HY590.45
Modern Topics in Scalable Storage Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy590-45
Raft

- Consensus algorithm for log replication
- Easier to understand compared to Multi-Paxos
Replicated state machine architecture

Raft
Server states

- **Follower**
  - Starts up
  - Times out, starts election
  - Discovers current leader or new term

- **Candidate**
  - Times out, new election
  - Discovers server with higher term

- **Leader**
  - Receives votes from majority of servers
Terms (epochs)

- **term 1**: election
- **term 2**: normal operation
- **t3**: no emerging leader
- **term 4**
Log entries

leader

followers

committed entries

Raft
Possible states of followers

Raft

Possible followers

leader for term 8

log index

1 2 3 4 5 6 7 8 9 10 11 12

(a) 1 1 1 4 4 5 5 6 6 6 6
(b) 1 1 1 4
(c) 1 1 1 4 4 5 5 6 6 6 6 6
(d) 1 1 1 4 4 5 5 6 6 6 6 7 7
(e) 1 1 1 4 4 4 4
(f) 1 1 1 2 2 2 3 3 3 3 3 3
When is an entry committed?

Leader S1, term 2

Leader S1, term 4

Leader S5, term 5

Leader S5, term 3

Leader S1, term 4

Raft
Properties

**Election Safety:** at most one leader can be elected in a given term. §5.2

**Leader Append-Only:** a leader never overwrites or deletes entries in its log; it only appends new entries. §5.3

**Log Matching:** if two logs contain an entry with the same index and term, then the logs are identical in all entries up through the given index. §5.3

**Leader Completeness:** if a log entry is committed in a given term, then that entry will be present in the logs of the leaders for all higher-numbered terms. §5.4

**State Machine Safety:** if a server has applied a log entry at a given index to its state machine, no other server will ever apply a different log entry for the same index. §5.4.3
Reconfiguration

Raft

![Diagram of reconfiguration process with servers and time axis showing two disjoint majorities](Image)
Joint consensus

$C_{old}$ can make decisions alone

$C_{new}$ can make decisions alone

$C_{old,new}$

$C_{old,new}$ entry committed

$C_{new}$ entry committed

leader not in $C_{new}$ steps down here

Raft
Log compaction - snapshots

1  2  3  4  5  6  7
1 x←3 1 y←1 1 y←9 2 x←2 3 x←0 3 y←7 3 x←5

snapshot
last included index: 5
last included term: 3
state machine state:
x ← 0
y ← 9

committing entries

log index
before

} after

Raft
Time to detect and replace crashed leader

Timing requirement

broadcastTime \ll electionTimeout \ll MTBF

Raft