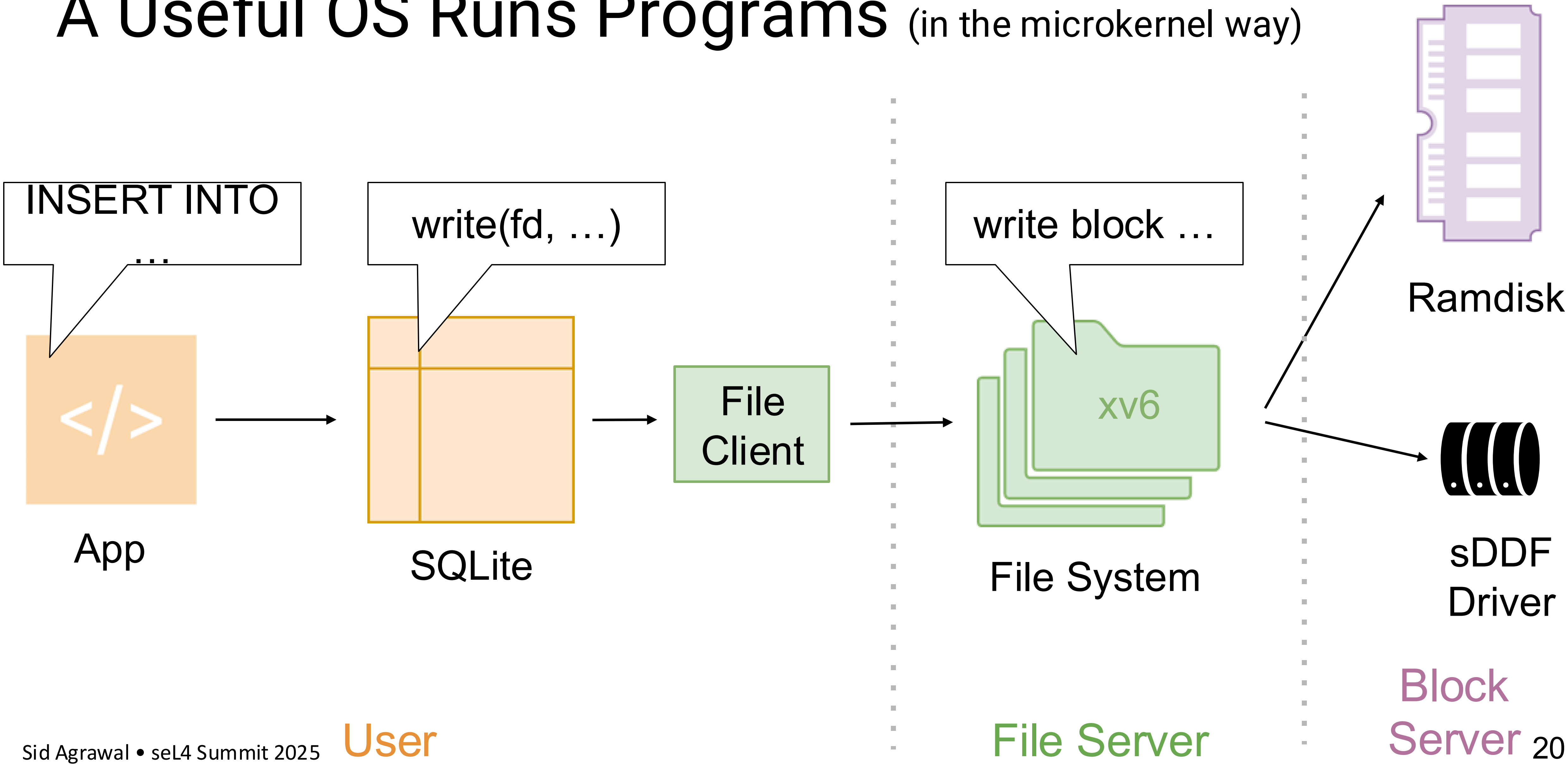
Apps
- Key Value Store server/client
- Virtual Machines



CelluOS Workflow

- **Deploy a Scenario**
  - Get to a steady state
- **Runtime Tracking**
  - Monitor resource allocation and dependencies
- **State Extraction**
  - Collect protection domain relationships
- **Graph Generation**
  - Build the OSmosis model representation in a GraphDB (such as Neo4J, network)
- **Visualization and Querying**
  - Query the graph

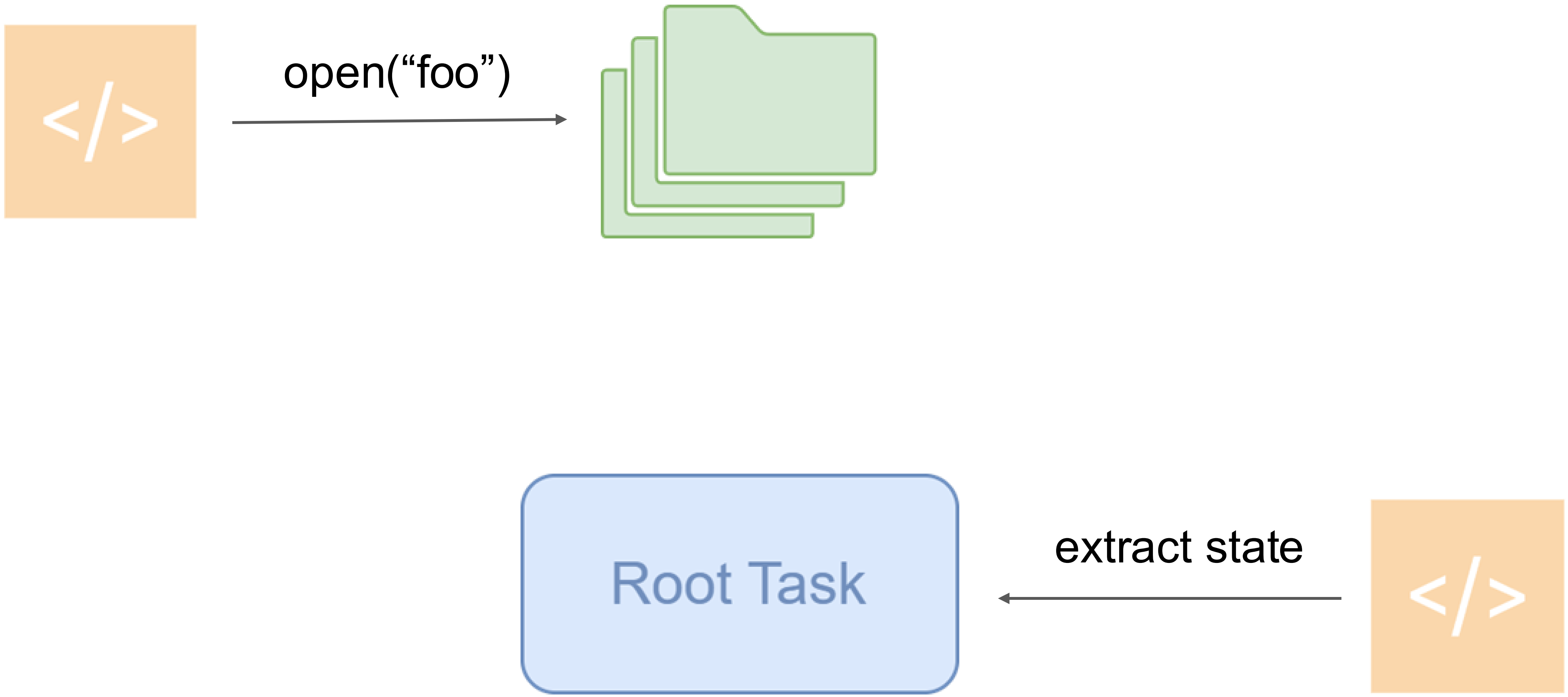
A Useful OS Runs Programs (in the microkernel way)



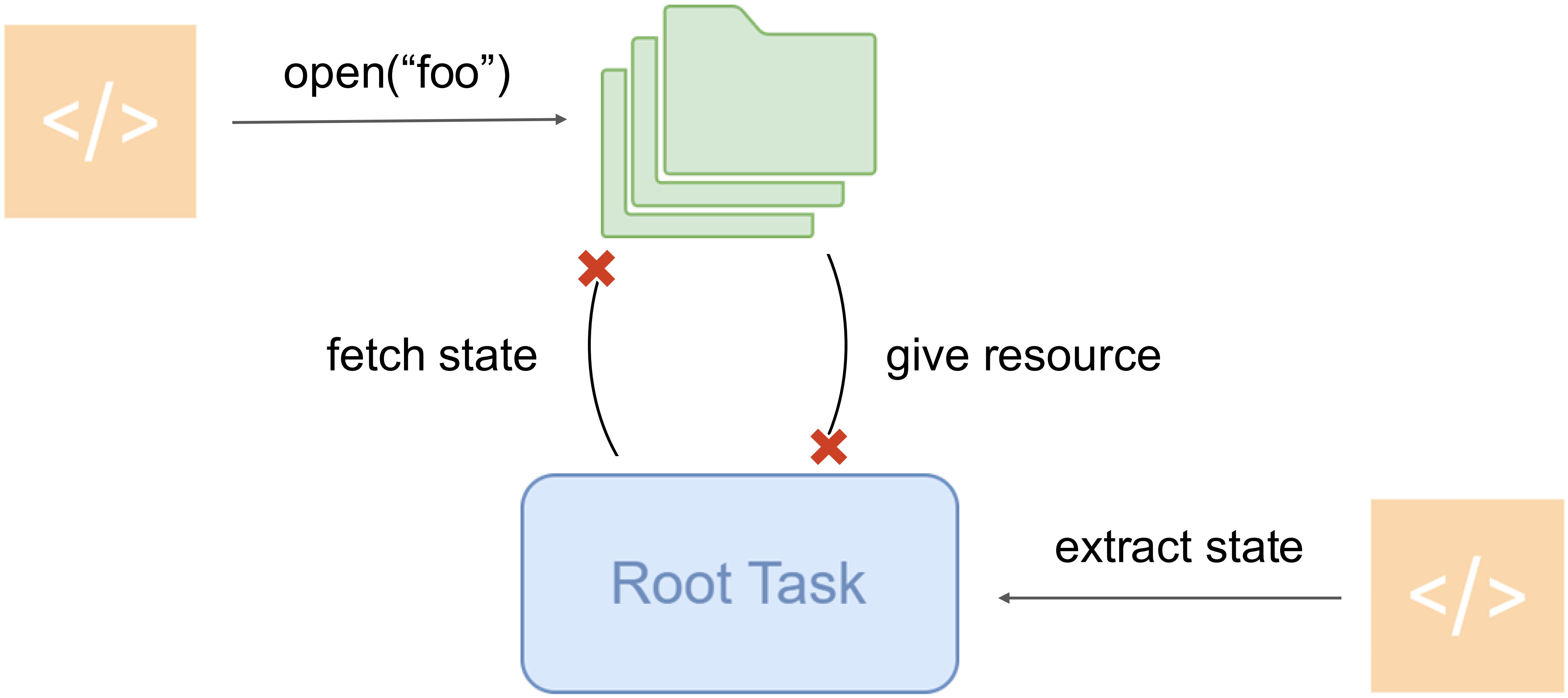
Resource Tracking

- Resources are ***badged endpoint*** capabilities
  - 64-bit badge: {Type, ResID, ServerID, OwnerID}
  - Root Task tracks every resource
- Root Task doesn’t have all edges
  - Resource Server tracks certain map edges
  - Virtual Machine PD tracks the OSmosis state for the guest
- Model state extraction has some challenges
  - Deadlock (next)

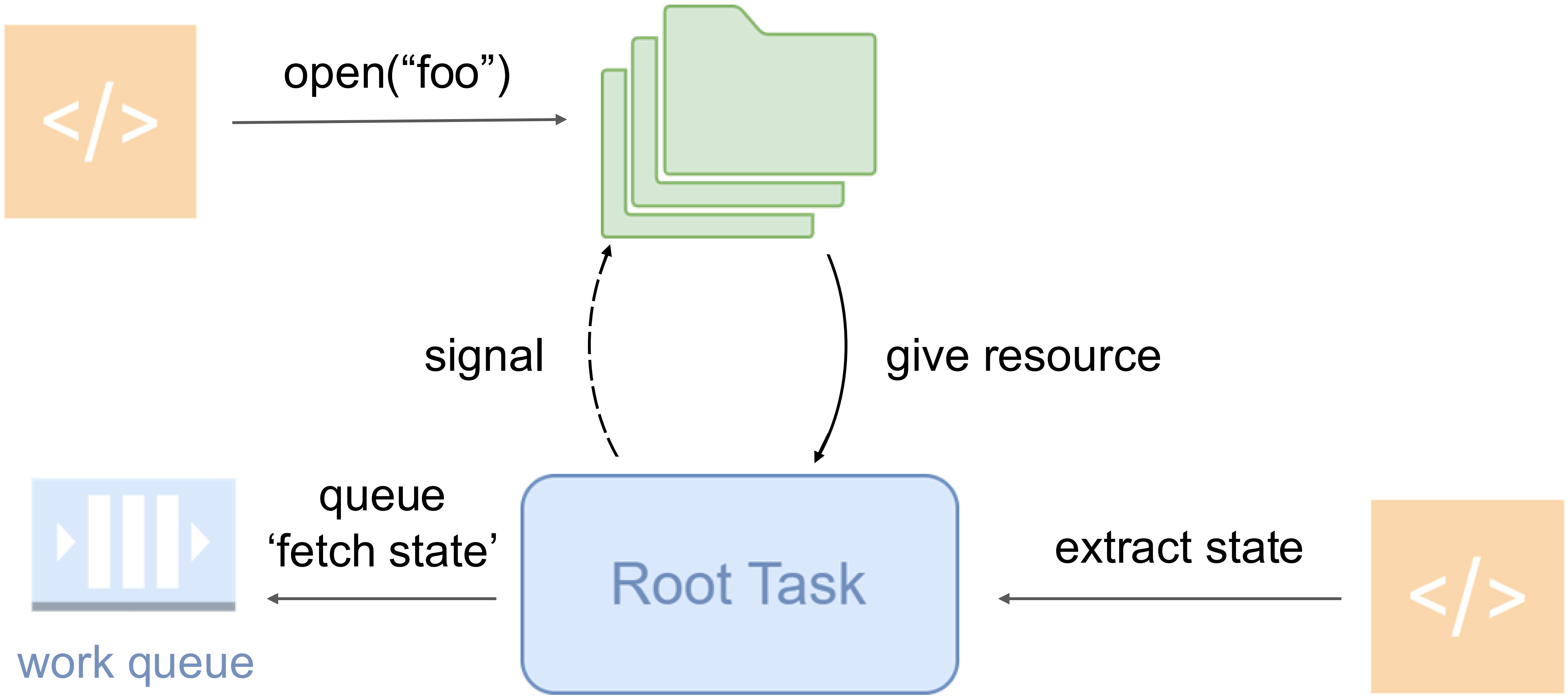
An Interesting Challenge: Communication & Deadlock



An Interesting Challenge: Communication & Deadlock



An Interesting Challenge: Communication & Deadlock



Virtual Machines as Protection Domains

- **Goal**
  - Connect OSmosis state from inside the guest (Linux) to the host (CellulOS)
- **Challenges**
  - Linux has no central place to track resources
- **Approach**
  - Gobble whatever info available in /proc

Mechanism	Kappa Modeling Approach	Key Details / Examples
Cgroups	Memory quotas as hierarchical resource spaces; PDs with hold edges to page quota resource	Doesn't capture over subscription.
Namespaces	Define hold-edges & permissions (not distinct resources).	Mount ns: access to host dir. PID ns: control other PDs User ns: controls effective userID
Syscalls	Request-edges (PD->Kernel) or attributes on hold-edges.	Models direct kernel requests or actions via resources.
Seccomp	Allow or block syscall.	E.g., prohibiting write.
Linux Capabilities	Request-edge or permission on hold-edge based on the syscall.	E.g., CAP_SYS_BOOT ('terminate' perm. on hold edge to kernel), CAP_AUDIT_WRITE ('write' permission on hold-edge to audit log file).

Bootstrapping an seL4 Project

Our Background

- Small group of graduate student(s), new to seL4

Challenges

- Learning capability-based programming model
- Understanding microkernel service decomposition
- Debugging distributed system interactions

Community Support

- Extensive help from the seL4 ecosystem
- Drivers and VMM from Microkit

Useful coursework

- Advanced OS class at UBC based on Barrellfish taught by Prof. Achermann
- Advanced OS class at UNSW based on seL4

Summary

- Implementing the Root Task
- RPC mechanism using protobuf
- Resource cleanup
  - Flexible cleanup policies
- OSmosis+CellulOS model state workflow
  - Neo4j scripts & docker container
  - Metrics calculations
- Model state extraction from Linux /proc
- Porting: SQLite, VMM, Drivers
- Lines of code:
  - C: 58k
  - Python: 2k

Programming for a Microkernel

Programming for Linux

Questions ?

Key Takeaway

- **Formal models** enable precise isolation comparison
- **seL4's capability system** facilitates accurate resource tracking
- **Microkernel architecture** simplifies protection domain decomposition

Documentation

- **Wiki:** https://cellulosdocs.readthedocs.io/en/cellulos/
- **Source Code:** https://github.com/sid-agrawal/osmosis
- **Research Papers:** <TBD>

Contact

- sid@sid-agrawal.ca