Student Presentation

Reproducing Egalitarian Paxos

COS 518: *Advanced Computer Systems*

Jinzheng Tu, Yu Zeng

4/17/2019

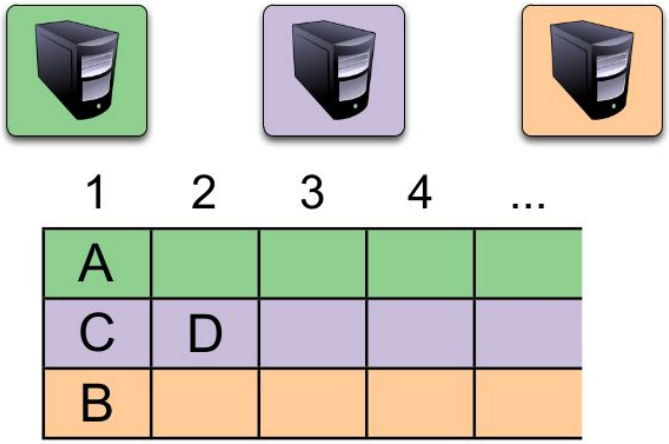
Problem Statement / Motivation

1. Paxos:
   High **latency** and low **throughput**.

2. Multi-paxos:
   Leader becomes the bottleneck for **availability**.

Key Idea

- Every replica is the leader for the requests it received.
- Latency and throughput
  - Replicas do not contend
  - Can choose any fast quorum
- Availability
  - workload balanced across all replicas



Key Challenges

- Multiple leaders for multiple requests
  - Keep the dependency relationships between commands coherent across all replicas
  - Execute commands in dependency order

Key Result (Evaluation)

```
➜ epaxos $ ./compose 3 --prod up -d
 Creating epaxos-server-0 ... done
 Creating epaxos-server-1 ... done
 Creating epaxos-server-2 ... done
# PUT <server> <key> <value>
➜ epaxos $ ./bin/client -n 3 put -v 0 114 514
 PUT: [114]=514 to 0 start
 PUT: RequestMsg 4972707484880603879 sent to 0
 PUT: [114]=514 to 0 committed
# GET <server> <key>
➜ epaxos $ ./bin/client -n 3 get 1 114
 <!NOT IMPLEMENTED!>
```

Key Result (Evaluation)

```
# 1000 random PUTs to 1000 keys on 3 servers
➜ epaxos $ ./bin/client -n 3 batch-put 0 1000
 -N 1000 --pipeline 10000 -S 3 --random-key
 --random-server >/dev/null && echo Good
Good
```

Key Result (Evaluation)

```
# Emulate 10% package loss
➜ epaxos $ make pumba-loss-10 && ...
 Good
# Emulate 1000ms delay
➜ epaxos $ make pumba-delay-1000ms && ...
 Good
# Random kill
➜ epaxos $ docker kill epaxos-server-0 && ...
...Nope...
# Emulate 0.1% package duplication
➜ epaxos $ make pumba-dup-0.1 && ...
...Nope...
```

Future Plan

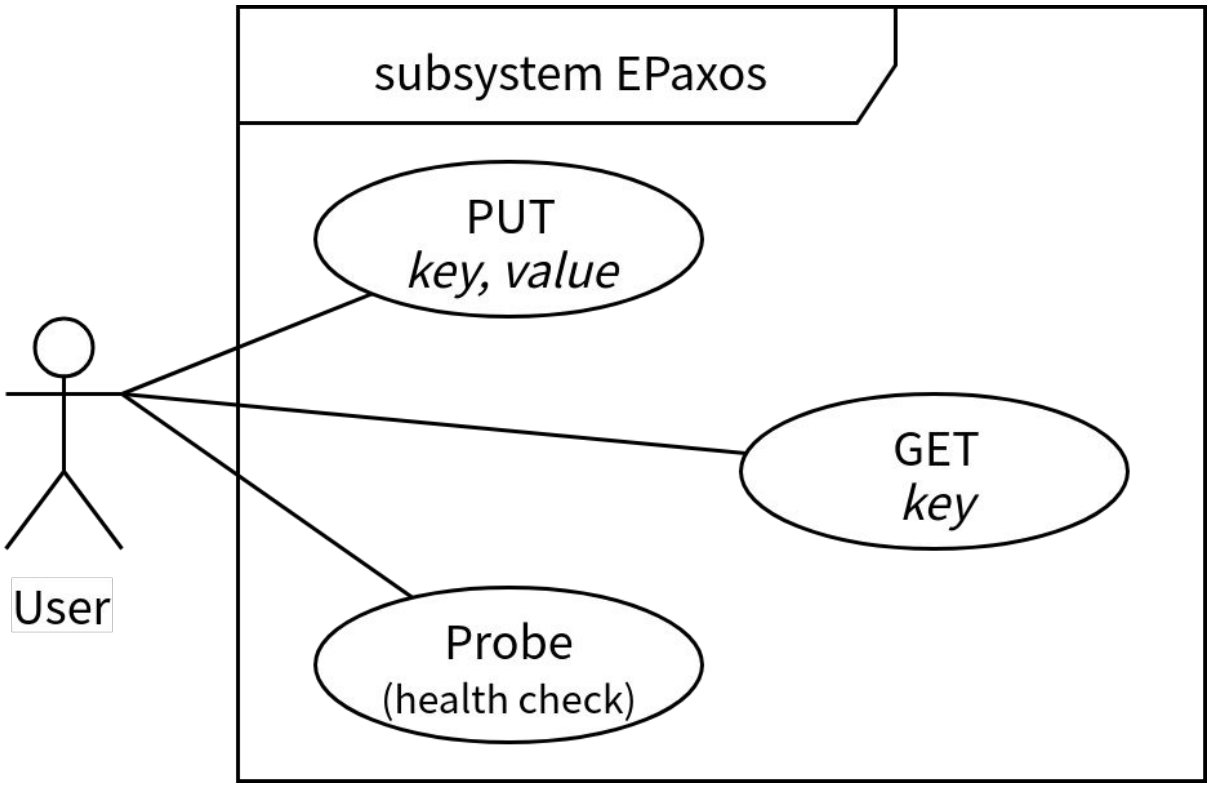
- Implement "Prepare" logic (recovery)
- Implement "GET" logic
- Debug
- and debug...
- Test correctness against (emulated) delay, duplication, loss, random kill, etc.
- Performance test by kubernetes on GKE!

Technical Details

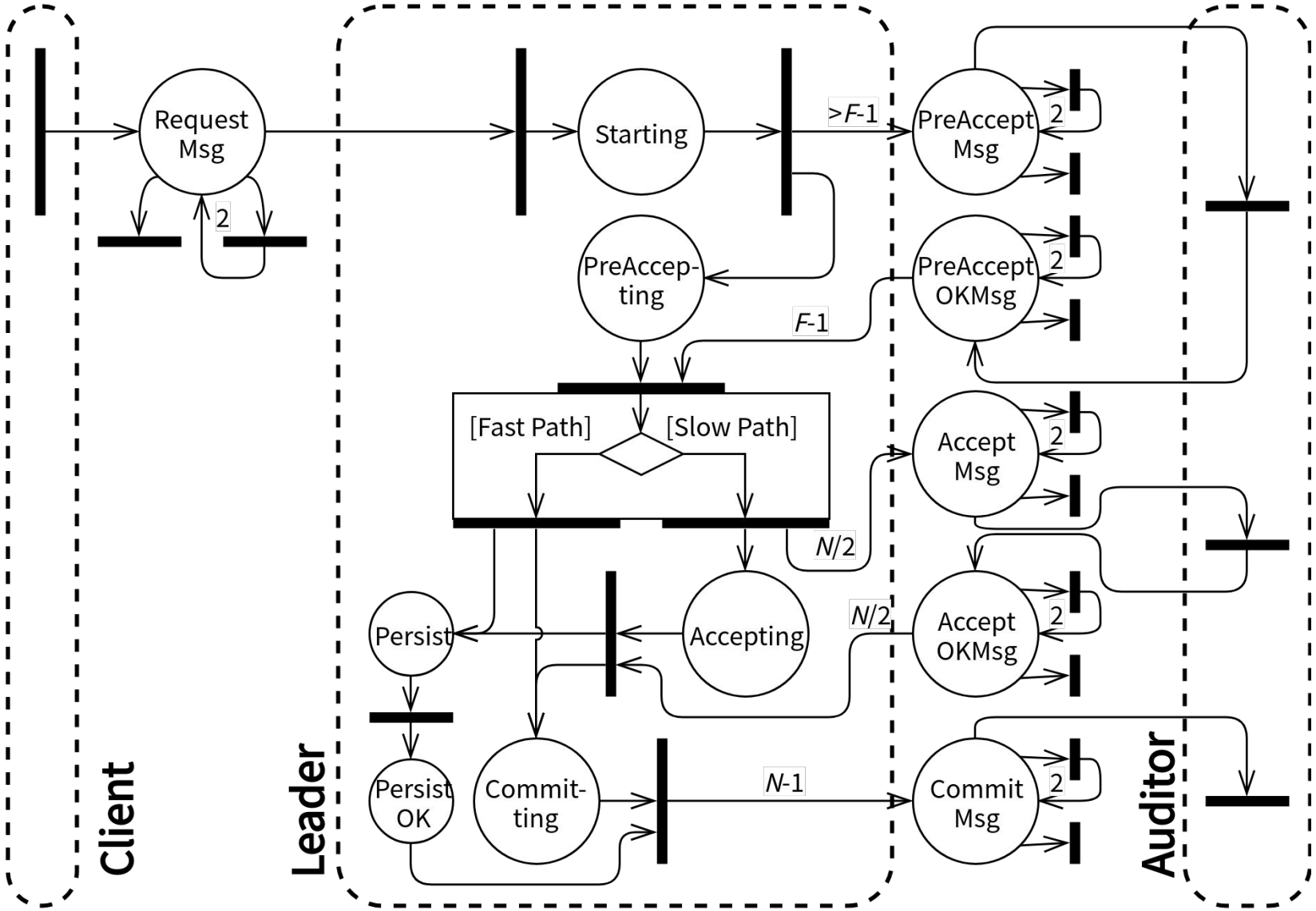
System Architecture

- "4+1" views:
  - Logical View
  - Process View
  - Development View
  - Physical View
  - ~~Scenario View~~

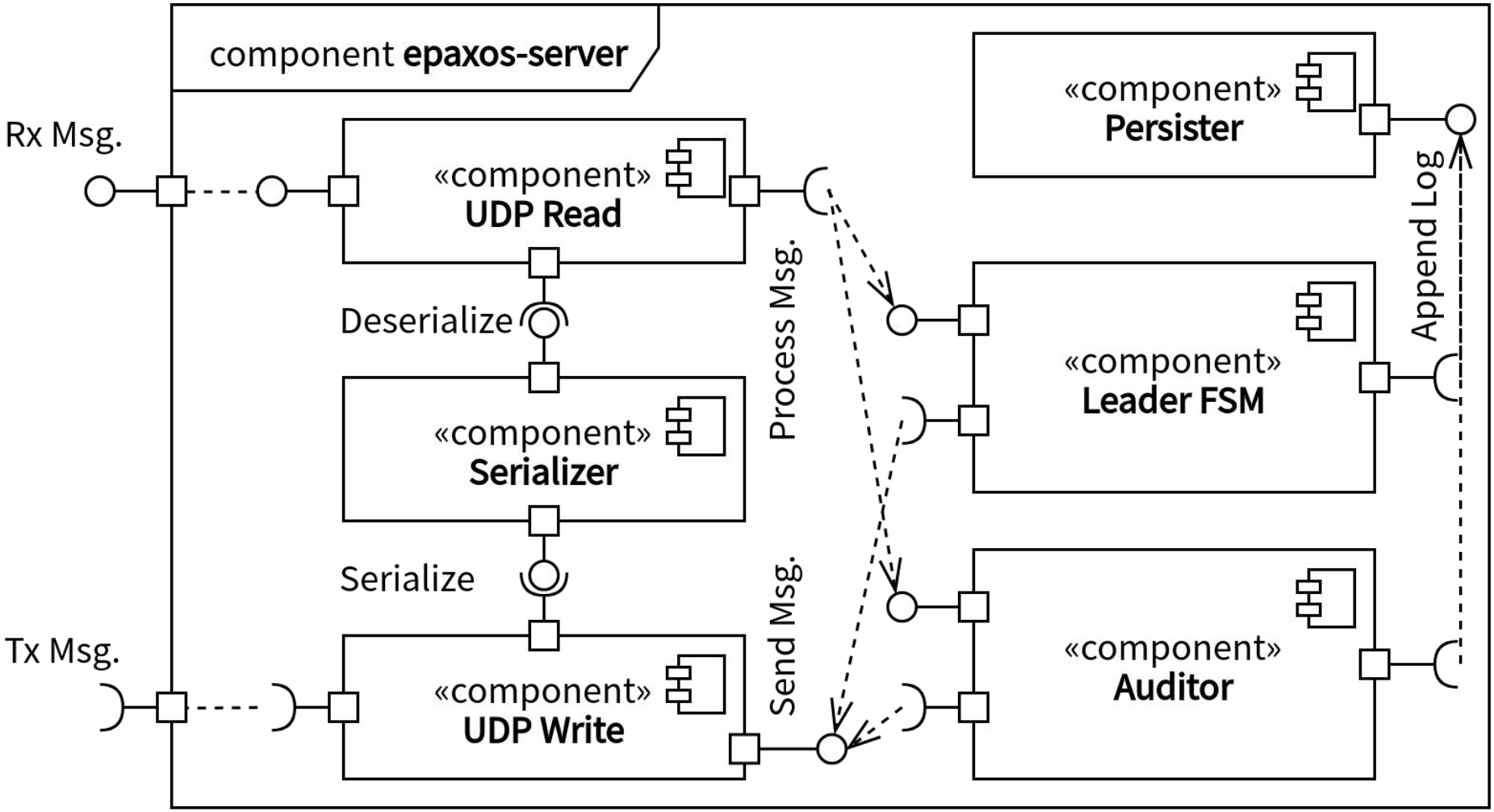
Logical View: Use Cases



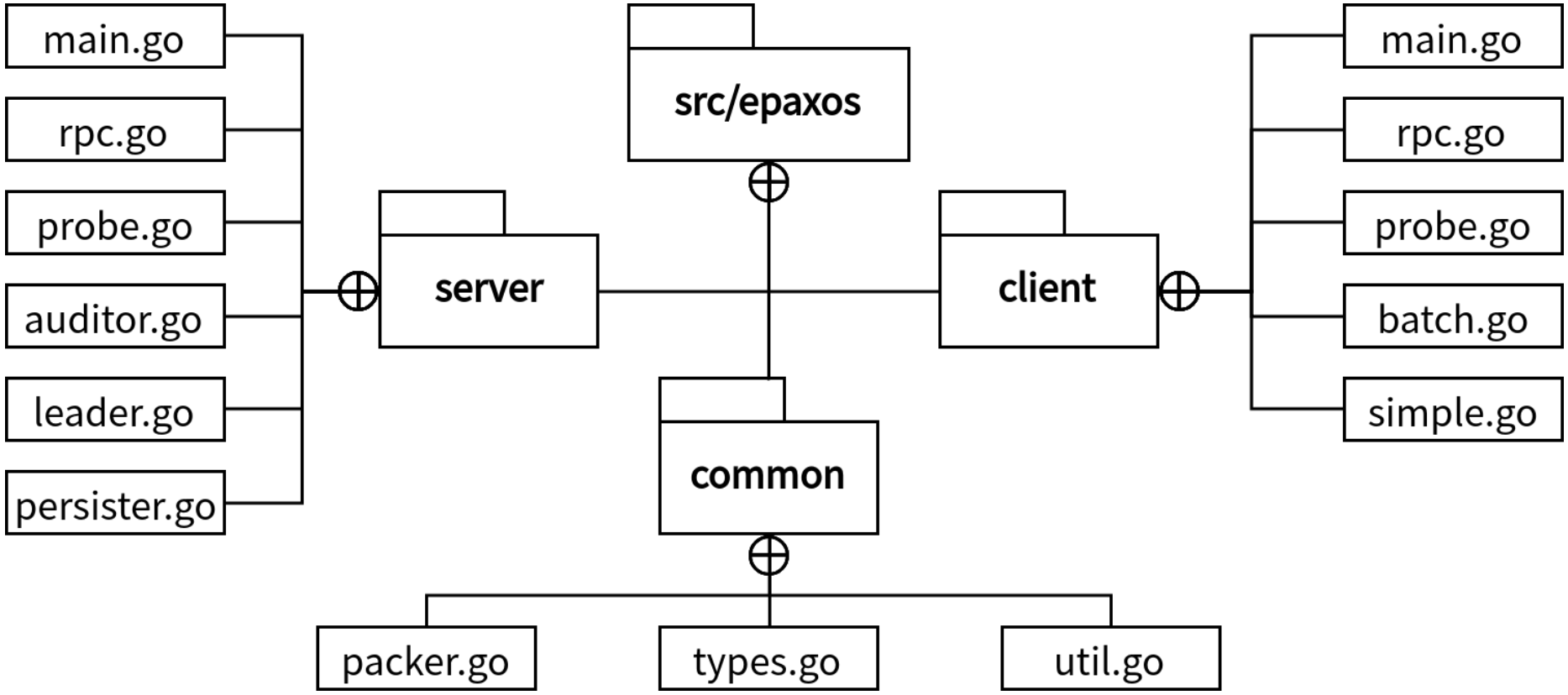
Process View: PUT



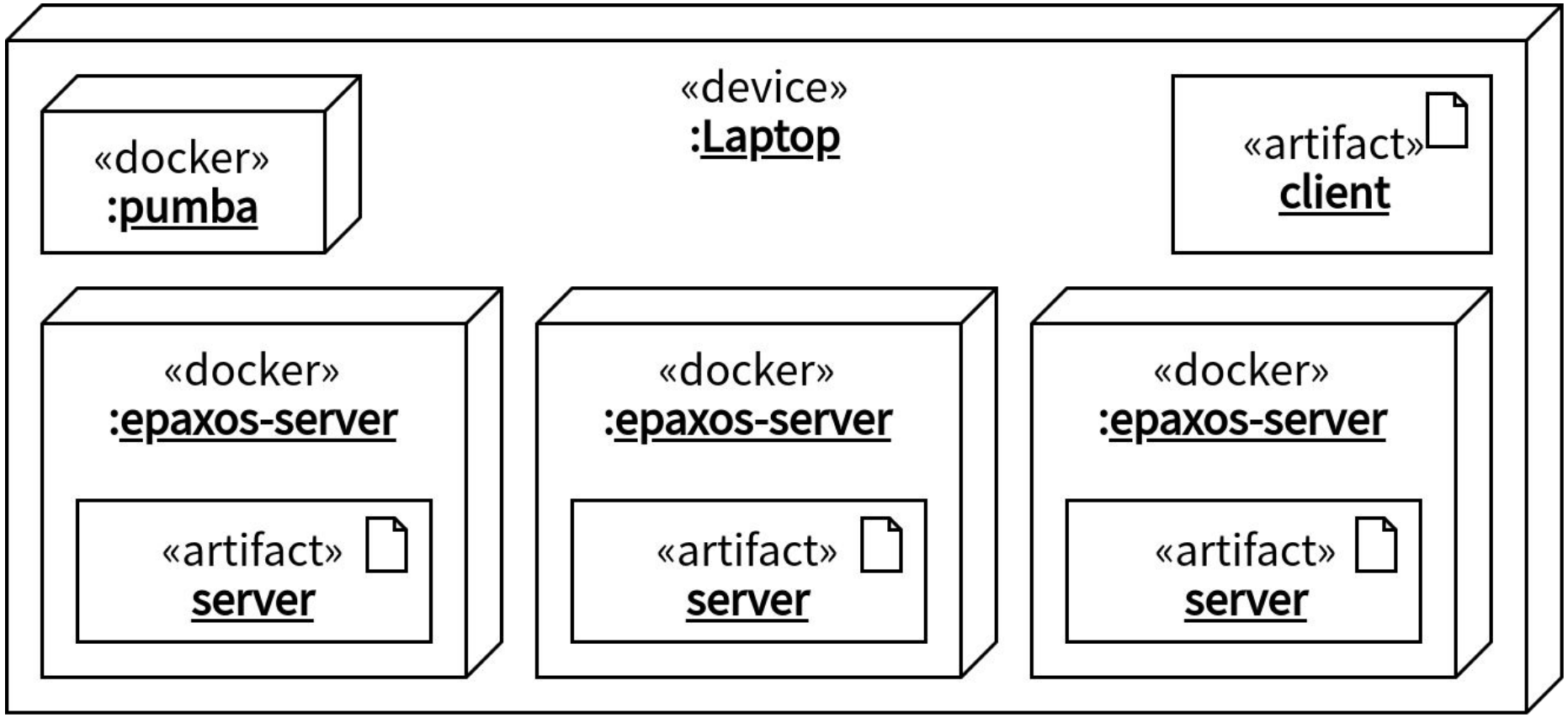
Development View: Components



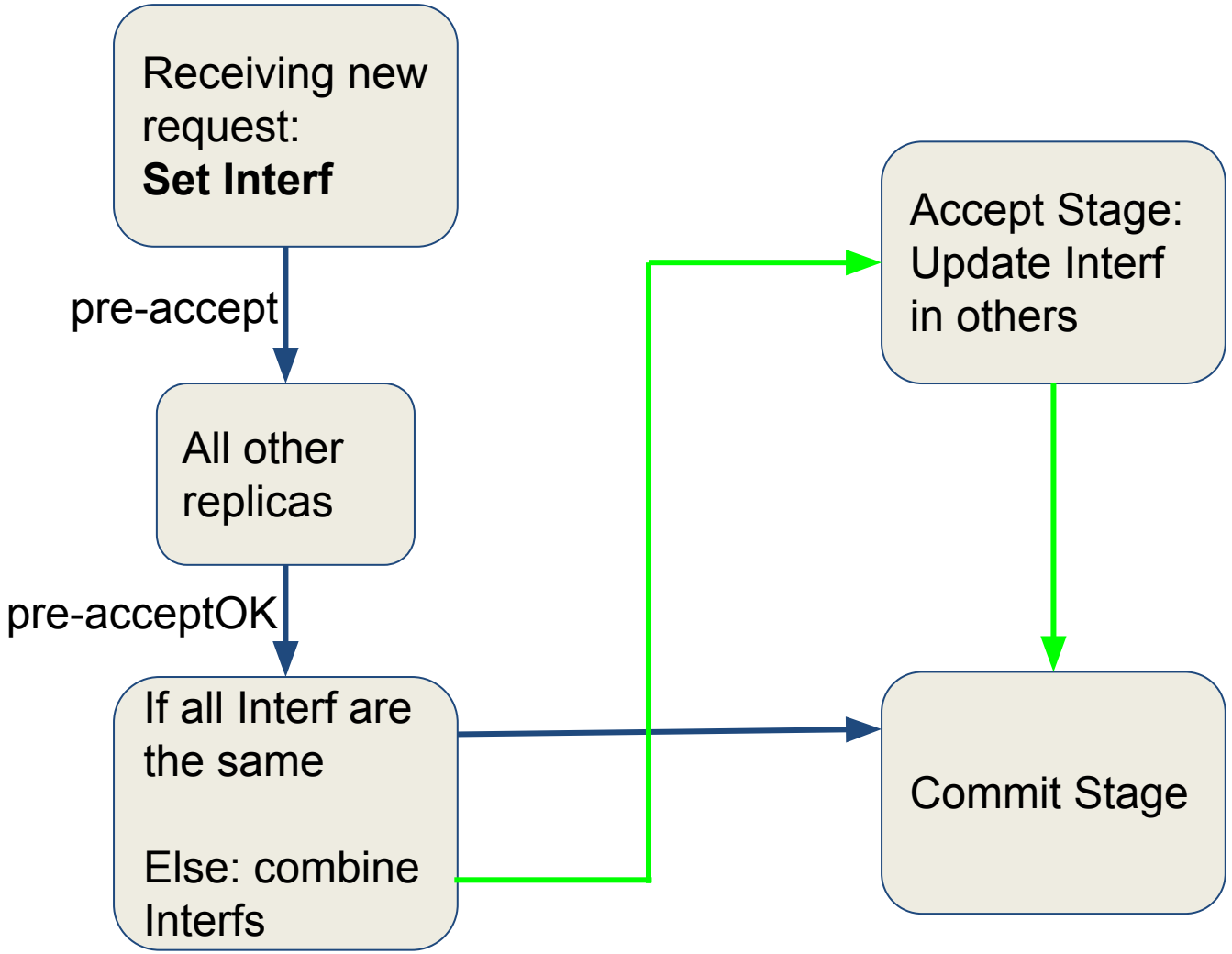
Development View: Packages



Physical View: Testing environment



Key Idea: How to make deps coherent



Key Idea: Deal with failure - Prepare

- One replica finds one potentially failed replica
  - Sends Prepare msg to all replicas
    - If replies include committed phase, run Commit phase
    - if replies include accepted phase, run Accept phase
    - if replies include > N/2 pre-accept phase, run Accept phase
    - Otherwise, run pre-accept phase