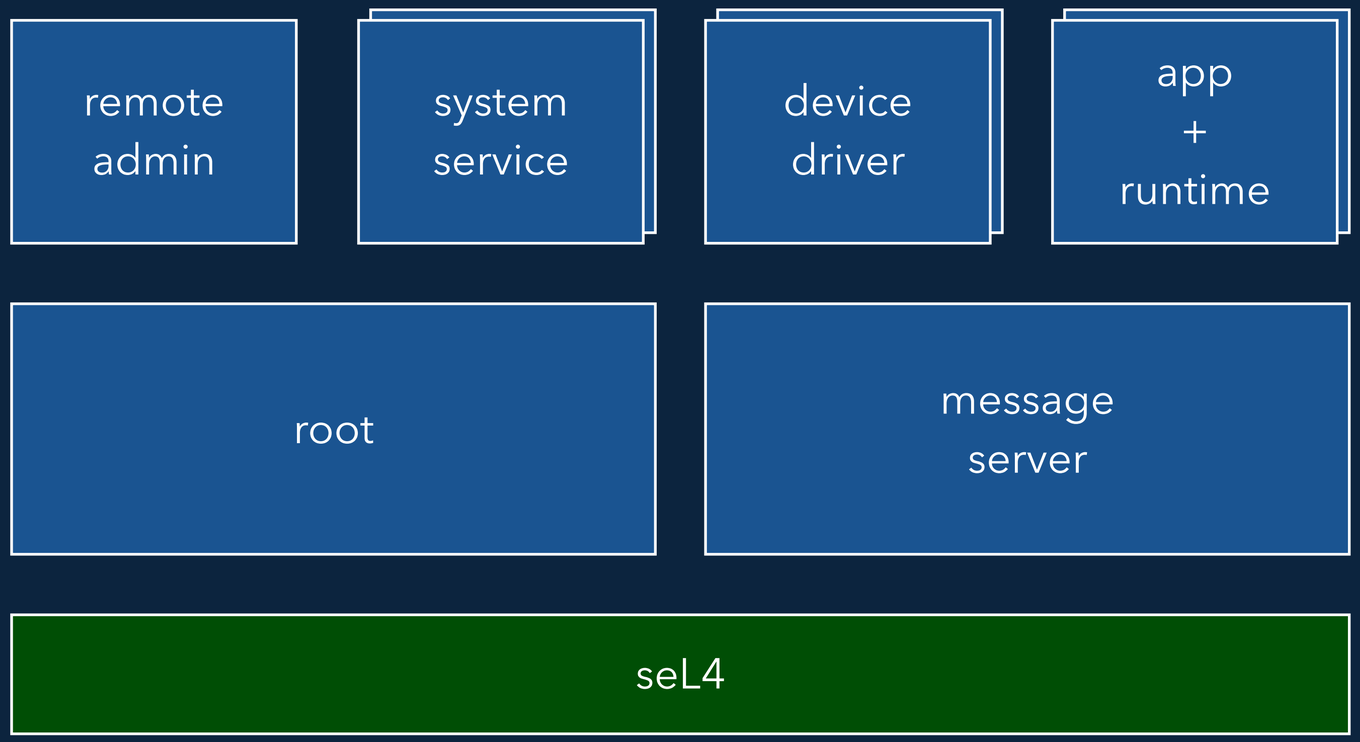
Dynamic seL4-based systems: designing for verification

Matt Brecknell

Verification Lead – Kry10

seL4 Summit 2023 – Minneapolis

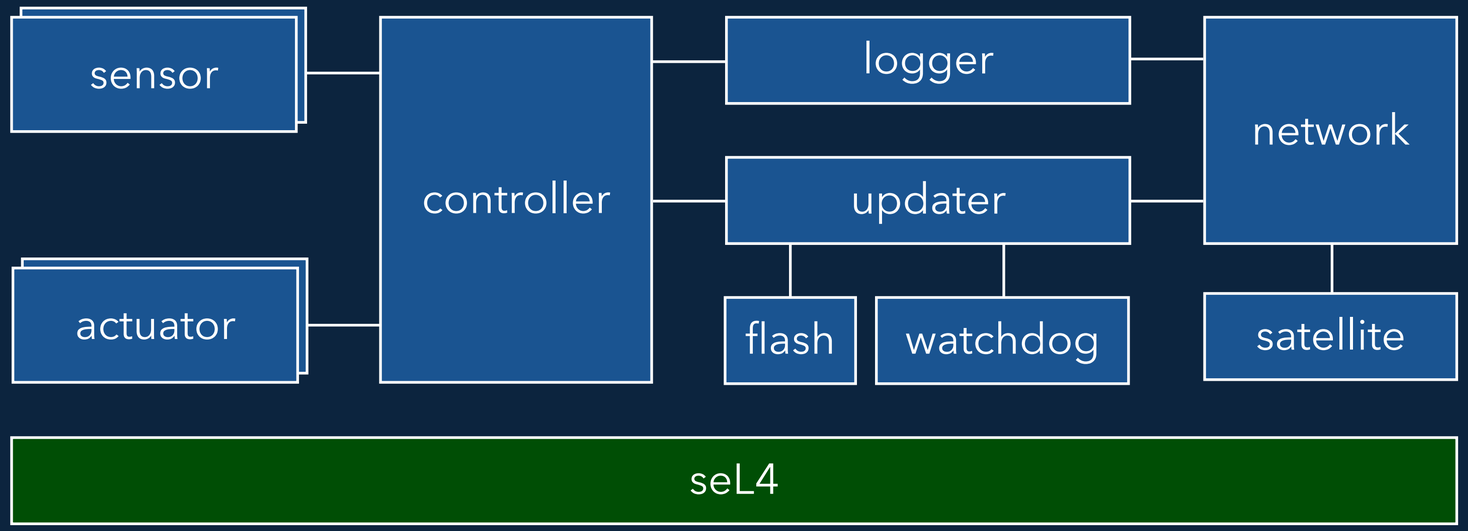
Kry10 Operating System



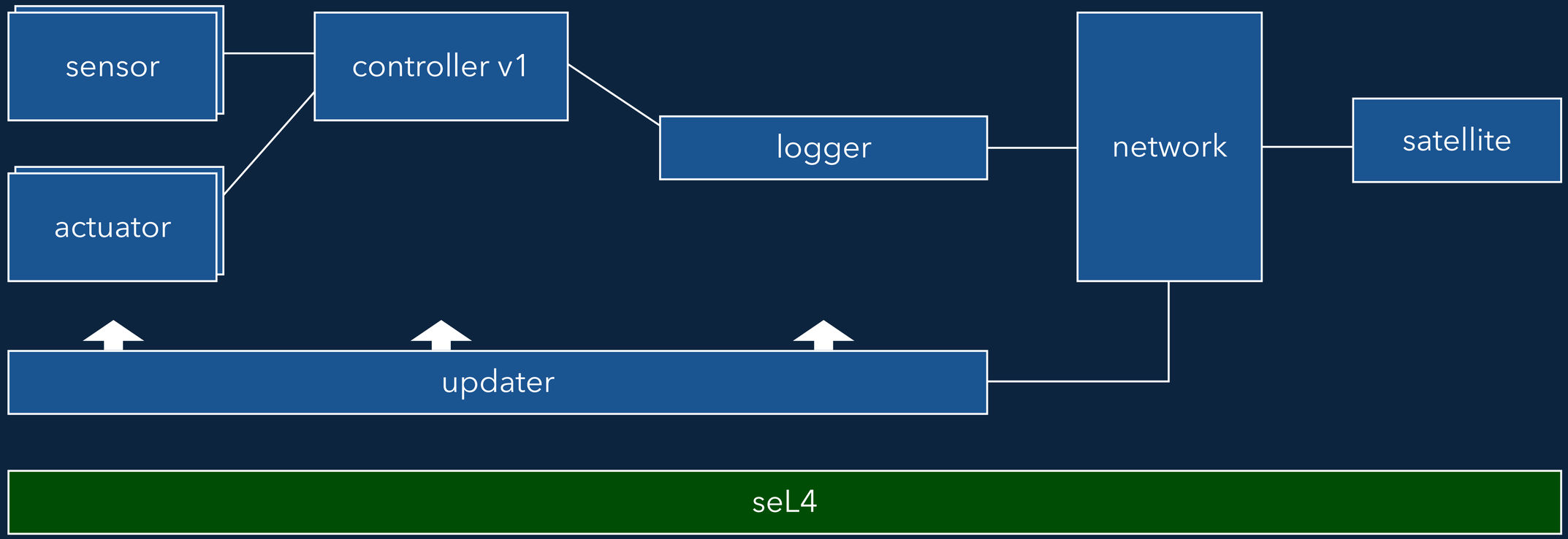
security
reliability
robustness
} resilience

manageability

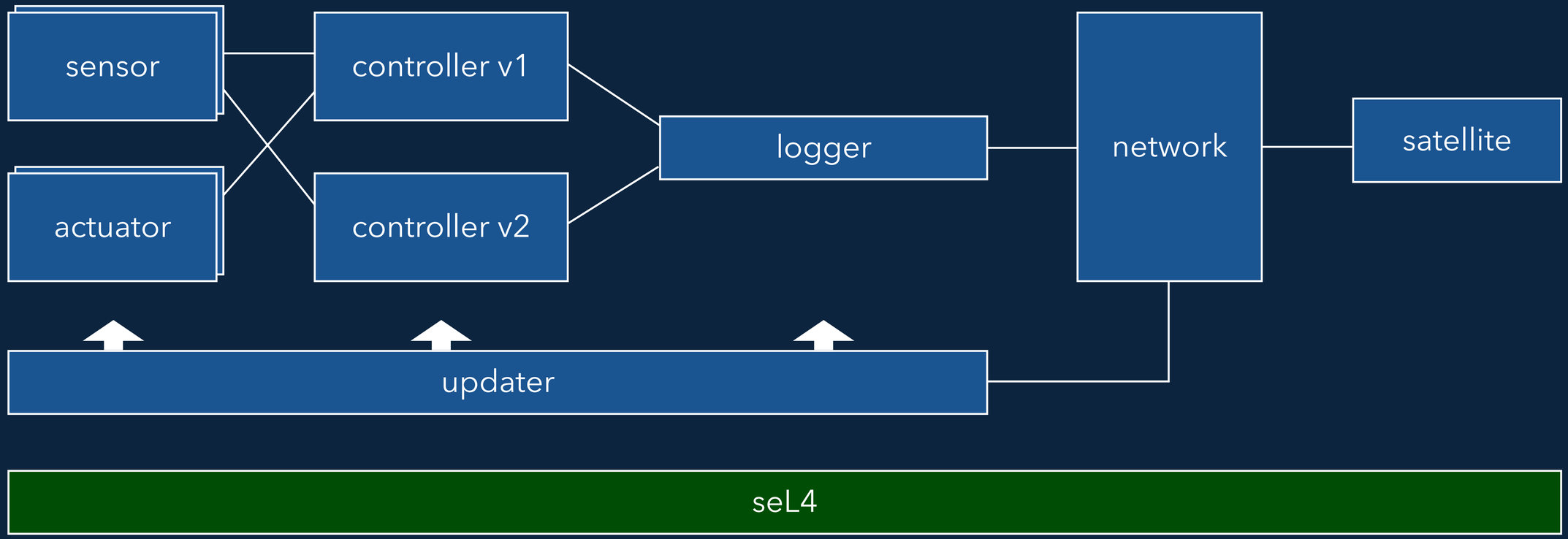
Story - chapter one



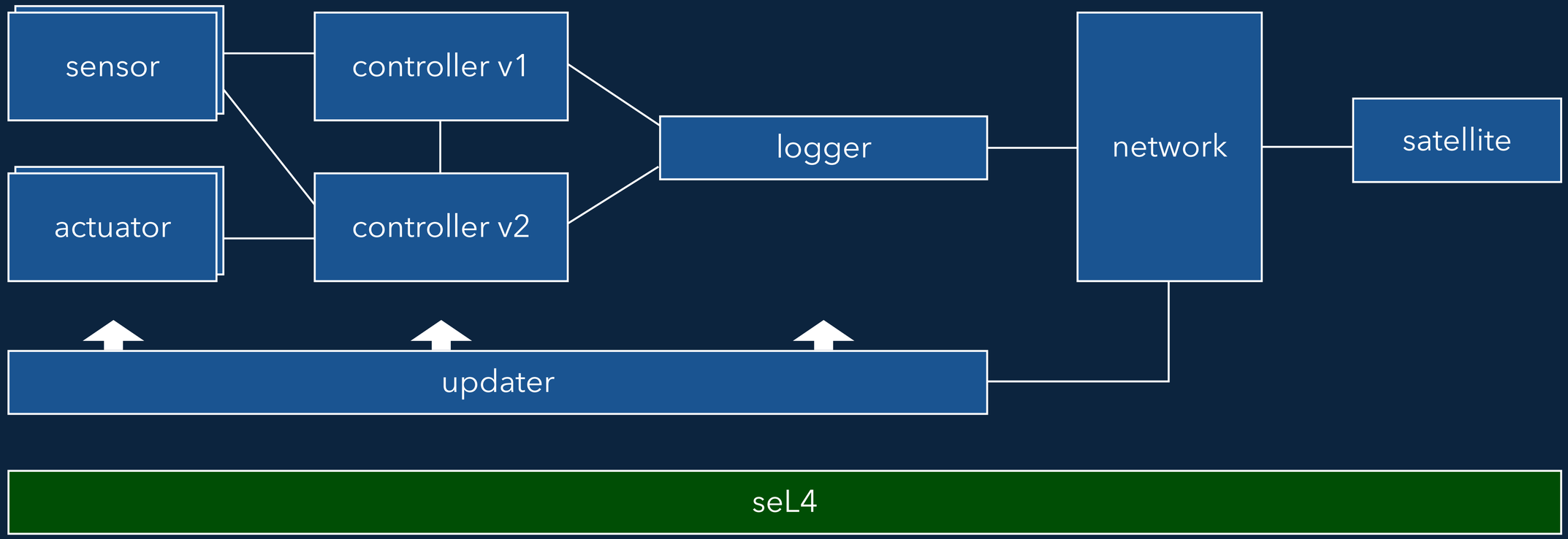
Story - chapter two



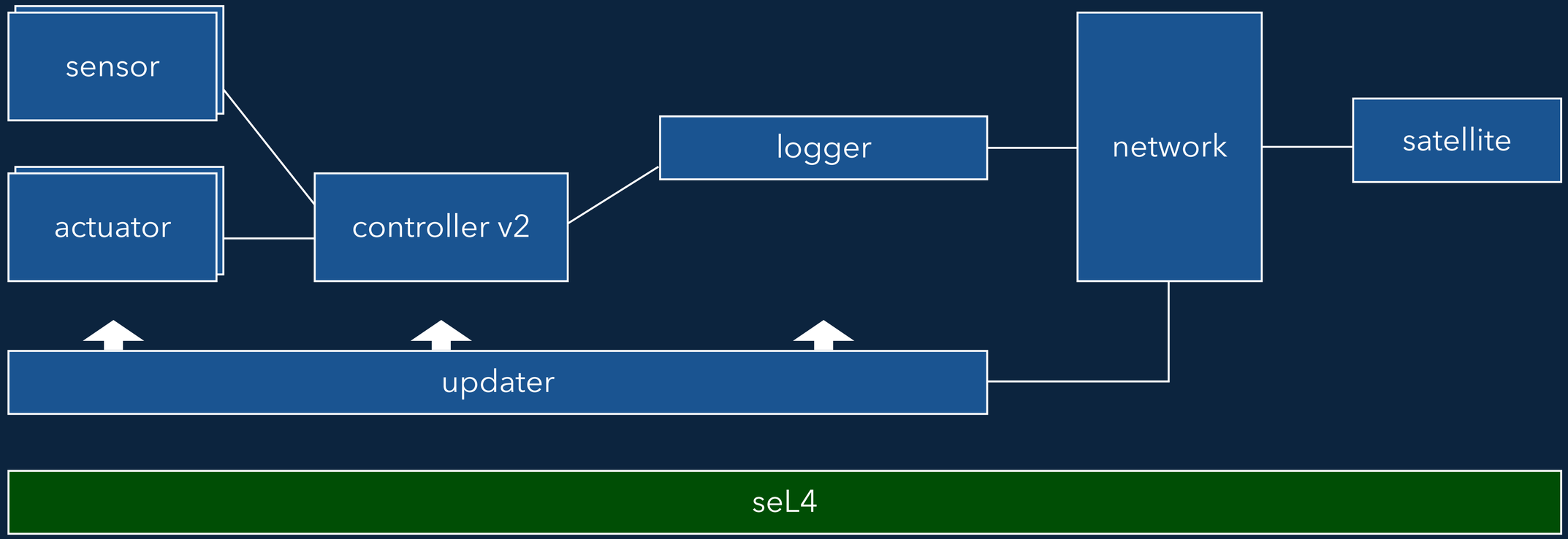
Story - chapter two



Story - chapter two



Story - chapter two

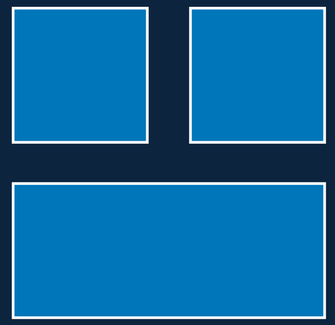


Dynamic behaviours

1 Add and remove components

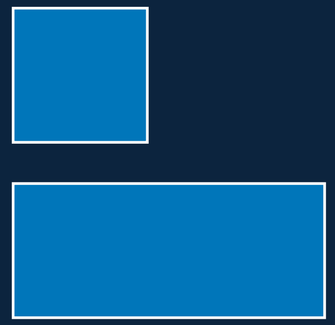
Dynamic behaviours

1 Add and remove components



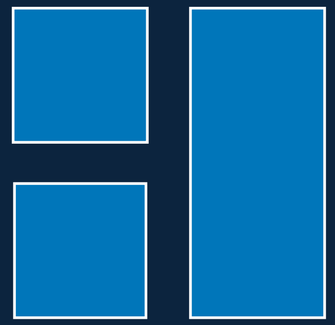
Dynamic behaviours

1 Add and remove components



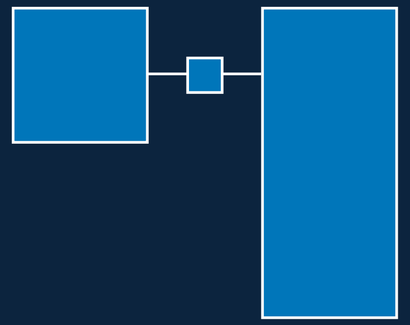
Dynamic behaviours

1 Add and remove components



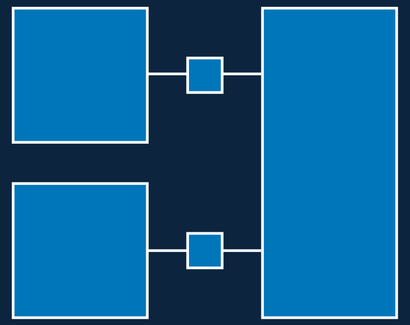
Dynamic behaviours

1. Add and remove components
2. Add and remove connections



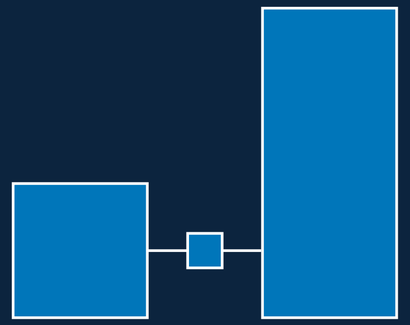
Dynamic behaviours

1. Add and remove components
2. Add and remove connections



Dynamic behaviours

1. Add and remove components
2. Add and remove connections

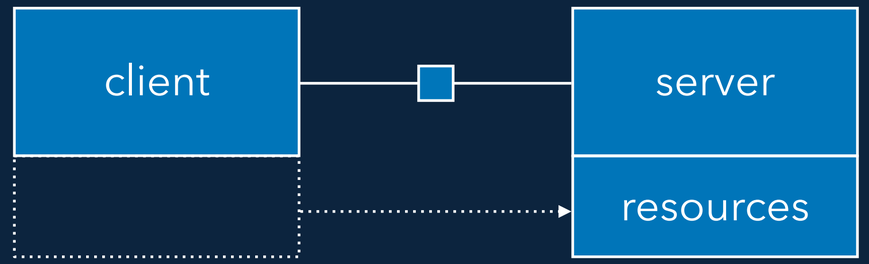


Dynamic behaviours

1. Add and remove components
2. Add and remove connections
3. Lend resources

Dynamic behaviours

1. Add and remove components
2. Add and remove connections
3. Lend resources



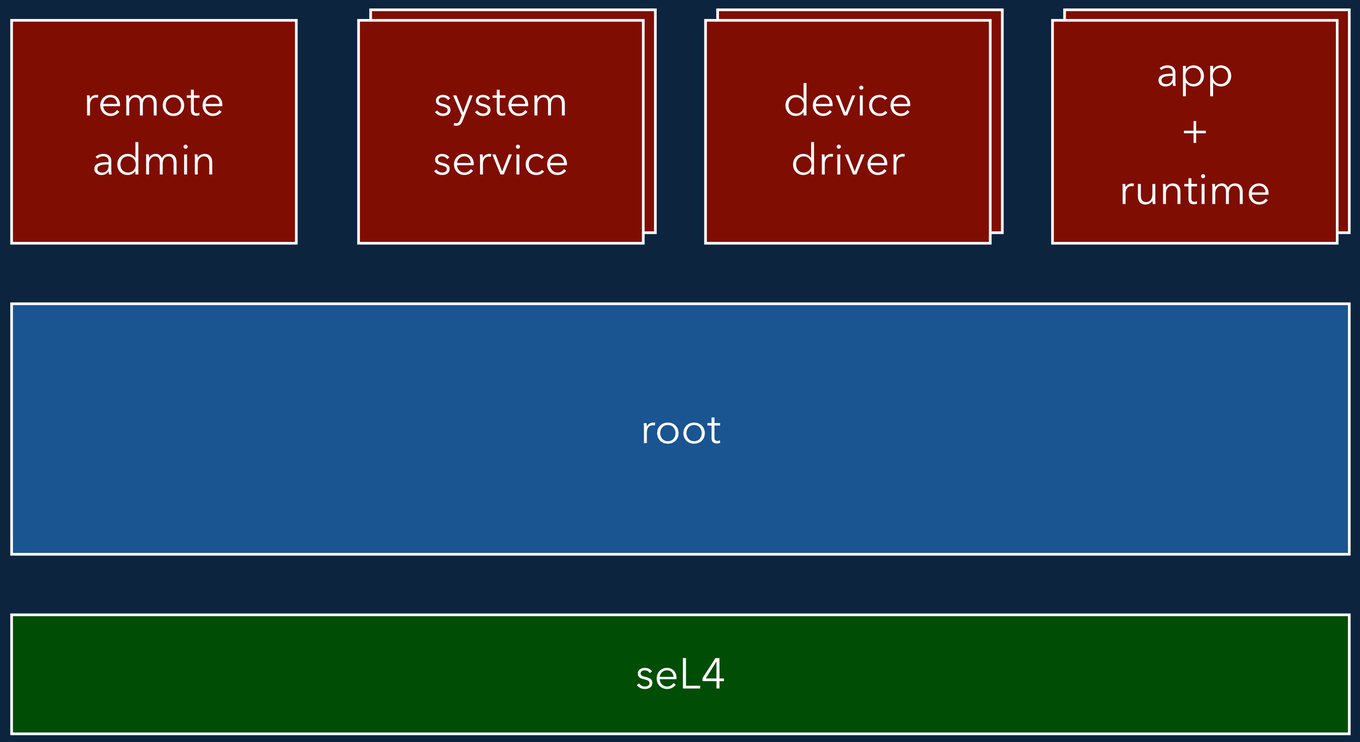
Dynamic behaviours

1. Add and remove components
2. Add and remove connections
3. Lend resources

Dynamic behaviours

1. Add and remove components
2. Add and remove connections
3. Lend resources
4. Coordinate with components

Generalising the updater pattern



unprivileged
untrusted
ephemeral

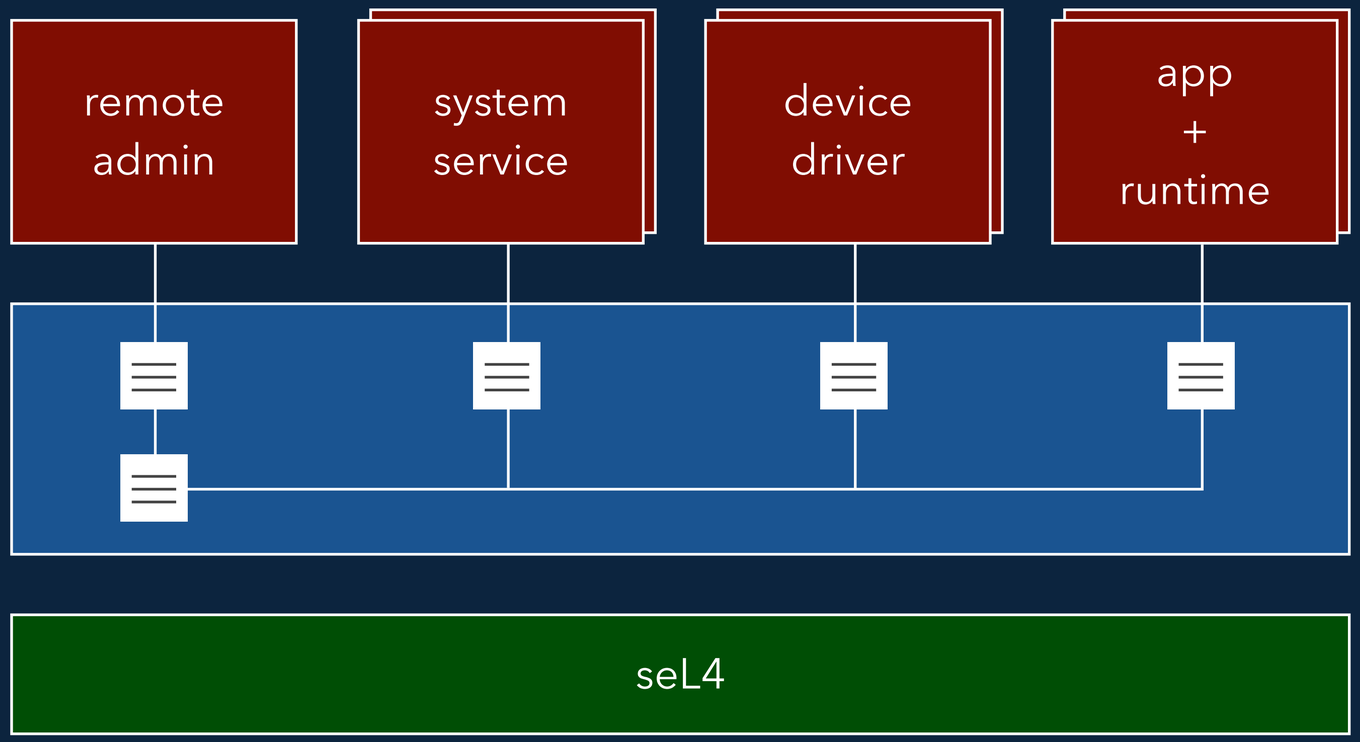
isolated (mostly)

privileged
trusted
eternal

mediates dynamic behaviour

maintains oversight

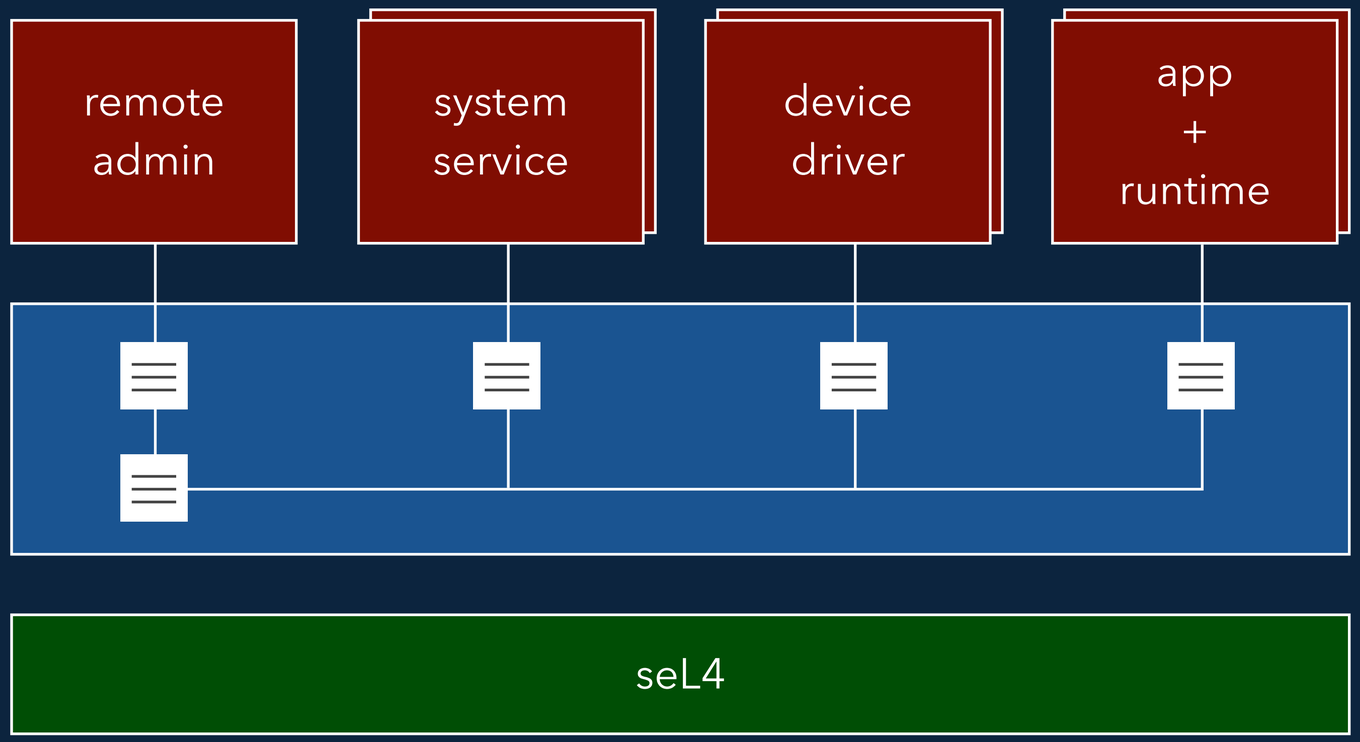
Behaviour as configuration



specify dynamic behaviours as configuration

configuration language
- low-level + high-level + abstraction mechanisms
- formal semantics
- libraries, base configuration, tools

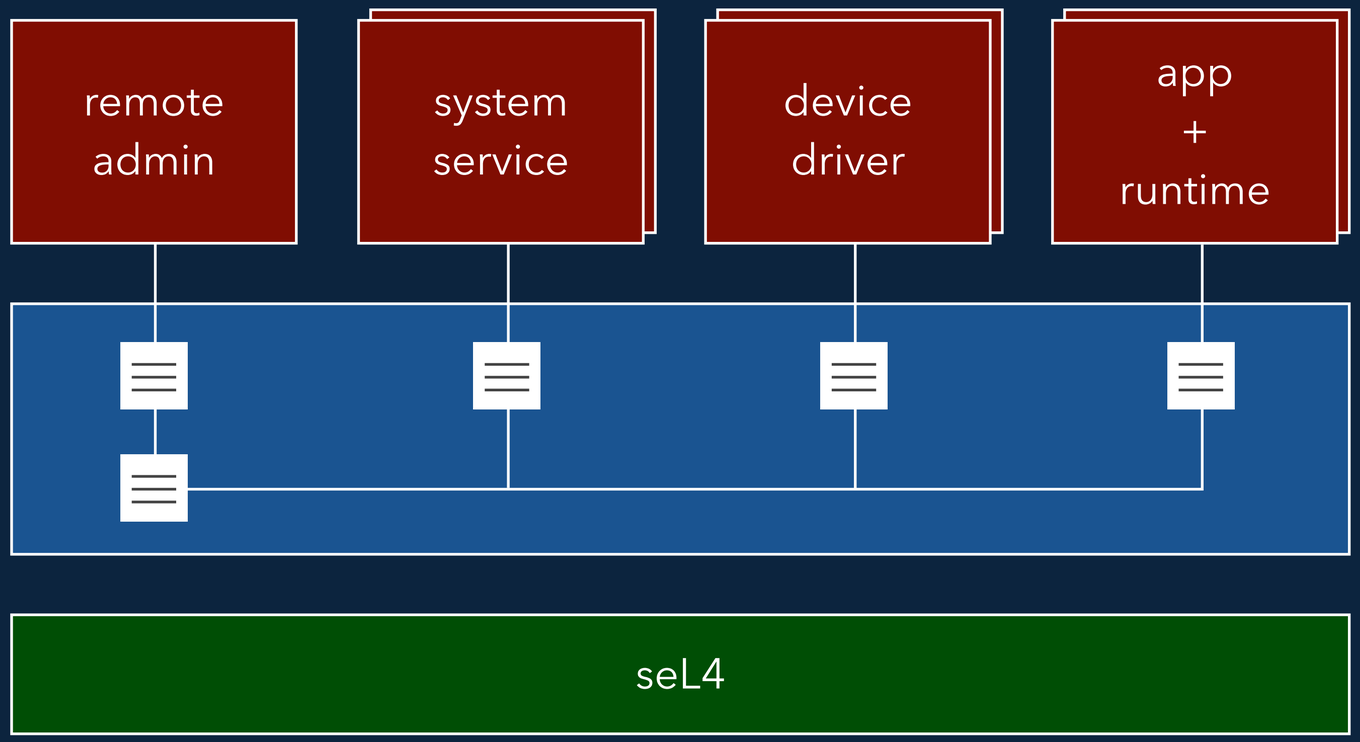
Behaviour as configuration



benefits

- separation of concerns
- stability
- reuse of assurance arguments

Behaviour as configuration



object-capability model
- low-level + high-level + abstraction mechanisms

object-capability model
- low-level only

High-level abstractions

1. Protection domains
2. Leases

Protection domains

Track resource ownership at runtime

Enforce separation

Leases

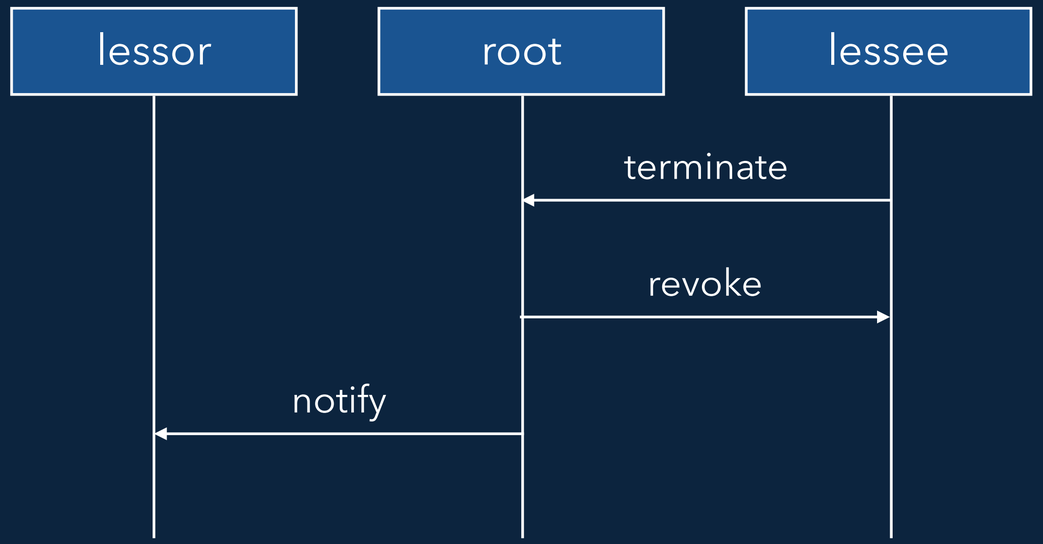
Temporarily assign resources
from one protection domain (the lessor)
to another (the lessee)

Can have subleases

Lease termination

Lessee terminates

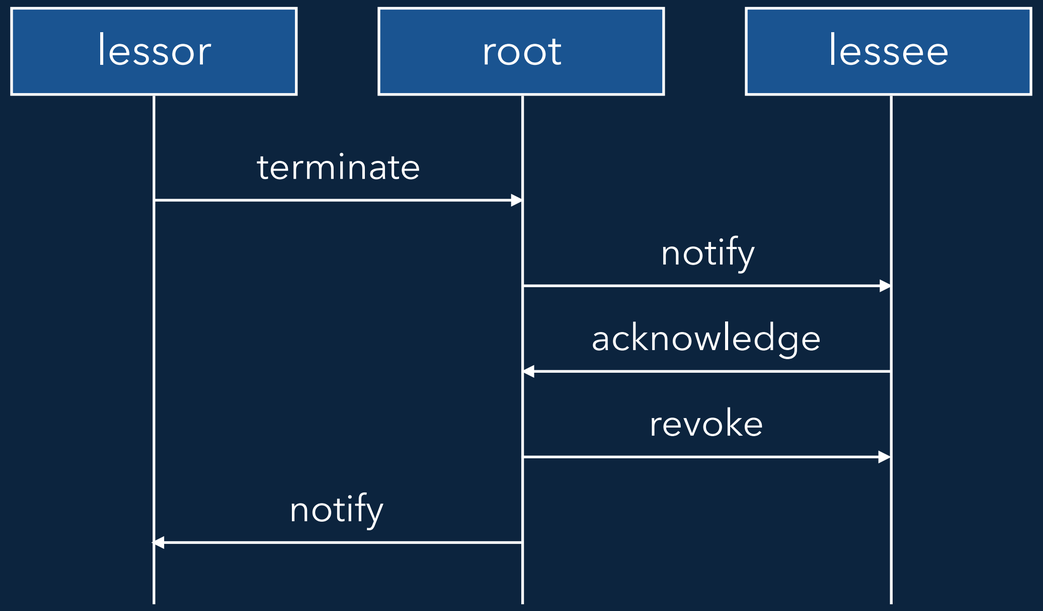
- We can assume the lessee has already stopped using leased resources

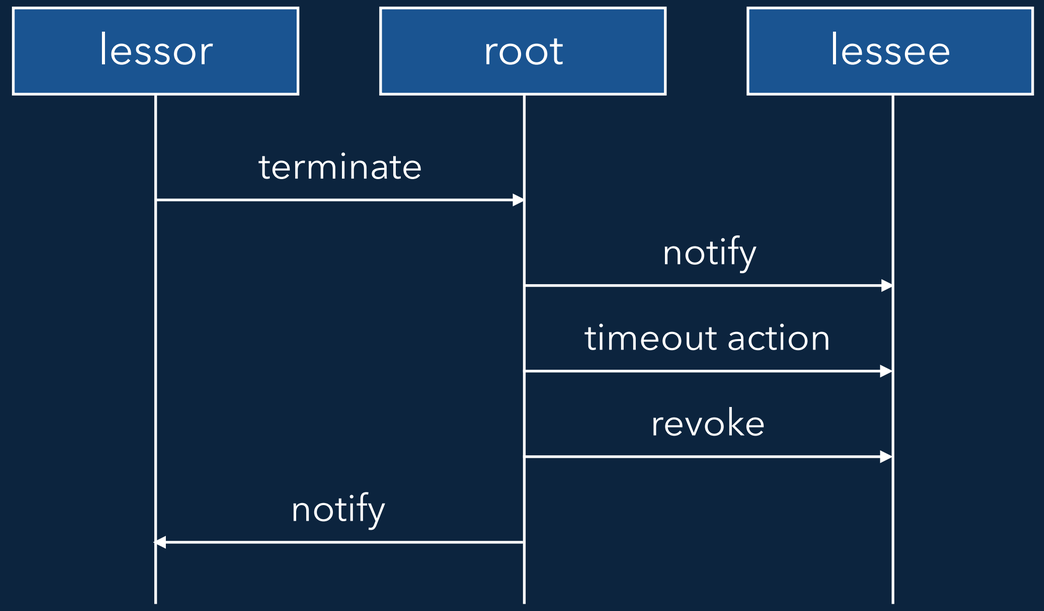


Lease termination

Lessor terminates

- We need to give the lessee a chance to stop using leased resources





Leases

Have two kinds:

- shared: endpoints, notifications, frames
- donated: untyped memory

Have rules:

- seL4 resources can only be leased if the root server controls the original cap

High-level abstractions

1. Protection domains
2. Leases

Abstraction mechanisms

Want to express allowable dynamic behaviours as configuration

- Protection domains and leases alone can’t do this

So enrich the model with abstraction mechanisms

- Make the root server an interpreter for an object-capability language

Abstraction mechanisms

Script	Binary-encoded executable function that operates on the root server object-capability model
Method	Script with some parameters already bound to root server capabilities

Script

Binary-encoded executable function that operates
on the root server object-capability model

- header describing parameters and return capabilities
- sequence of instructions

Script

Instruction set

- operations on root server and seL4 capabilities
  - create, accept, terminate a lease
  - map, uncap, read, write a frame
  - signal a notification
  - create or call a method
- maybe a capDL-like declarative fragment
- computation
  - arithmetical, logical, etc.
- control flow
- cryptography
  - authenticate remote commands and attest to their completion

Method

Script with some parameters already bound to
root server capabilities

A method does not allow retrieval of its bound capabilities
- It only allows execution

Using methods, we can create new kinds of capabilities
- Open and close connections
- Load a bundle of applications, and provide each with its connection methods

We can give method capabilities to untrusted code
- They appear as badged endpoint capabilities
- In fact, this is the only way untrusted code can access the root server

Methods give us fine control over the possible dynamic system behaviours,
including those that can be invoked by untrusted code

Packaging a system

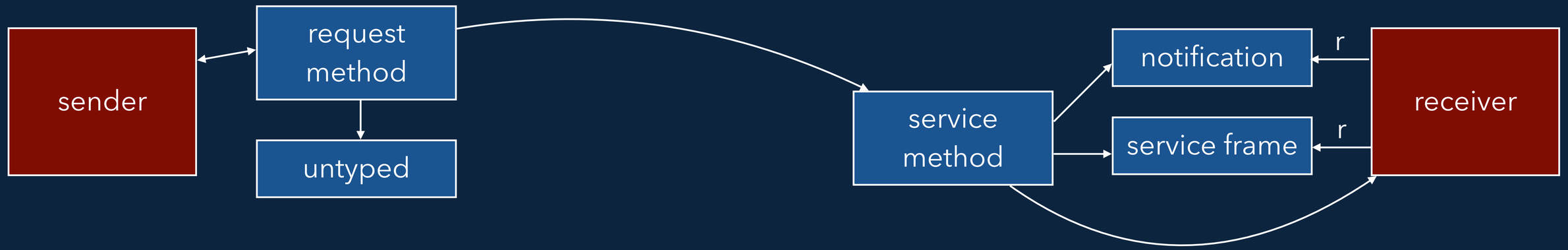
App package

- constructor script
  - parameters:
    - protection domain capability
    - initial leased resources
      - untyped
      - frames with binary images
      - service provider methods
      - service request methods
- binary images
- requirements for initial resources
- service provider scripts
- service request scripts

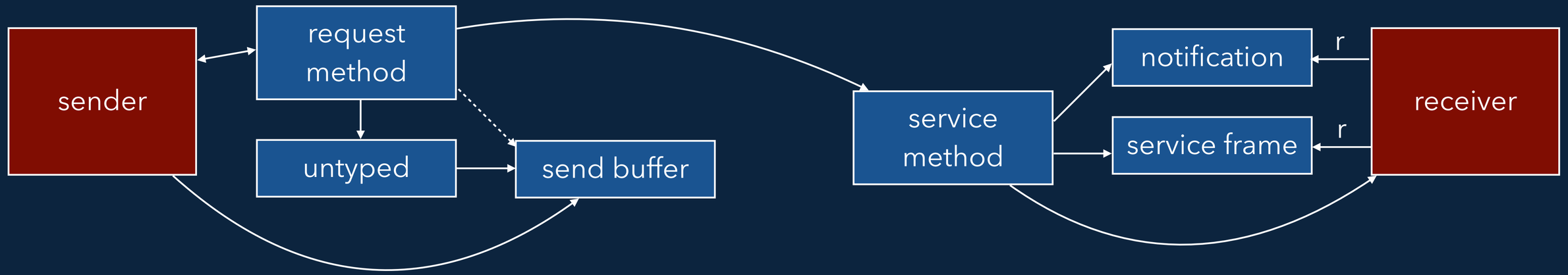
System description

- collection of app packages
- mapping of service requests to providers

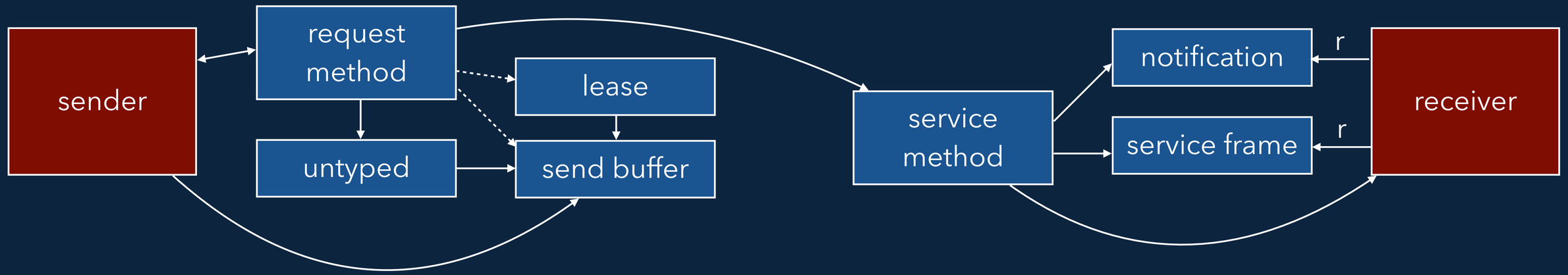
Method example: Data diode connection setup



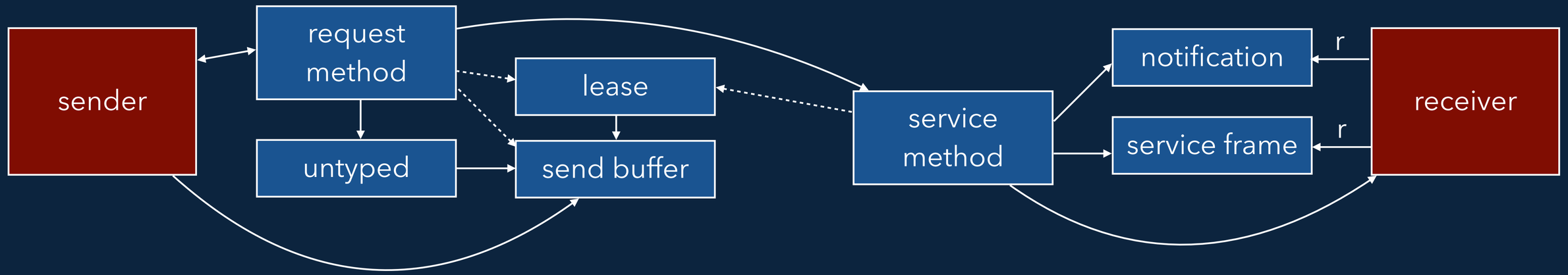
Method example: Data diode connection setup



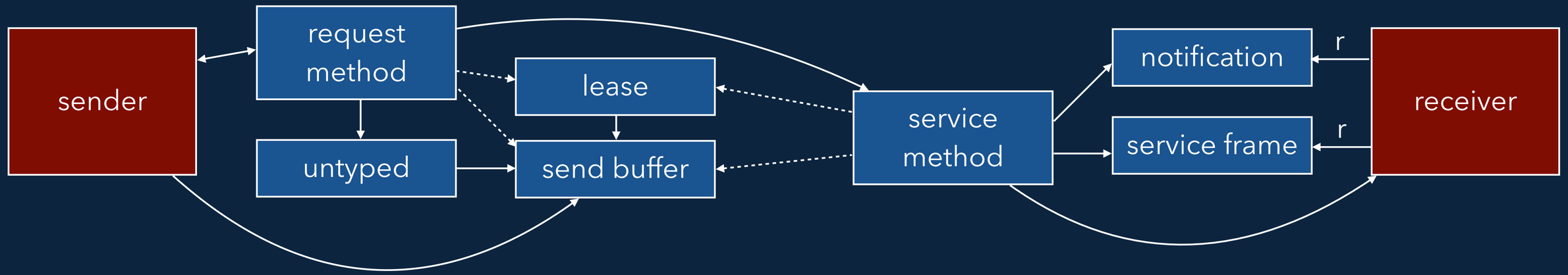
Method example: Data diode connection setup



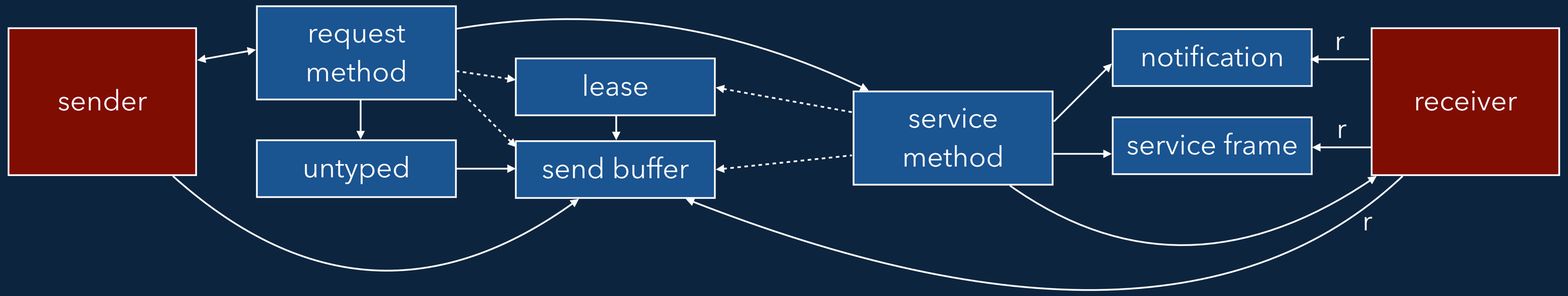
Method example: Data diode connection setup



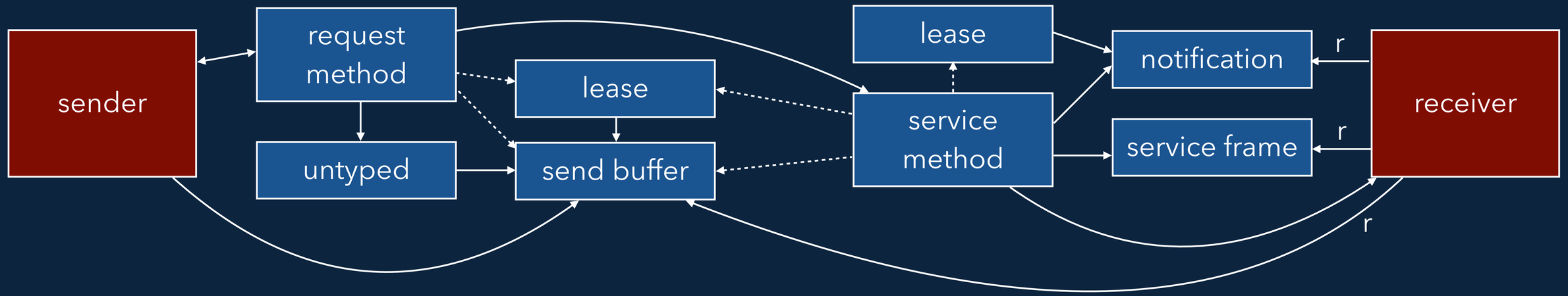
Method example: Data diode connection setup



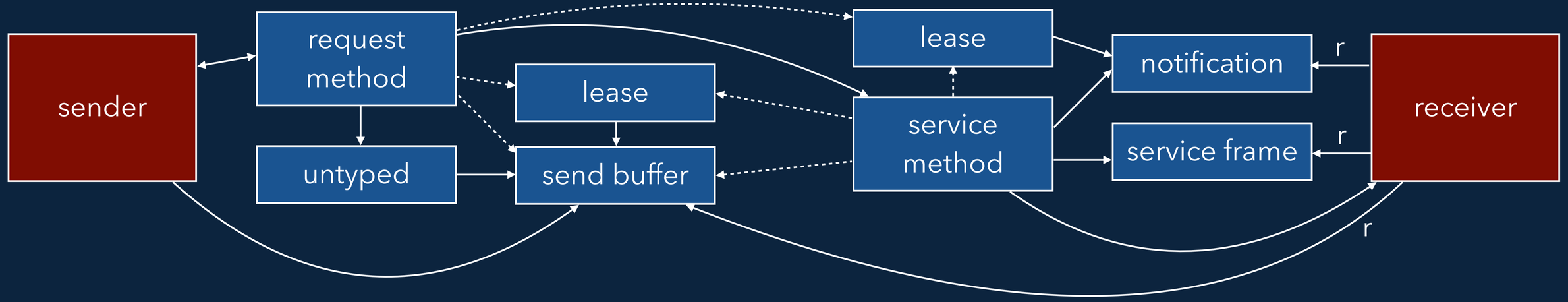
Method example: Data diode connection setup



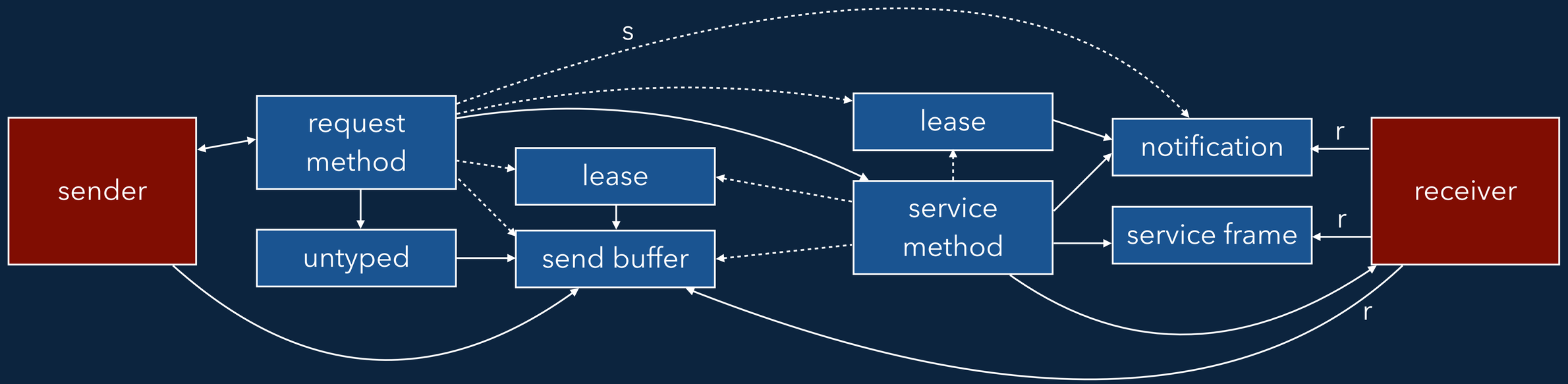
Method example: Data diode connection setup



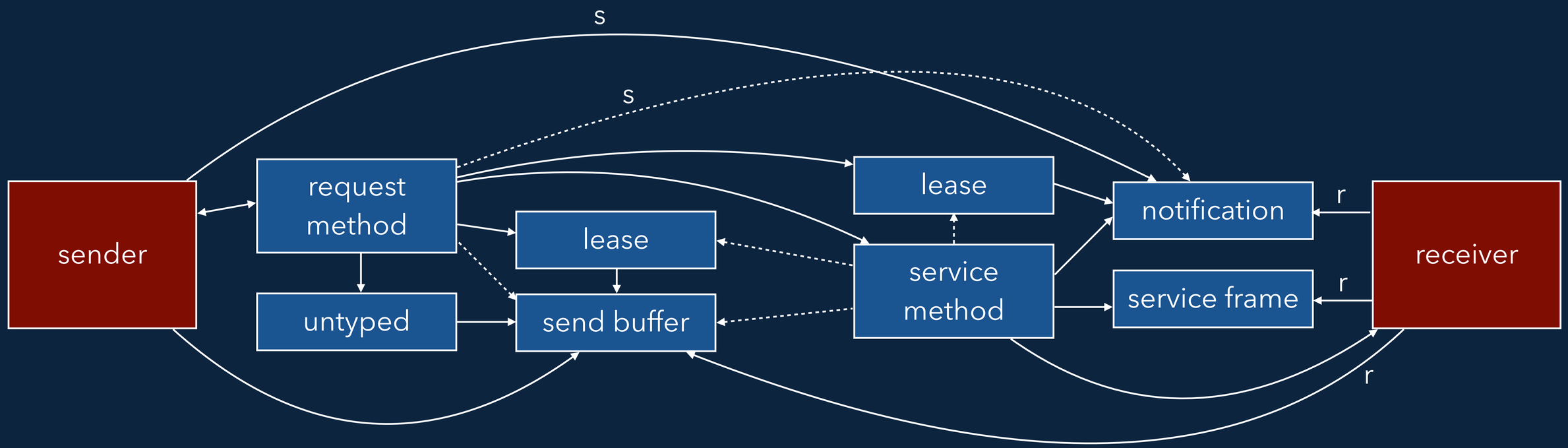
Method example: Data diode connection setup



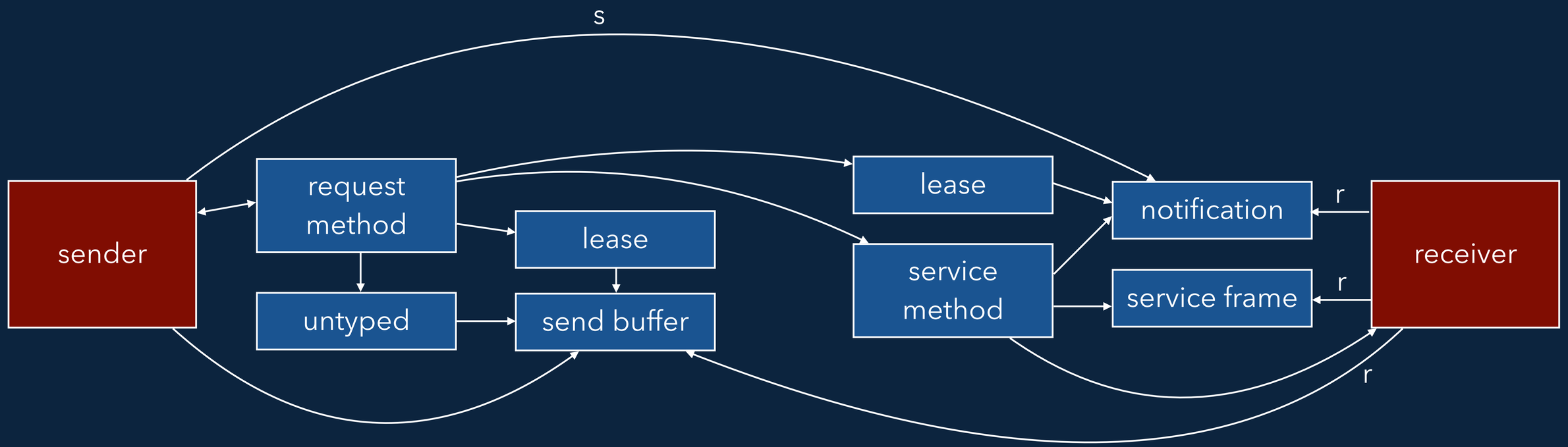
Method example: Data diode connection setup



Method example: Data diode connection setup



Method example: Data diode connection setup



Status and plans

Now

- Elaborating design concepts and details

Later

- Prototype and experiment with example systems
- Formalise root server object-capability model
- Make tools for building and reasoning with configurations
- Create user-space verification frameworks
- Prove functional correctness

Summary

Problem How to build dynamic systems so we can
- verify that systems meet their security and design goals, even after updates
- without slowing down systems developers

Solution Enrich the capability model with abstraction mechanisms, including methods