TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Practical Byzantine Fault Tolerance (Miguel Castro, Barbara Liskov)

presented by Bjoern Doebel

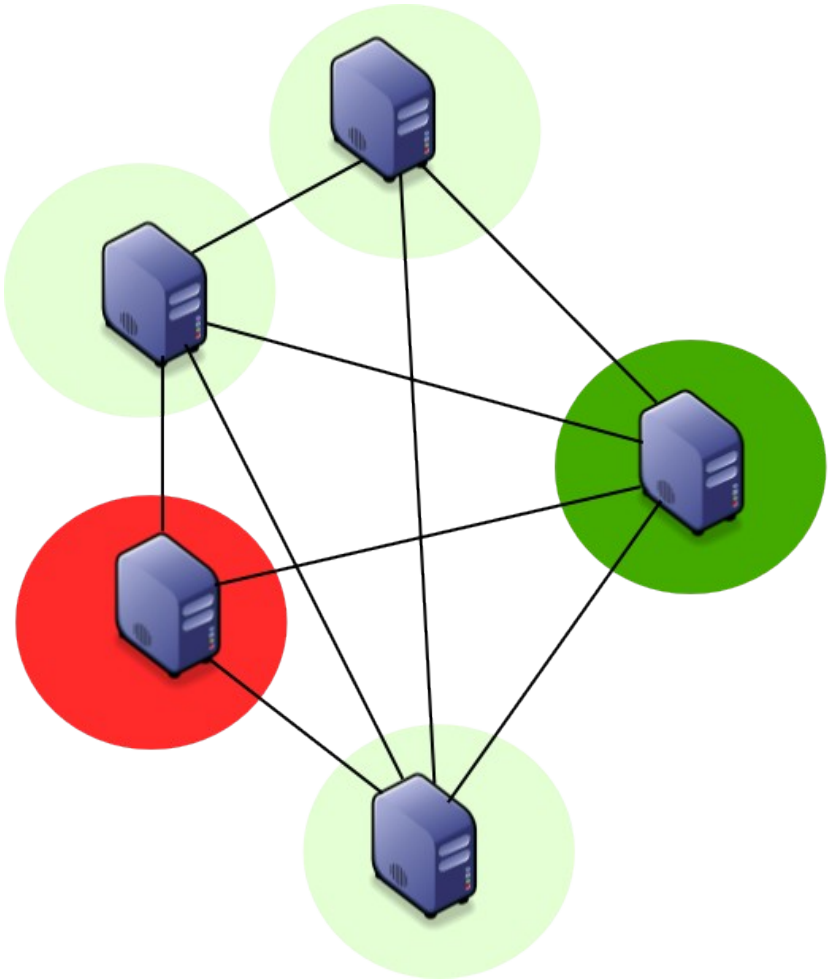
Dresden, 2008-11-05

Motivation

- Byzantine Faults
  - Undetected failures
  - Solution: majority voting
    - *n* replicas, *f* faults tolerated -> $n > 3f$
- State machines
  - Need total order of executed requests
- Solutions typically synchronous
  - Slow

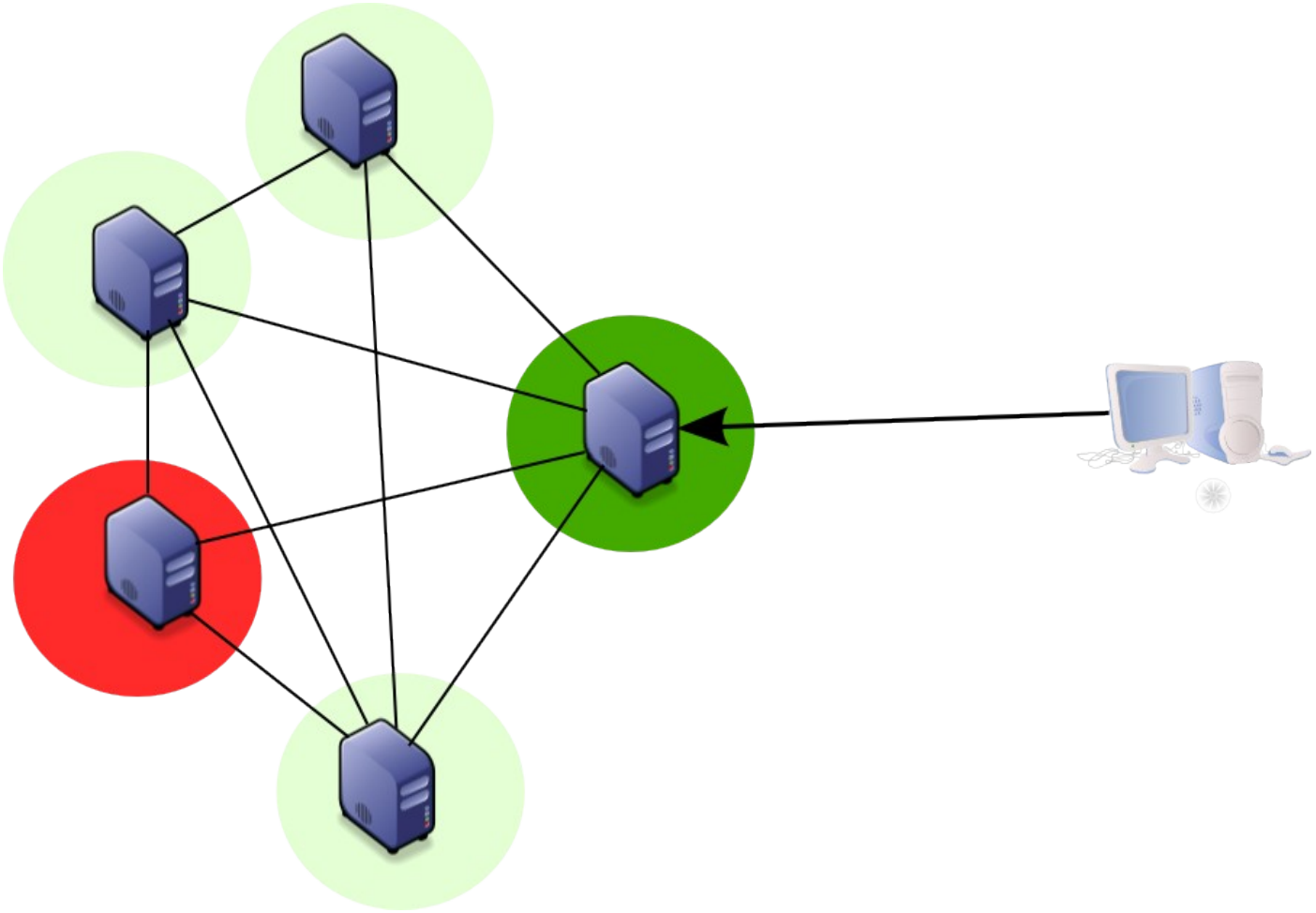
TECHNISCHE UNIVERSITÄT DRESDEN

BFT Protocol



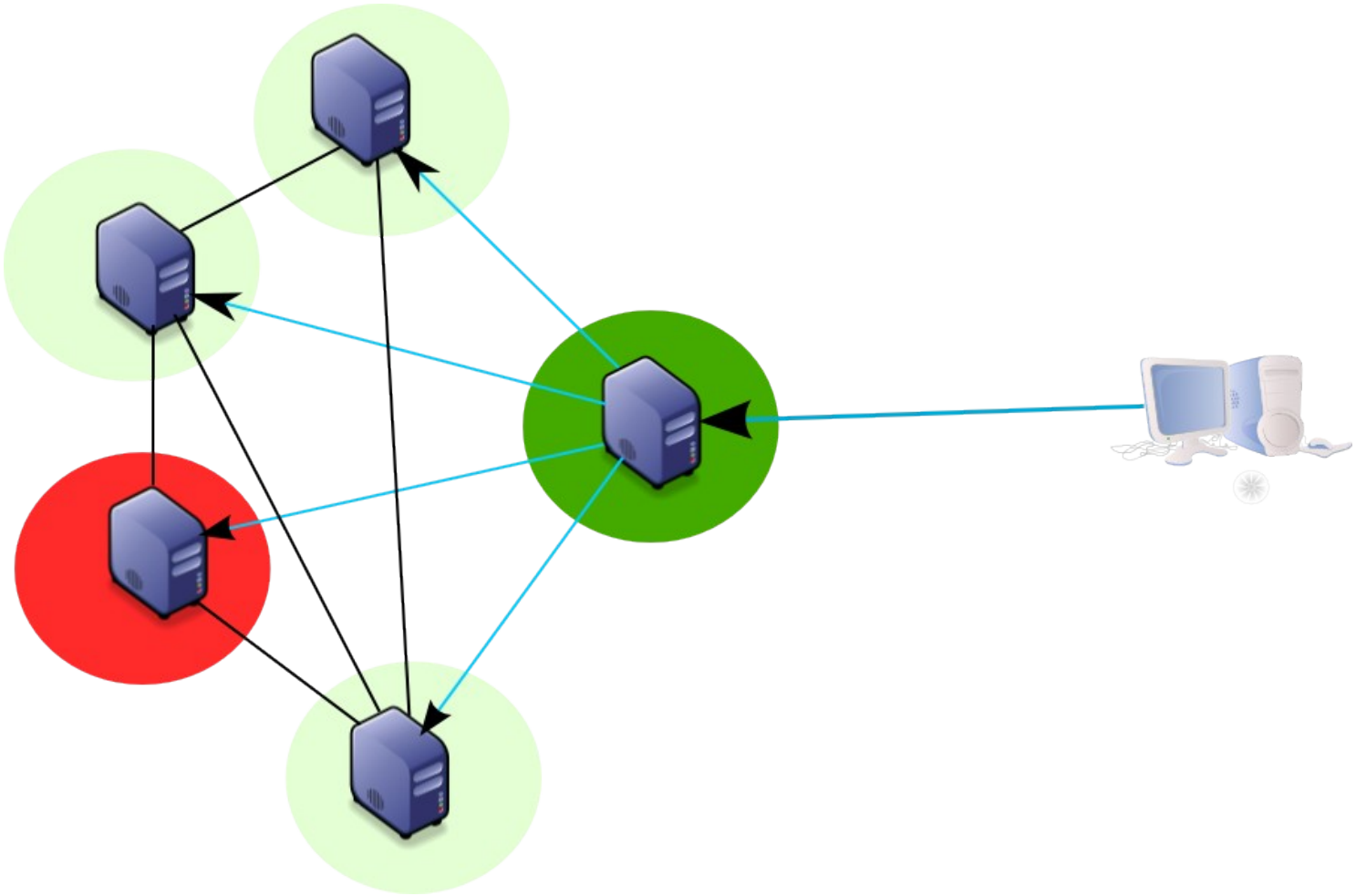
TECHNISCHE UNIVERSITÄT DRESDEN

BFT Protocol



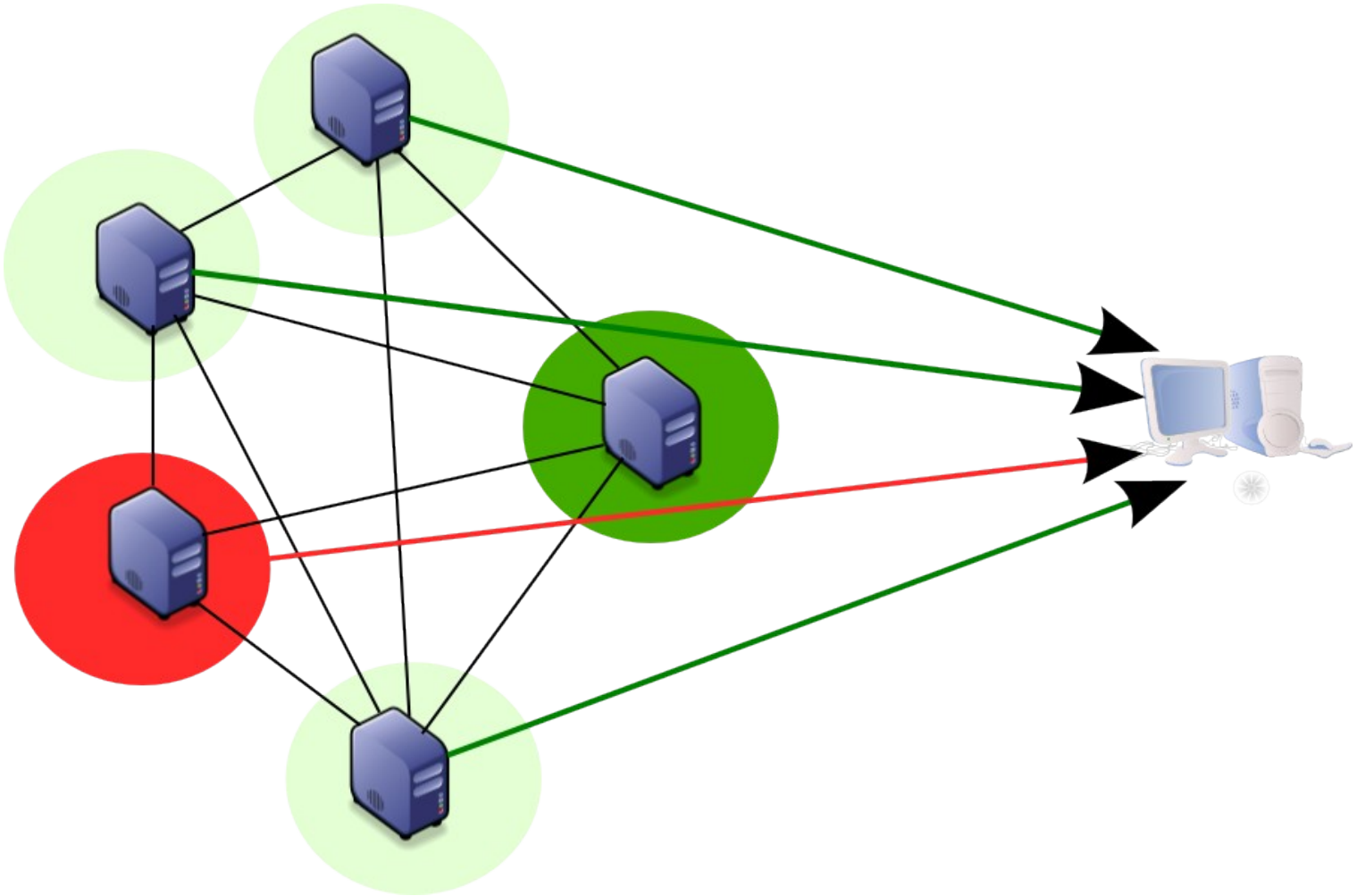
TECHNISCHE UNIVERSITÄT DRESDEN

BFT Protocol



TECHNISCHE UNIVERSITÄT DRESDEN

BFT Protocol



Protocol stages

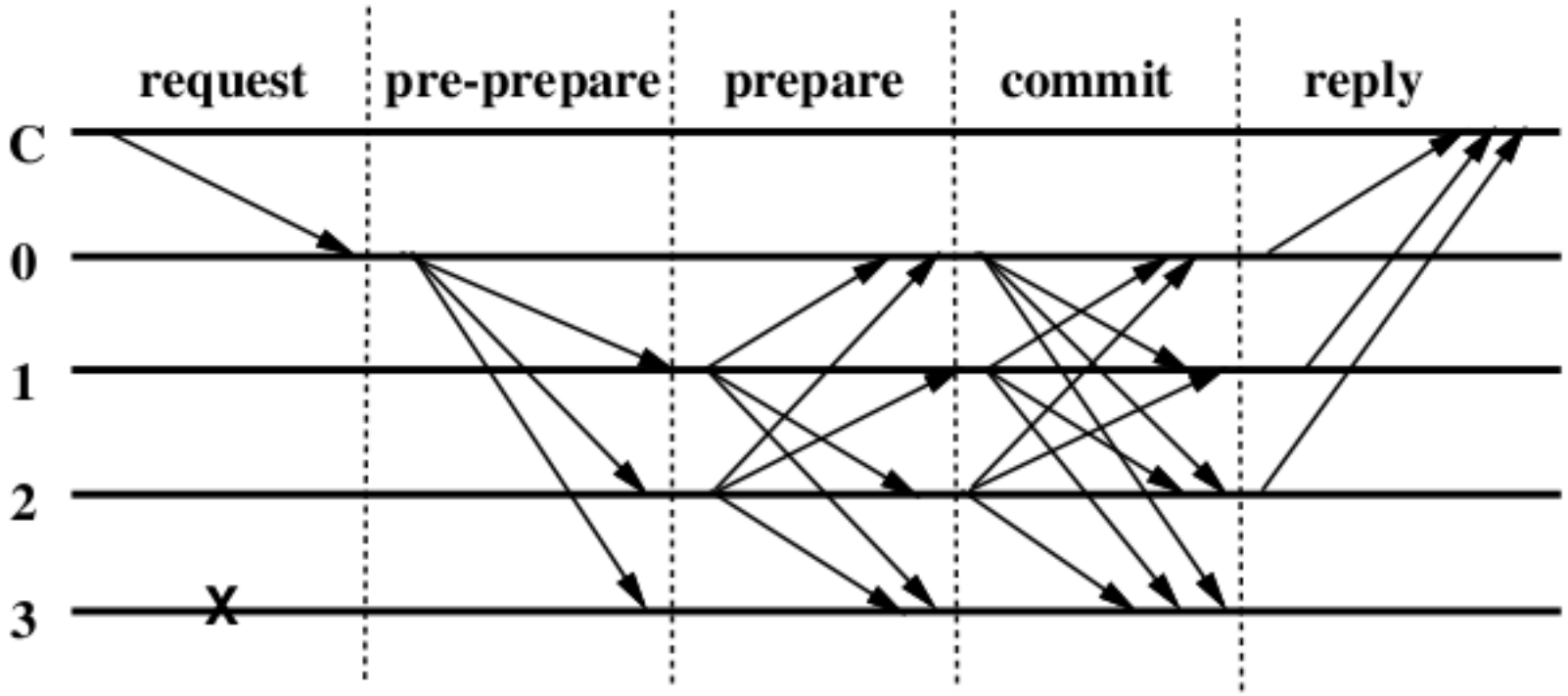


Figure 1: Normal Case Operation

Maintaining Service State

- Periodic checkpoints of local service state using message log
  - Proof generation
    - multicast checkpoint msg to all peers
  - 2f+1 commit messages with same log
    - *stable checkpoint*
    - Can discard previous log entries

Views

- Each replica maintains a view on who is the primary
- Clients / replicas may detect faulty primary
- Replicas initiate view change protocol
  - Don't accept requests anymore
  - Broadcast view change to next primary
  - Await replies (with timeout in case next one is faulty, too)

The “practical” part

- Implemented Byzantine NFS daemon
  - “...does not implement view changes or retransmissions at present.”
- Propose some protocol optimizations
  - Only send one result, rest of replicas only sends signature
  - Replicas reply tentatively to the client (and commit later)
- Evaluate performance with Andrew benchmark
  - ~20% performance overhead

Musings

- Did I understand the protocol?
- Is it ok, to only measure normal-case performance?
- Is this stuff relevant or only a scientific thing?

TECHNISCHE UNIVERSITÄT DRESDEN

Musings...