seL4 and BEAM: a match made in Erlang

Ihor Kuz

Kry10 Limited

seL4 Summit Oct 2022

Kry10 goals

Security

Robust in the face of attacks
Protection at ***all*** levels
(hardware, software, services)

Resilience

Robust in the face of faults
Self-healing, Fast Recovery, Minimize downtime

Usability

Simple development environment
Trusted software libraries
A pleasure to use

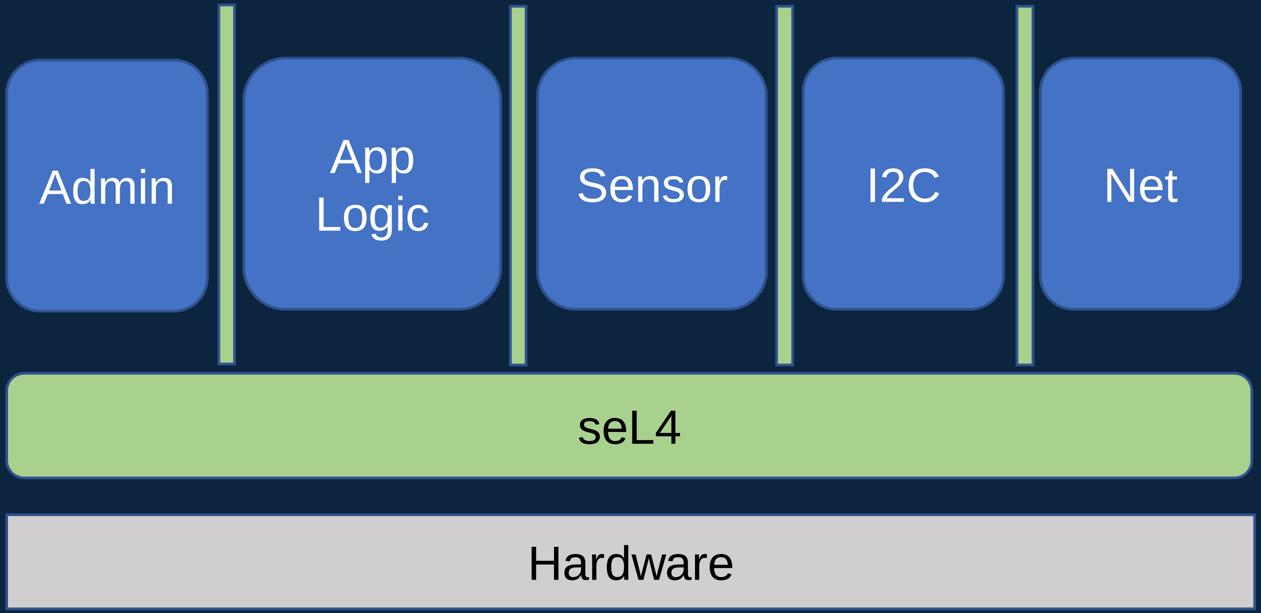
Security and Resilience

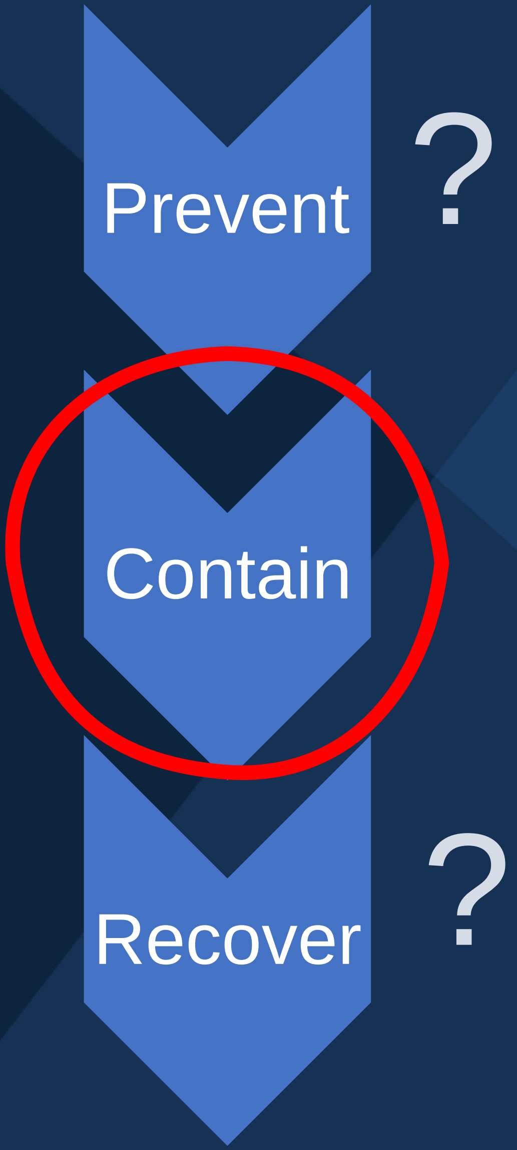
Prevent
- Vulnerabilities from being exploited
- Faults from occurring

Contain
- Stop attacks from propagating
- Stop faults from propagating

Recover
- Restart exploited or faulty components
- Fix the problems

seL4


Admin
App Logic
Sensor
I2C
Net
seL4
Hardware


Prevent
?
Contain
Recover
?

BEAM, Erlang, Elixir

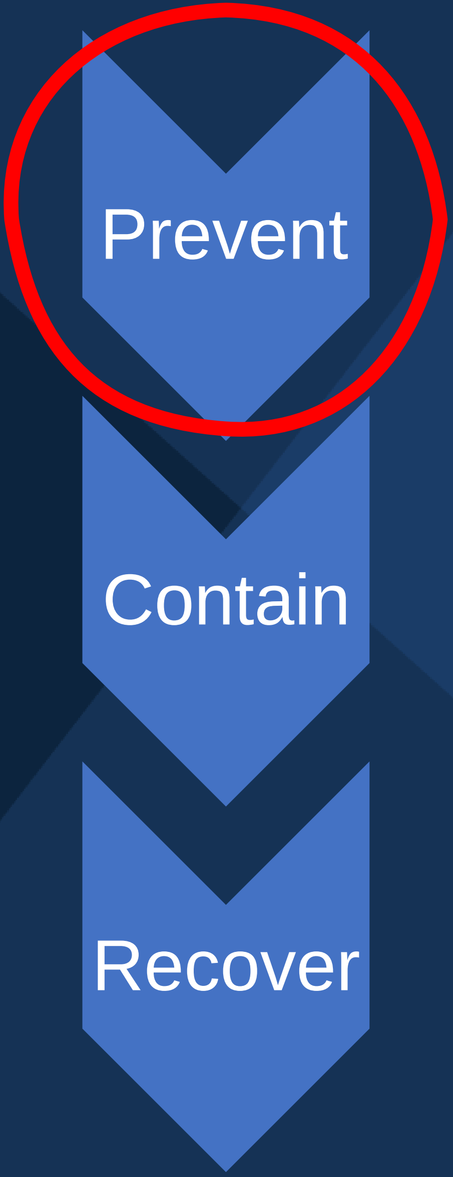
- BEAM community
  - Meetups
  - Conferences
  - Development companies
- Popular language
  - Elixir: 2nd most "loved" language (after Rust)[1]
- Popular frameworks
  - Phoenix (web framework): most "loved" framework[1]

[illegible]
- [illegible] functional programming
- [illegible] tolerant,
- [illegible]
- [illegible]
- [illegible]
- Transparent distribution model
- Standard library
  - 3rd party: frameworks, packages
  - Dependency management

1: https://survey.stackoverflow.co/2022

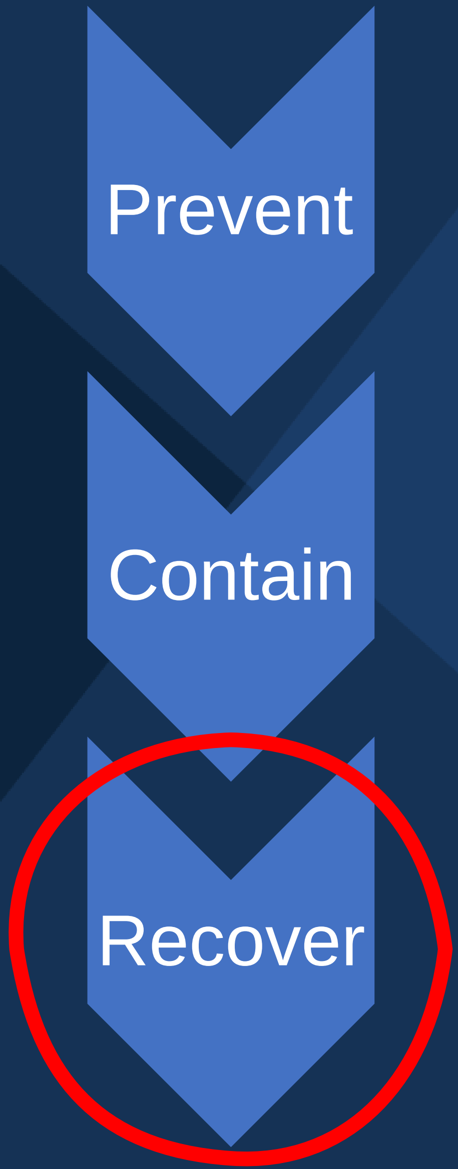
BEAM: “prevent”

- Prevent faulty software within seL4 protection domains
- Language support
  - Elixir, frameworks, libraries, dependency management, tools
- Concurrency
  - Model for concurrency within seL4 protection domain
- Communication/distribution model
  - Model for communication between seL4 protection domains
  - Frameworks for scalable and reliable distribution between seL4 protection domains
- Isolation
  - Memory isolation within seL4 protection domain
  - No shared memory between processes



BEAM: “recover”

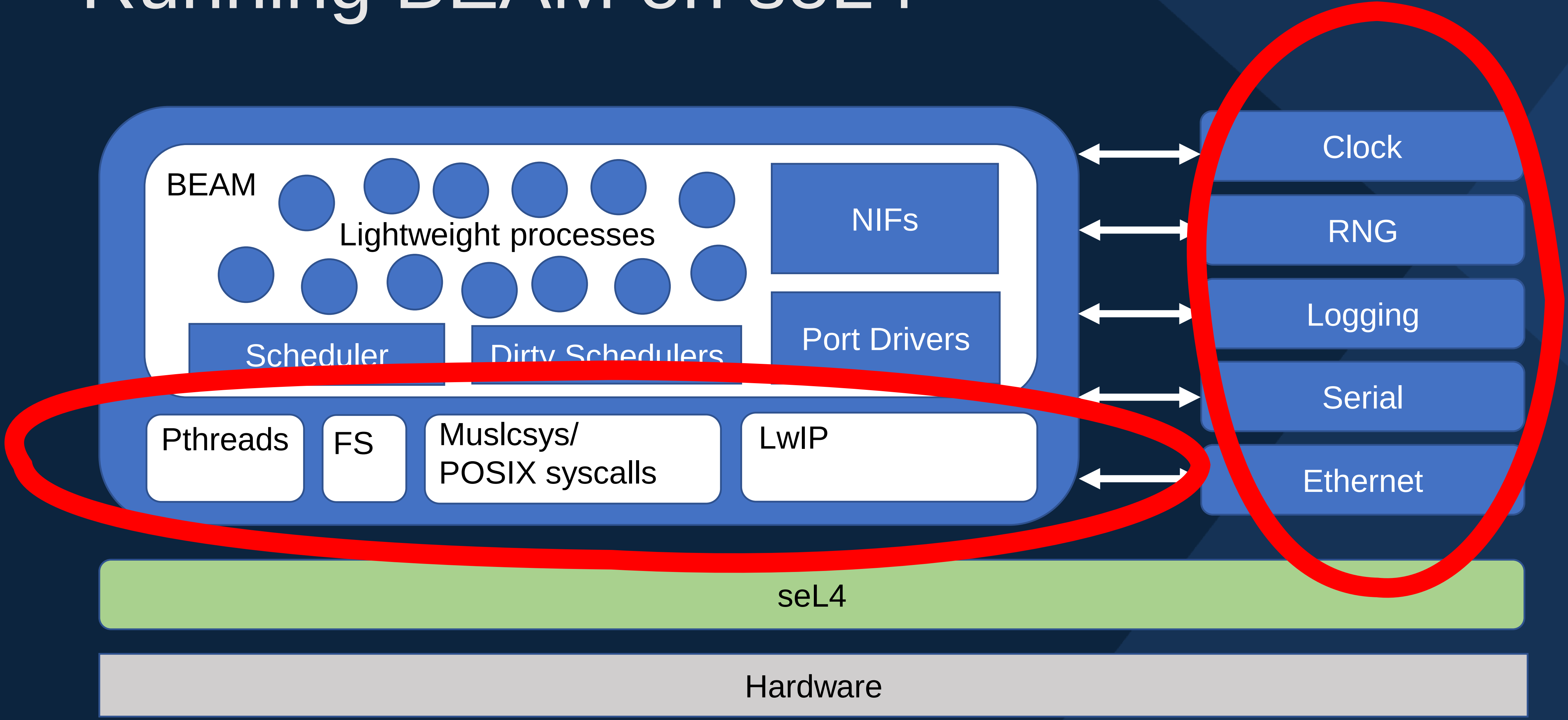
- Process independence
  - Processes don’t share memory
  - Crashing process cannot corrupt other processes
- Error and crash detection
  - Linking and monitoring
  - Exit process on error
  - Detect exited processes
- Supervision trees
  - Mechanism and policy to detect and restart failed processes
  - “Crash and restart” model for processes



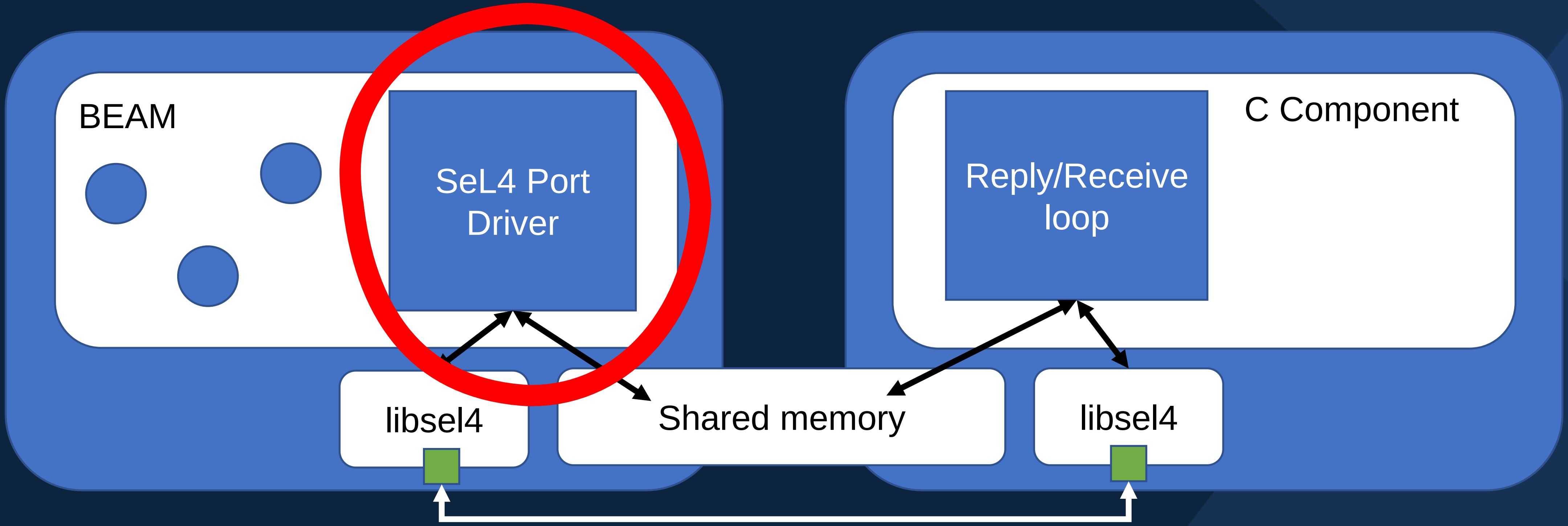
seL4 + BEAM: Challenges

- Run BEAM on seL4
- Communicate outside BEAM
  - BEAM to native components
  - BEAM to BEAM
- Elixir-based development environment for seL4-based systems
  - Develop app (business) logic in Elixir
  - Use existing Elixir/Erlang libraries and frameworks
  - Elixir tools for setup, build, deploy
  - Only need C, Rust, CMake for low-level elements

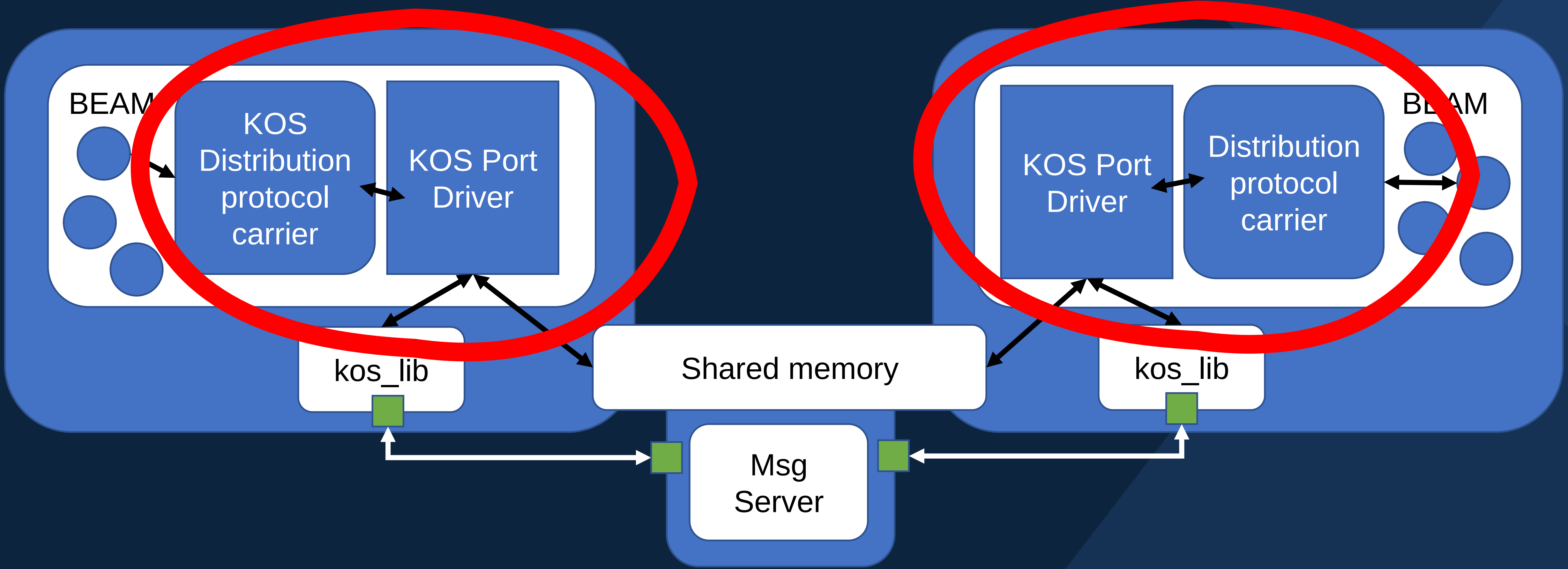
Running BEAM on seL4



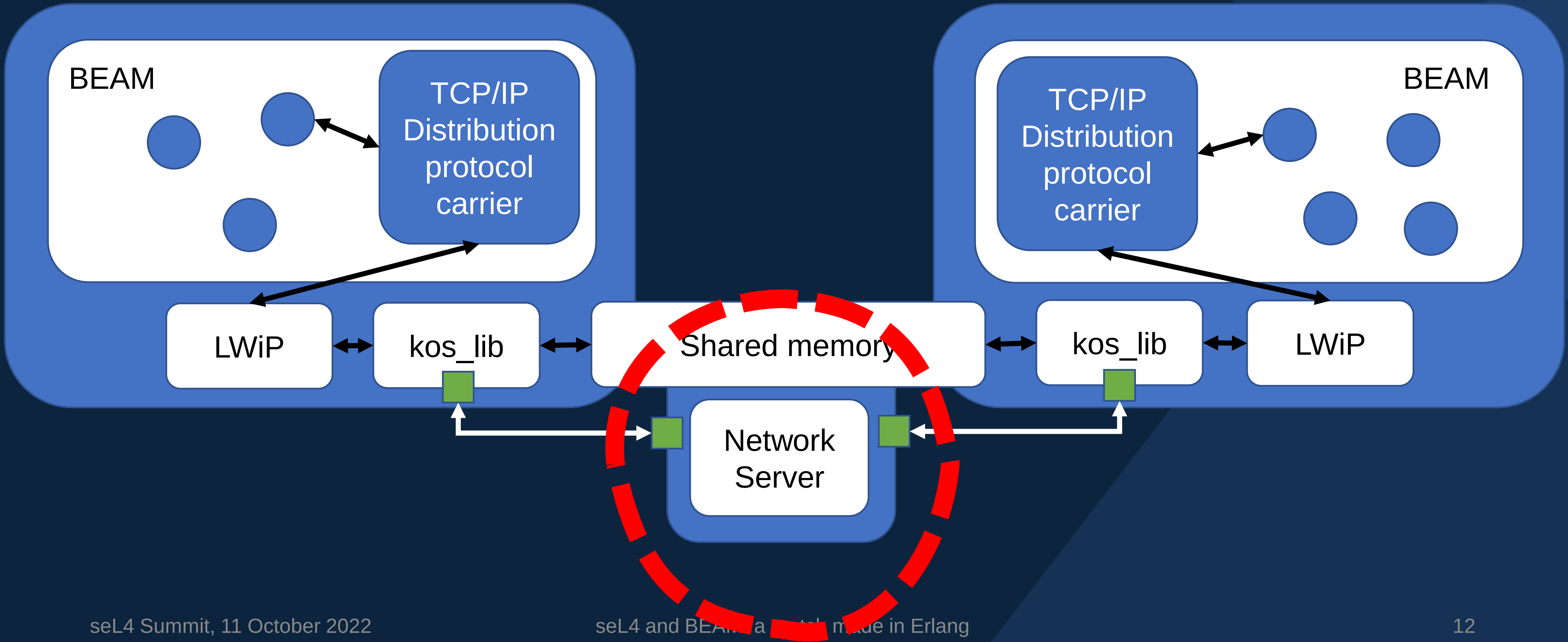
Communication (seL4 IPC)



Communication (BEAM 2 BEAM)



Communication (BEAM 2 BEAM)


BEAM
TCP/IP Distribution protocol carrier
LWiP
kos_lib
Shared memory
Network Server
kos_lib
LWiP
TCP/IP Distribution protocol carrier
BEAM

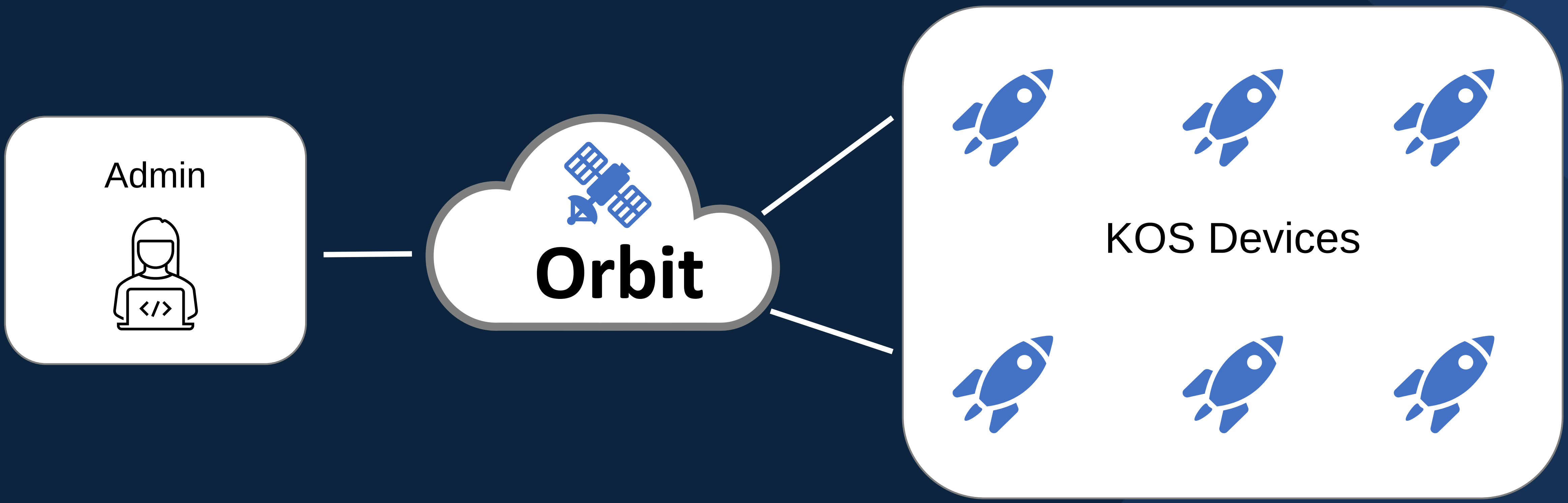
Building apps with Elixir

- Hex: Erlang/Elixir package manager
  - Most work out of the box on BEAM on seL4
- TCP/IP: gen_tcp, HTTP client libs, etc.
- Web Framework: Phoenix
- UI: Scenic
- Drivers: Circuits
- Iex: interactive elixir shell
  - Toolshed: system commands for iex

Developing for seL4 within Elixir

- Develop application logic for a system using only Elixir
  - (and Elixir’s tools, packages, etc.)
- Mix
  - Elixir build tool
  - Mix new, mix compile, mix run
- Mix for KOS
  - Extend mix with KOS module
  - Mix kos.new
  - Mix kos.new_app
  - Mix kos.build
  - Mix kos.run

Orbit – the server side



Demo

Summary

- Building secure and resilient systems:
  - Prevent, Contain, Recover
- seL4 provides “contain”
- BEAM + Erlang/Elixir provide “prevent” and “recover”
- BEAM runs on seL4
  - Supported by partial POSIX syscalls and library
  - Added code to do seL4-based IPC
- Can do full system development from Elixir