HY-559
Infrastructure Technologies for Large-Scale Service-Oriented Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy559
Coordination services

• API for
  – Storing and querying cluster state
    • Live machines, association to services, roles
  – Express interest in conditions, notifications

• High availability and data consistency
  – Replication
  – Order on state updates

• Google Chubby (Paxos), Apache ZooKeeper (ZAB)
Order on state updates
Paxos algorithm

- Way to build fault-tolerant distributed systems
  - Replicated state machines (RSM)

- Consensus via message exchange
  - Asynchronous: no timing guarantees
  - Network can delay, reorder, lose (but not corrupt) packets

- Can guarantee safety
  - Replicas will agree on a single value

- Need additional assumptions to ensure progress
Informally

- Three roles: Proposer, acceptor, learner
- Simplest, but fault-intolerant solution: single acceptor
- With >1 acceptors, agreement by a majority required
- If single value proposed, that value should be chosen
  - Thus, an acceptor must accept the first value proposed to it
- However, this may lead to fragmented electorate
  - Multiple proposals by each proposer should be possible
  - Identify each proposal by a unique integer N
Informally

• After consensus, an acceptor cannot change its mind
  – A value is chosen when single proposal with that value accepted by a majority of the acceptors

• Allow multiple proposals to be chosen, but guarantee that all chosen proposals have the same value
• Be able to agree in the presence of up to $f$ failures
  • $2f+1$ nodes
  • Agreement when majority ($f +1$) agrees on a value
Need to try to get a majority to accept
Informally

- Allow multiple proposals to be chosen, but guarantee that all chosen proposals have the same value.

- If proposal $N$ with value $\nu$ is chosen, every higher numbered proposal issued by any proposer should have value $\nu$.

- A proposer wanting to issue a proposal numbered $N$ must learn the highest-numbered proposal $< N$ (if any) that has been or will be accepted by a majority.
Informally

• A proposer wanting to issue a proposal numbered $N$ must learn the highest-numbered proposal $< N$ (if any) that has been or will be accepted by a majority
  – Easy to learn about values already accepted
  – Hard to predict the future

• Control the future by extracting a promise that there will not be any acceptances of proposals $< N$
Paxos – phase 1

<table>
<thead>
<tr>
<th>Client</th>
<th>Prepare N</th>
<th>Proposer</th>
<th>Acceptor</th>
<th>Learner</th>
<th>Prepare N</th>
<th>Proposer</th>
<th>Acceptor</th>
<th>Learner</th>
<th>Value v'</th>
</tr>
</thead>
</table>

- Highest-numbered prepare request acknowledged
- Highest-numbered proposal accepted
- Written to stable store
Paxos – phase 2

- Client
- Propose $N, v$
- Proposer
- Acceptor
- Learner
- Propose $N, v$
- Value $v'$
Paxos – communicate agreement

- Client
- Proposer
- Acceptor
- Learner

Decide $N, v$

Value $v'$
Paxos – majority learns outcome
Paxos – learning chosen value

client

proposer
acceptor
learner

proposer
acceptor
learner

proposer
acceptor
learner

client

prepare \( N' \), value \( v' \)

prepare \( N' \), \( N' \), _

prepare \( N' \), \( N' \), _

prepare \( N' \), \( N' \), _
Paxos – propagate chosen value
Paxos – everyone learns outcome
Example

ballots: xxxx00  xxxx01  xxxx02
proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (\_,\_) \)  \( (\_,\_) \)  \( (\_,\_) \)

proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (0,v_0) \)  \( (1,\_) \)  \( (1,\_) \)

proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (0,\_) \)  \( (0,\_) \)  \( (\_,\_) \)

proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (0,v_0) \)  \( (1,\_) \)  \( (1,\_) \)

proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (0,\_) \)  \( (0,\_) \)  \( (\_,\_) \)

proposers  \( v \)  \( v' \)  \( v'' \)
acceptors  \( (0,v_0) \)  \( (2,v'_1) \)  \( (2,\_) \)
Example (contd.)
Lamport: implementing a state machine

• How to run multiple instances of Paxos
  – Assume the existence of a distinguished proposer (leader)
  – A leader will run Paxos for a number of instances
  – The leader may crash, at which point there may be gaps in the chosen instances (1-134, 138, ..)
  – A new leader will try to fill in those slots or propose \textit{no-op}
  – As soon as gap fills, commands can be executed

• Multi-Paxos
  – New leader: execute phase 1 for infinitely many instances
  – Acceptors can respond with reasonably short messages
  – Cost of Paxos effectively the cost of executing phase 2
Multi-Paxos

- New leader @N
- Learn accepted values for past instances

If a majority has not accepted anything for instances > I

Skip prepare phase until a propose is rejected!
Multi-Paxos

Servers play all roles

Replicas write to disk prior to sending ACK