Open-NFP Summer Webinar Series: CAANS: Consensus As A Network Service

Huynh Tu Dang, Daniele Sciascia, Pietro Bressana, Han Wang, Ki Suh Lee, Hakim Weatherspoon, Marco Canini, Fernando Pedone, and Robert Soulé
Università della Svizzera italiana (USI), Cornell University, and Université catholique de Louvain
Outline

- Introduction
- CAANS
- Demo
- Results
- Conclusion
Introduction

Many distributed problems can be reduced to consensus
- E.g., Atomic commit, leader election, atomic broadcast

Consensus protocols are the foundation for fault-tolerant systems
- E.g., OpenReplica, Ceph, Chubby, etc.,

Paxos: one of the most widely used consensus protocols
- Paxos is slow
- Optimizations: Generalized Paxos, Fast Paxos
Motivations

Network devices are becoming:

- More powerful: NFP-6xxxx, PISA chips, FlexPipe
- More programmable: custom pipeline processing
Motivations

Network devices are becoming:

- More powerful: NFP-6xxxx, PISA chips, FlexPipe
- More programmable: custom pipeline processing

High level languages:

- OpenFlow, PX, P4

Co-design networks and consensus protocols
Background: Consensus Problem
Background on Paxos Consensus

Client

submit requests

Proposer

coordinate requests

Acceptor

learn the outcome

Learner
Background on Paxos Consensus

Client

Coordinator

Distinguished Proposer

prepare(2)

Acceptor

Acceptor

Acceptor
Case 1: acceptor has NOT accepted any message previously
Background on Paxos Consensus

Case 2: acceptor has ACCEPTED proposal(1,x)

Client

Coordinator

Acceptor

Acceptor

Acceptor

propose(2, 1, x)

proposal: 2
value: x
Background on Paxos Consensus

Client → Coordinator

Coordinator → Acceptor

promise(2, ø, ø)

request(v)

Coordinator → Acceptor

proposal: 2

value: v

Coordinator → Acceptor

Coordinator → Acceptor
Background on Paxos Consensus

Client \rightarrow Coordinator

Coordinator \rightarrow Acceptor

request(v) \rightarrow accept(2,v)

proposal: 2
value: v

Acceptors
Background on Paxos Consensus

Client \rightarrow Coordinator

request(v) \rightarrow accept(2,v) \rightarrow accepted(2,v)

Coordinator \rightarrow Acceptor

proposal: 2 value: v

Coordinator \rightarrow Acceptor \rightarrow Learner

Coordinator \rightarrow Learner

Coordinator \rightarrow Learner
Bottleneck Experiment Setup

Coordinator + Acceptor + Learner

Client + Acceptor + Learner

Acceptor + Learner
Coordinator Bottleneck

Client offered load at maximum rate
Messages are 102 Bytes
Minimum latency is 96 µs
To increase replication factor, we launch multiple learner processes on each servers.
CAANS: Consensus As A Network Service
Consensus / Network Design Space

Exploit
Better Service
Guarantees

No lost messages

FiFO Delivery

Best effort

Traditional Paxos

Stateless processing

Persistent storage

stateful processing

Implement Paxos in the network
Design Goals

A network consensus library:

- Compatible with the software library
- Independent targets
- High consensus throughput
- Low end-to-end latency
Consensus as a network service

Coordinate requests
Consensus as a network service
P4Paxos Data Flow

Ingress

Parser

- Eth
- IPv4
- UDP
- Paxos

Control

- forward_tbl
- round_tbl
- paxos_tbl

Actions

- forward
- read_round
- drop
- handle_phase1a
- handle_phase2a
P4Paxos Data Flow

N: proposal number

forward_tbl → forward

is Paxos?

round_tbl → read_round

drop

Packet’s N >= Acceptor’s N?

paxos_tbl → handle_phase1a

handle_phase2a

Egress

https://github.com/open-nfpsw/NetPaxos
DEMO: Debugging Paxos Acceptor
Paxos Acceptor

- Store Accept Message in Registers
- Modify Paxos header
- Read accepted proposal from Registers
Results: Absolute Performance
Result: Forwarding Latency

**Latency (ns)**

- **Agilio-CX**
- **NetFPGA SUME**

<table>
<thead>
<tr>
<th>Latency (ns)</th>
<th>Agilio-CX</th>
<th>NetFPGA SUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>300</td>
<td>720</td>
<td>720</td>
</tr>
<tr>
<td>600</td>
<td>810</td>
<td>790</td>
</tr>
</tbody>
</table>

*Numbers are provided by other sources*

---

Packet Generator

DUT

software 96μs vs. hardware 340ns
Experiment Setup

Coordinator & Acceptors:
Libpaxos: software processes
*CAANS: hardwares

* https://github.com/usi-systems/p4paxos
Experiment Setup

Clients & Learners:
software processes
End-to-End Latency

<table>
<thead>
<tr>
<th>Throughput (Messages/Second)</th>
<th>Latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>100</td>
</tr>
<tr>
<td>25000</td>
<td>100</td>
</tr>
<tr>
<td>30000</td>
<td>300</td>
</tr>
<tr>
<td>90000</td>
<td>200</td>
</tr>
<tr>
<td>120000</td>
<td>300</td>
</tr>
</tbody>
</table>

- **Libpaxos**
- **CAANS**
CAANS: Fault Tolerance

Throughput (Msg/S)

- Coordinator Failure
- Acceptor Failure

Links are down at the 3rd second.
Coordinator failure: requests are switched to a software coordinator.
Acceptor failure: Paxos tolerates failure of a minority of nodes.
Summary
Consensus is important in distributed systems

- Optimizations exploit network assumptions

SDN allows network programmability
- OpenFlow, P4

CAANS: Co-design Network and Consensus
- Implement Paxos in network devices
- Achieve high performance
Future Work

• Checkpoint Acceptor’s log
• Fast fail-over to software/hardware coordinator
• Use kernel-bypass API to increase learner’s performance
• Full-fledged deployment in traditional networks
Acknowledgements

• Thank Bapi Vinnakota, Mary Pham, Jici Gao, and Netronome for providing us with a hardware testbed, and support with using their toolchain

• Thank Gordon Brebner and Xilinx for donating two SUME boards, providing access to SDNet, performing measurements

• Thank Google! Faculty award helped support this research
The End

Huynh Tu Dang,
Università della Svizzera italiana (USI)

tudang.github.io
NetPaxos

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