Consensus as a Network Service

Huynh Tu Dang, Pietro Bressana,
Han Wang, Ki Suh Lee, Hakim Weatherspoon,
Marco Canini, Fernando Pedone, and Robert Soulé
Università della Svizzera italiana (USI),
Cornell University, and KAUST

Consensus is a Fundamental Problem

- Many distributed problems can be reduced to consensus
  - E.g., Atomic broadcast, atomic commit
- Consensus protocols are the foundation for fault-tolerant systems
  - E.g., OpenReplica, Ceph, Chubby
- Any improvement in performance would have HUGE impact

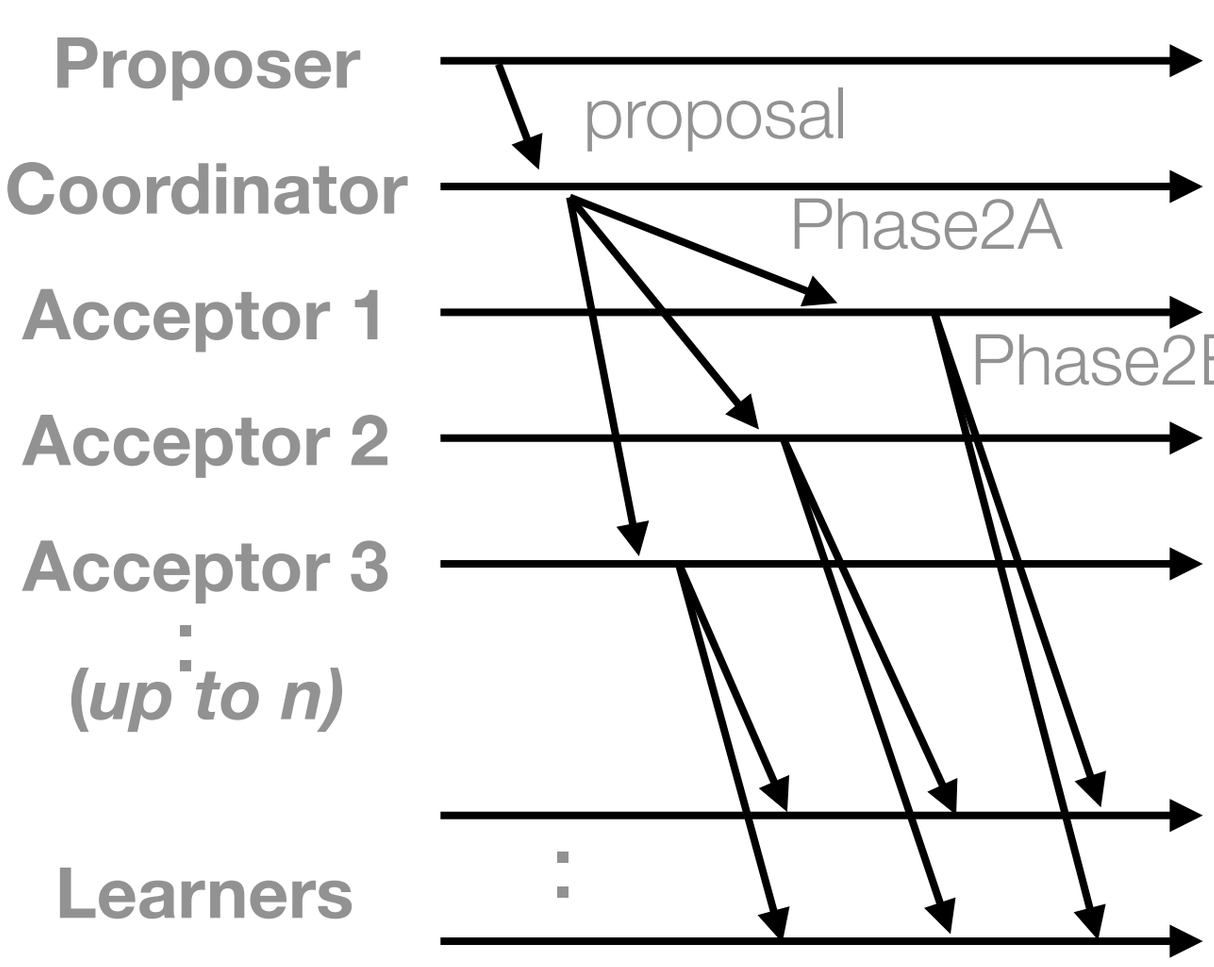
Key Idea: Move Consensus Into Network Hardware

- This work focuses on Paxos
  - One of the most widely used consensus protocol
  - Has been proved to be correct
- Enabling technology trends:
  - Hardware is becoming more *flexible*: e.g. PISA, FlexPipe, NFP-6xxx
  - Hardware is becoming more *programmable*: e.g., POF, PX, and P4

Outline of This Talk

- Introduction
- **Consensus Background**
- Design, Implementation & Evaluation
- Conclusions

Paxos Roles and Communication



- Proposers propose values
- A distinct proposer assumes the role of Coordinator
- Acceptors accept a proposal, promise not to accept any other proposals
- Learners require a quorum of messages from Acceptors, “deliver” a value

Design

Design Goals 1:
Be a Drop-In Replacement

- István et al. [NSDI ’16] implement ZAB in FPGAs, but require that the application written in the Hardware Description Language
- High-level languages make hardware development easier
- Implementing LevelDB in P4 might still be tricky….

Standard Paxos API

```
void submit(struct paxos_ctx * ctx,
            char * value,
            int size);
```

Send a value

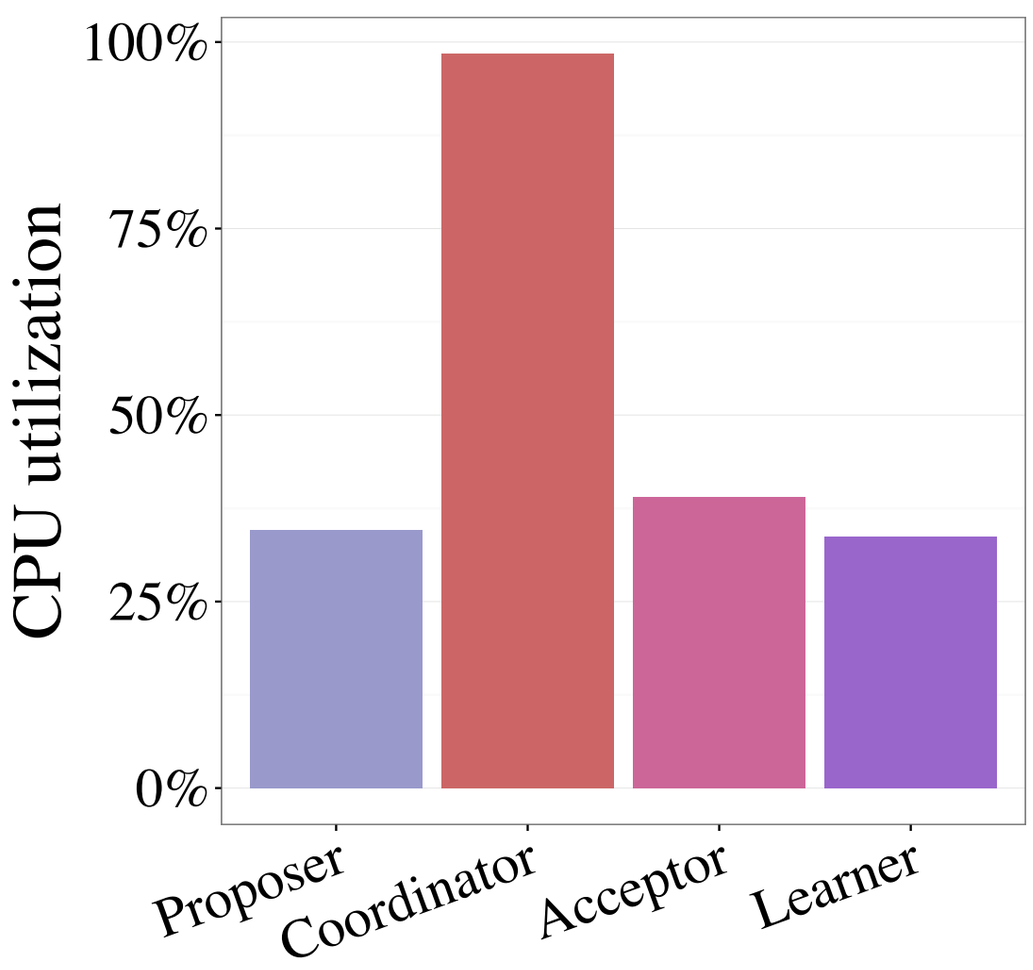
```
void (*deliver)(struct paxos_ctx* ctx,
                int instance,
                char * value,
                int size);
```

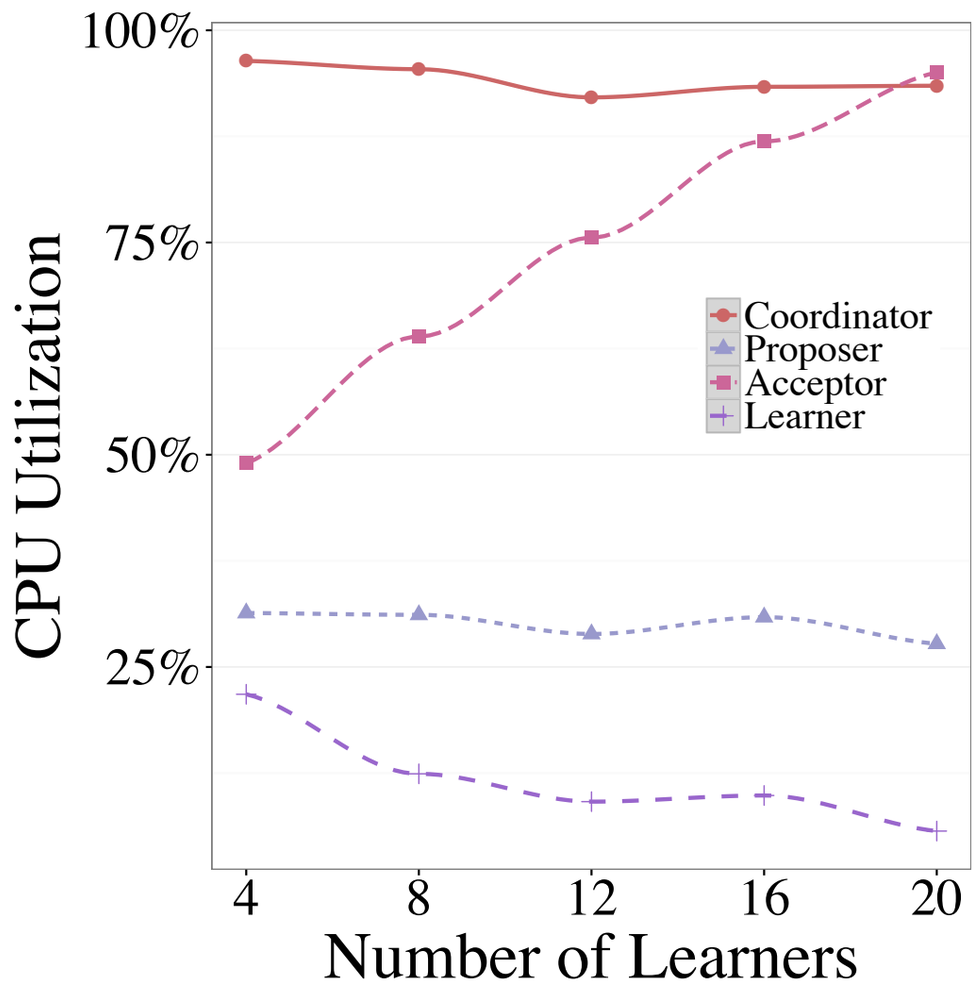
Deliver a value

```
void recover(struct paxos_ctx * ctx,
             int instance,
             char * value,
             int size);
```

Discover prior value

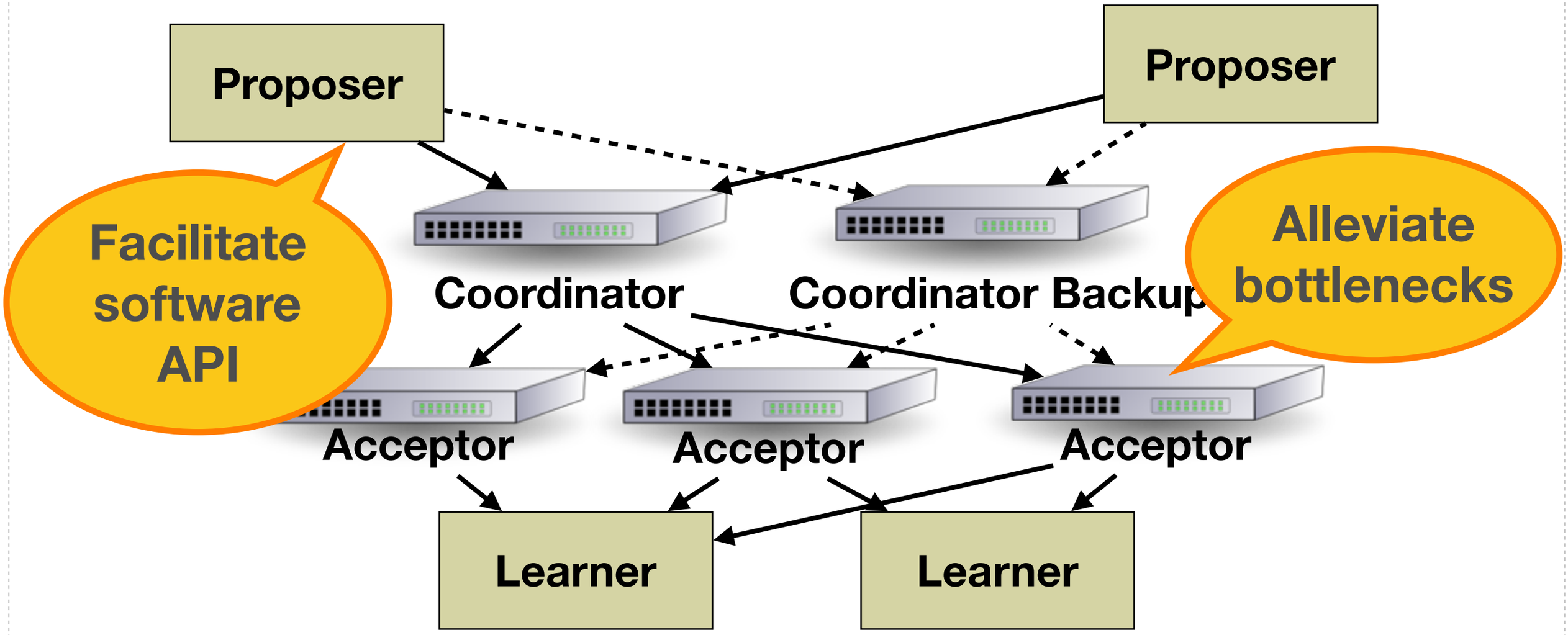
Design Goals 2: Alleviate Bottlenecks





Coordinator and acceptors are to blame!

Hardware/Software



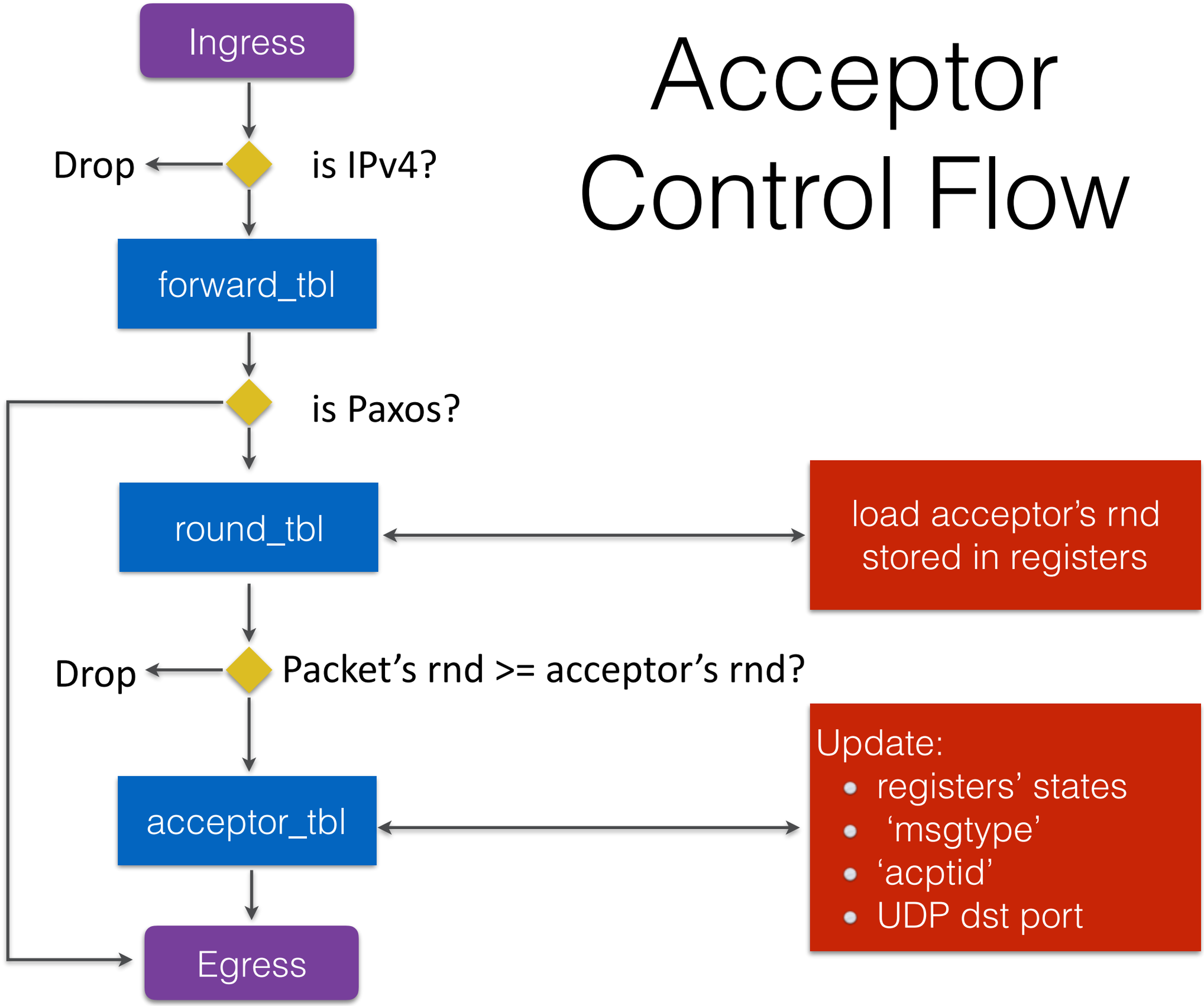
Challenge: map Paxos logic into stateful forwarding decisions

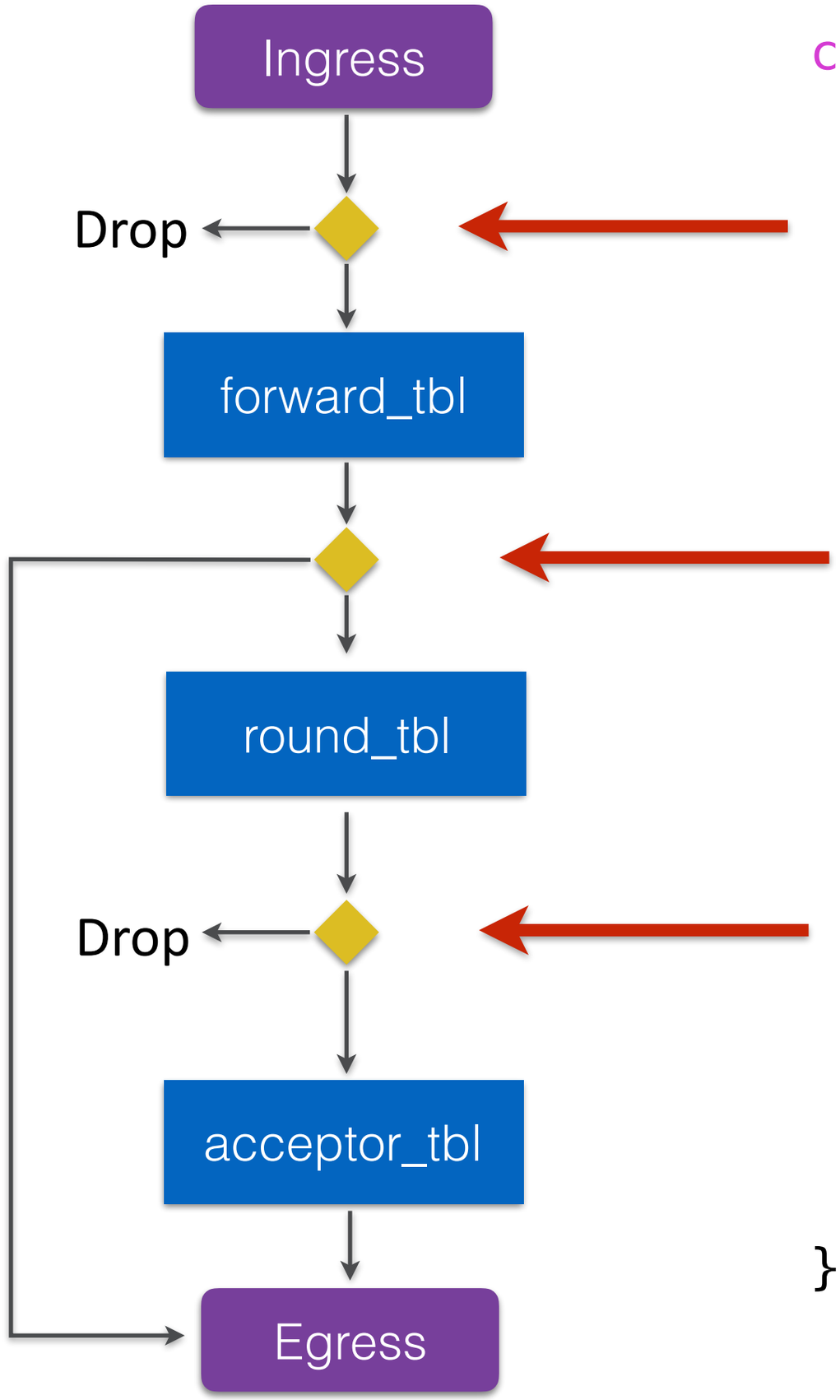
NetPaxos: Header Definition & Parser

```
header_type paxos_t {
    fields {
        msgtype  : 16;
        inst     : 32;
        rnd      : 16;
        vrnd     : 16;
        acptid   : 16;
        paxosval : 256;
    }
}
```

```
parser parse_ethernet {
  extract(ethernet);
  return parse_ipv4;
}
parser parse_ipv4 {
  extract(ipv4);
  return parse_udp;
}
parser parse_udp {
  extract(udp);
  return select(udp.dstPort) {
    PAXOS_PROTOCOL: parse_paxos;
    default: ingress;
  }
}
parser parse_paxos {
  extract(paxos);
  return ingress;
}
```

Acceptor Control Flow





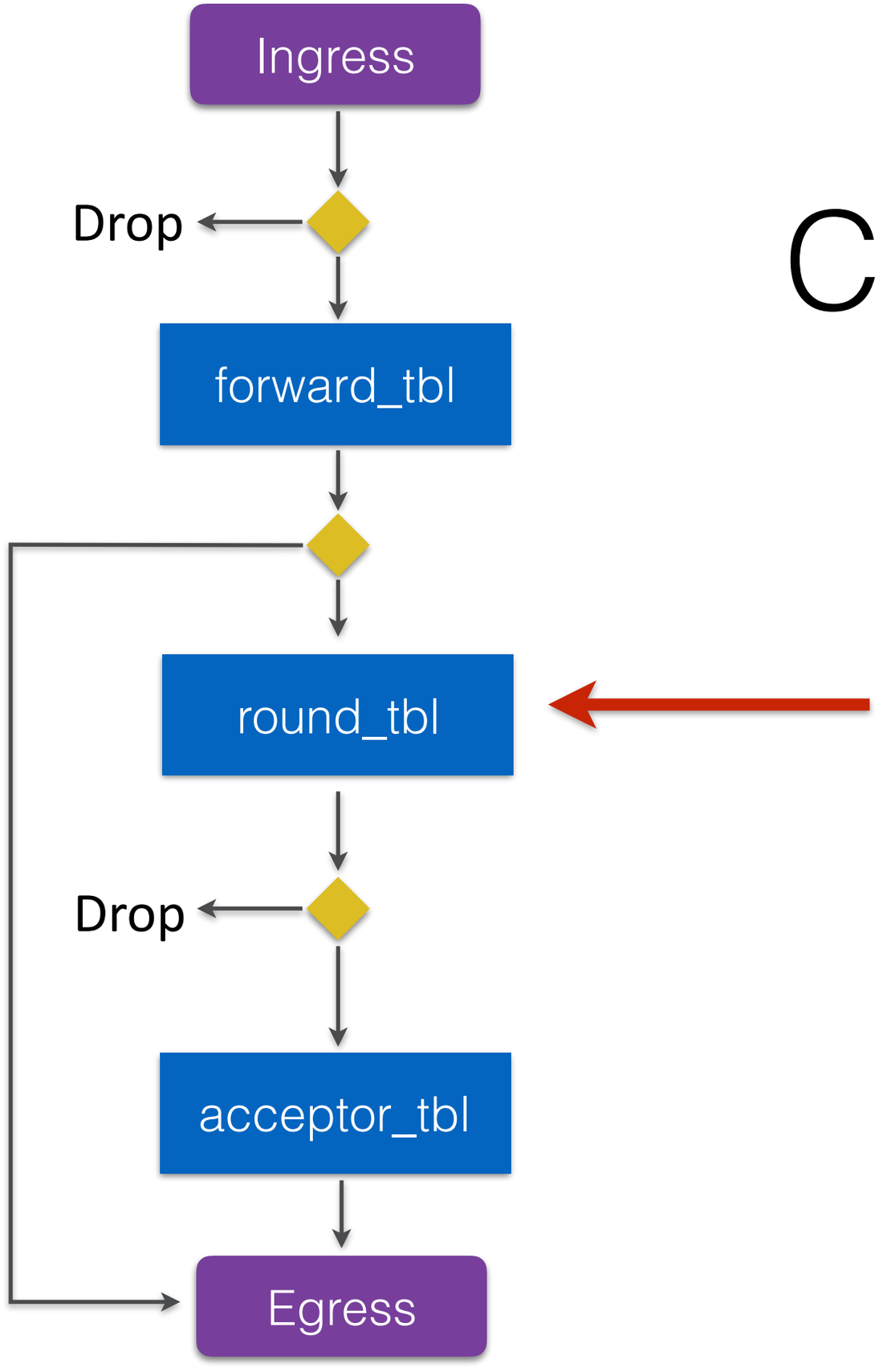

```
control ingress {

  if (valid(ipv4)) {
     apply(forward_tbl);
  }

  if (valid(paxos)) {
     apply(round_tbl);


     if(paxos.rnd >= current.rnd){
        apply(acceptor_tbl);
     }
  }
}
```

Acceptor Control Flow



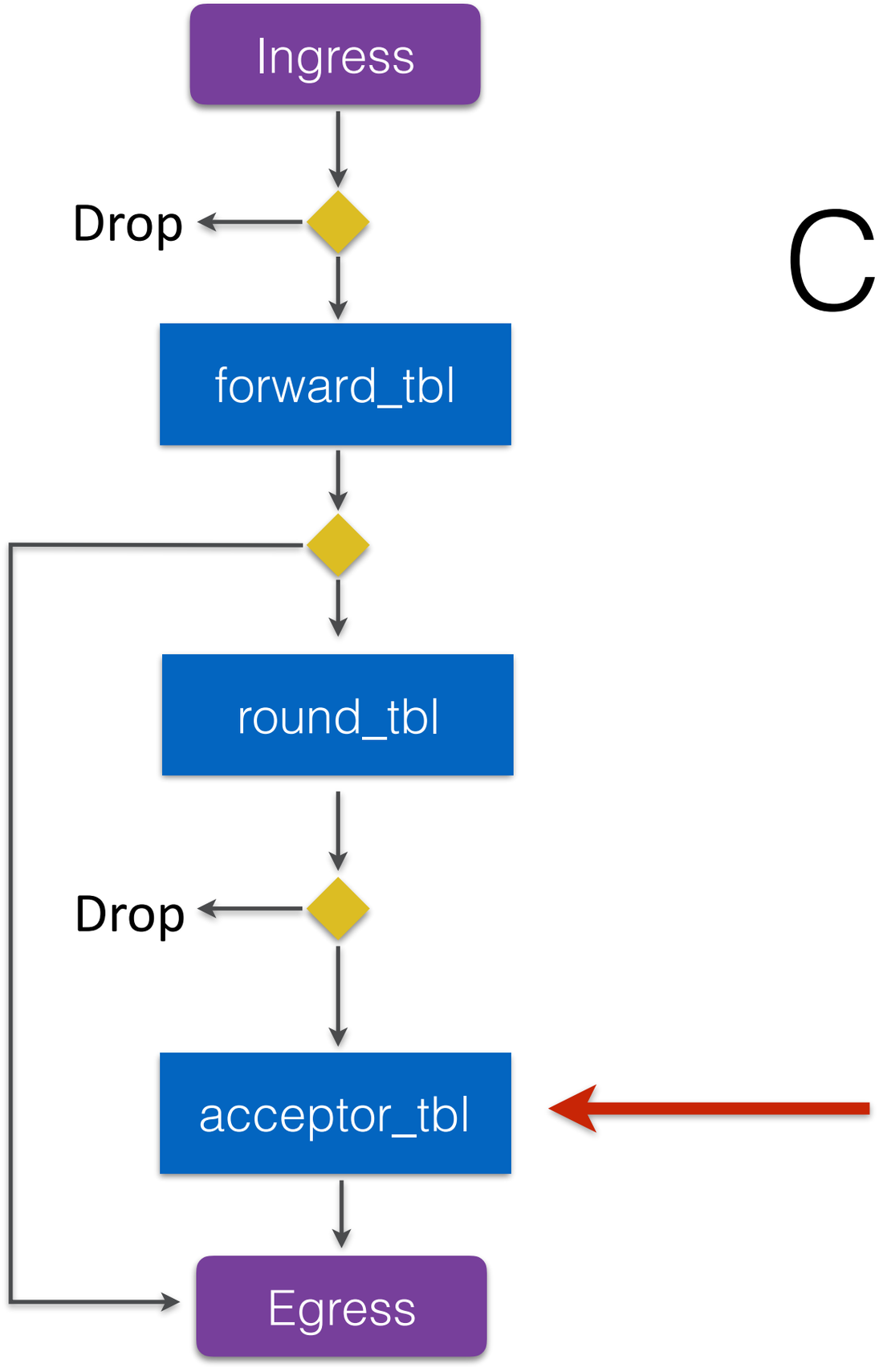
round_tbl

```
// uint16_t rounds_regs[64000];
register rounds_reg {
    width : 16;
    instance_count : 64000;
}

action read_round() {
    // uint16_t current.round = rounds_reg[paxos.inst]
    register_read(current.round, rounds_reg, paxos.inst);
}

table round_tbl {
    actions { read_round; }
    size : 1;
}
```

Acceptor Control Flow



acceptor_tbl

```
action handle_2a(learner_port) {
  // rounds_reg[paxos.inst] = paxos.rnd
  register_write(rounds_reg, paxos.inst, paxos.rnd);

  // vrounds_reg[paxos.inst] = paxos.rnd
  register_write(vrounds_reg, paxos.inst, paxos.rnd);

  // values_reg[paxos.inst] = paxos.rnd
  register_write(values_reg, paxos.inst, paxos.paxosval);

  register_read(paxos.acptid, acceptor_id, 0);
  modify_field(paxos.msgtype, PAXOS_2B);
  modify_field(udp.dstPort, learner_port);
}

table acceptor_tbl {
  reads { paxos.msgtype : exact };
    actions { handle_1a; handle_2a };
}
```

Implementation

- Source code
  - Proposer and learner written in C
  - Coordinator and acceptor written in P4
- 4 Compilers
  - P4C
  - P4FPGA
  - Xilinx SDNet
  - Netronome SDK
- 4 Hardware target platforms
  - NetFPGA SUME (4x10G)
  - Netronome Agilio-CX (1x40G)
  - Alpha Data ADM-PCIE-KU3 (2x40G)
  - Xilinx VCU109 (4x100G)
- 2 Software target platforms
  - Bmv2
  - DPDK (work in progress)

P4 Compilers

Compiler	Target	Remark
P4C	Software Switch	Supports most of the P4 constructs
P4@ELTE	DPDK	Does not support register operations. Limits field length to 32 bits
P4FPGA	FPGAs	Must write modules for unsupported P4 constructs
Xilinx SDNet	FPGAs	Does not support register operations. Requires a wrapper for the packet stream
Netronome SDK	Netronome ISAs	Works only with Netronome devices. Custom actions can be written in Micro-C
Barefoot Capilano	Barefoot Tofino	Tbps switch

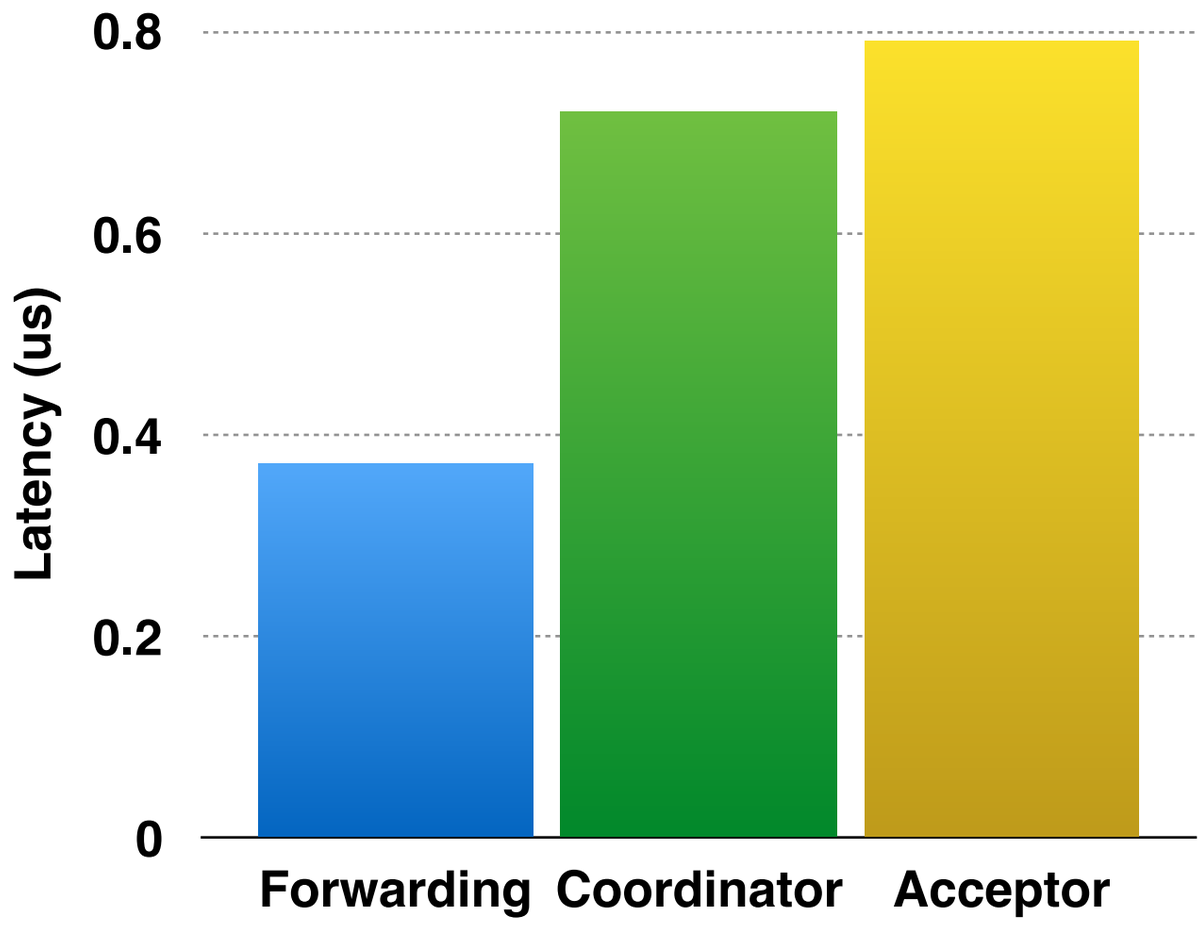
Evaluation

Experiment: What is the Absolute Performance?

- Run Coordinator / Acceptor in isolation
- Testbed:
  - NetFPGA SUME board in a SuperMicro Server
  - A Packet generator for offering load

Absolute Performance

- Measured on NetFPGA SUME using P4FPGA
- Throughput is over **9 million consensus messages / second** (close to line rate)
- Little overhead latency compared to simply forwarding packets

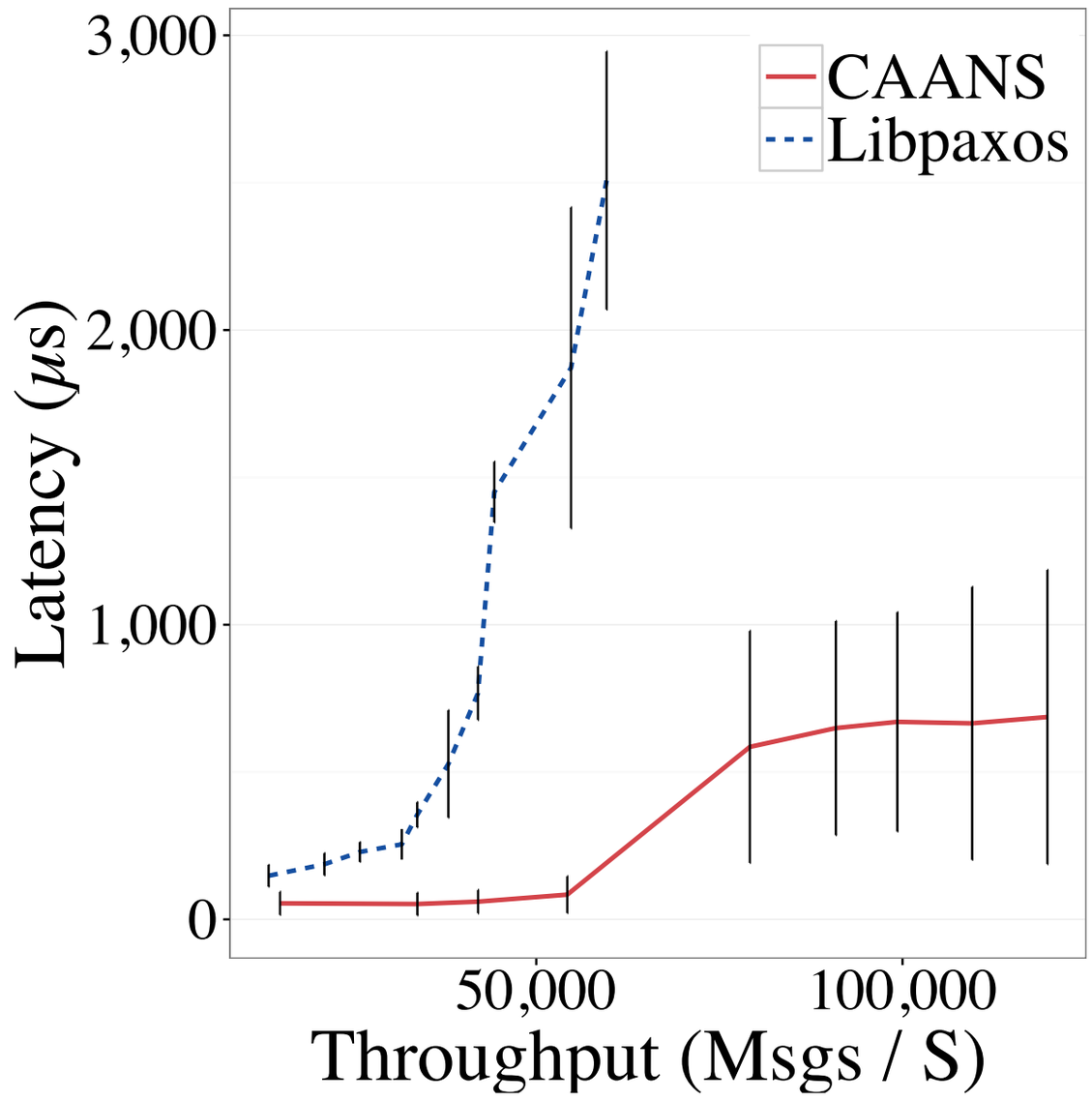


Experiment:
What is the End-to-End Performance?

- Comparing NetPaxos to a software-based Paxos (Libpaxos)
- Testbed:
  - 4 NetFPGA SUME boards in SuperMicro Servers
  - An OpenFlow-enable 10 Gbps switch (Pica8 P-3922 switch)

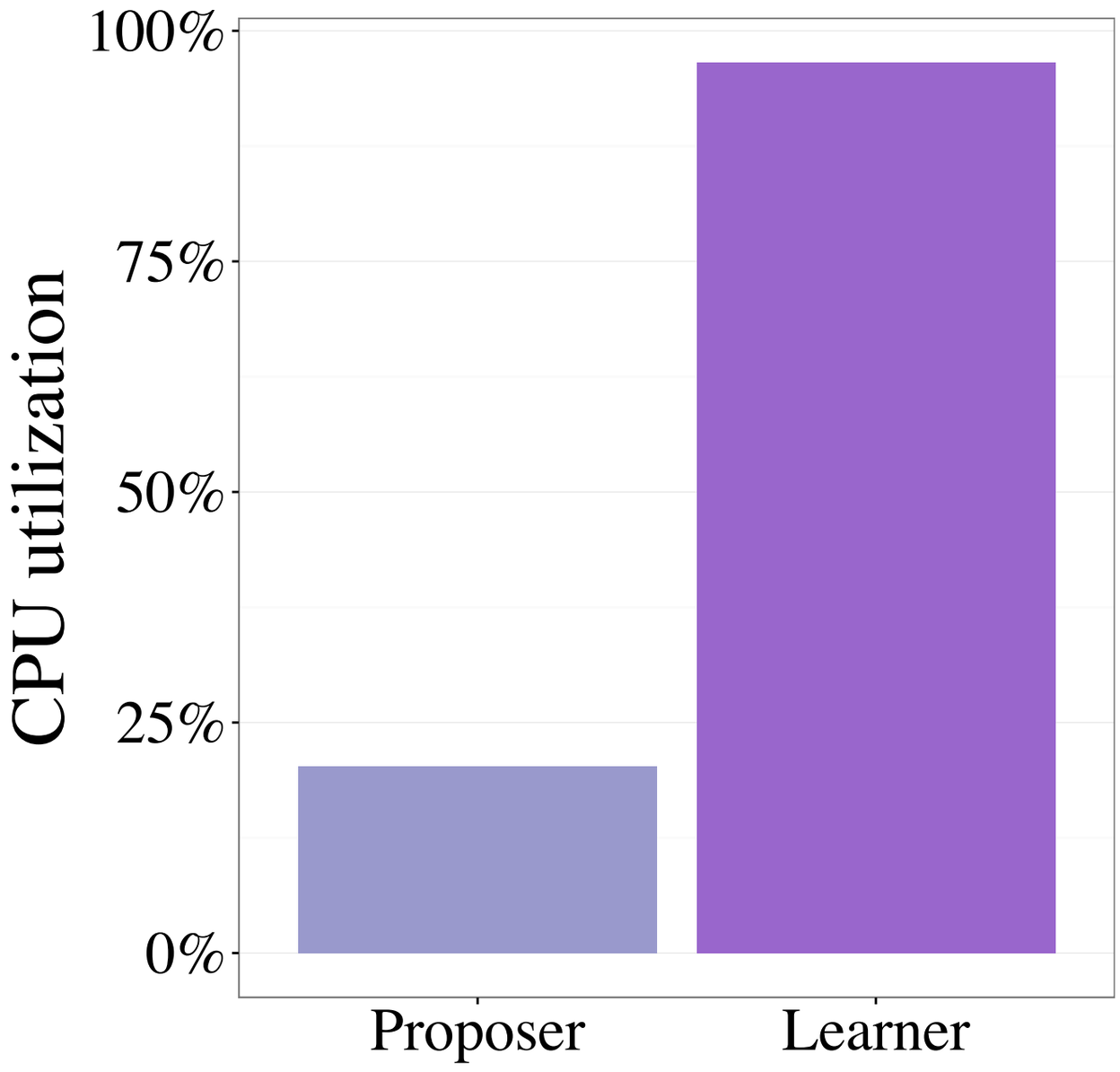
End-to-End Performance

- 2.24x throughput improvement over software implementation
- 75% reduction in latency
- Similar results when replicating LevelDB as application



Next Steps

- **We make consensus great again!**
- **The ball is now in the application developer's court**
- **Suggests direction for future work**



Lessons Learned

Outlook

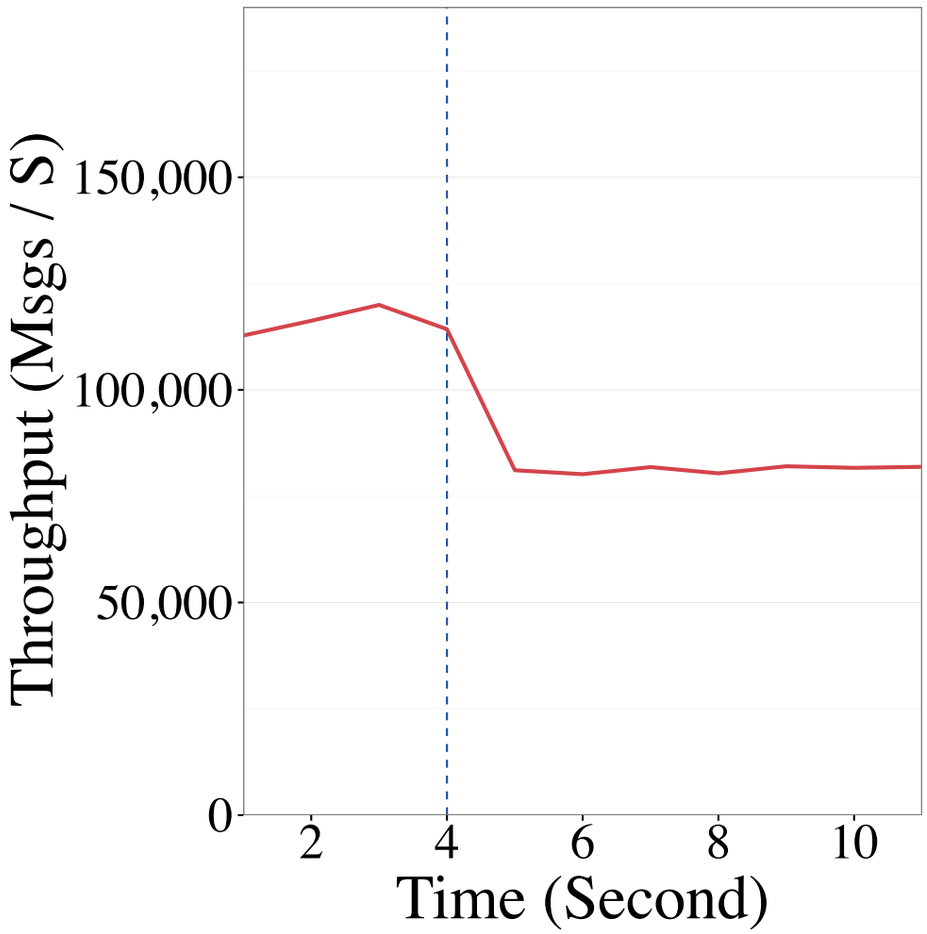
- The performance of consensus protocols has a dramatic impact on the performance of data center applications
- Moving consensus logic into network hardware results in significant performance improvements

“a HUGE wave of consensus messages is approaching”

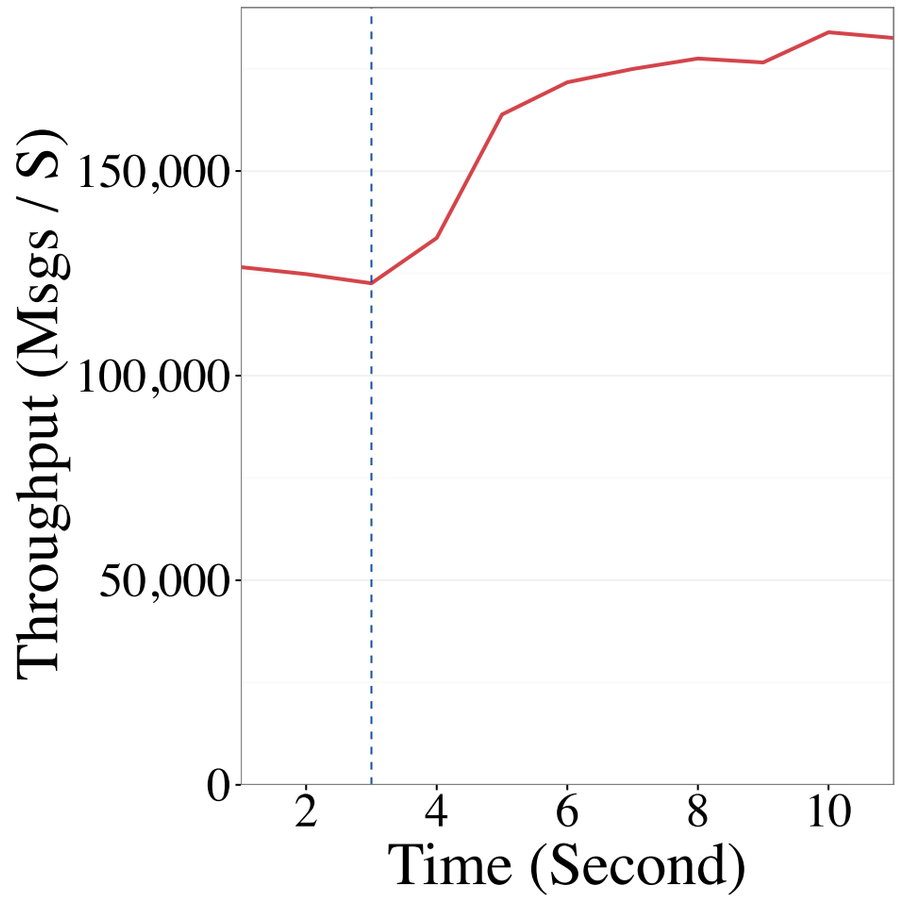
http://www.inf.usi.ch/faculty/soule/netpaxos.html

Questions & Answers

Performance After Failure



Coordinator failure
with software backup



Acceptor failure

End-to-End Experiment
NetPaxos Setup