Lecture 14: Practical Byzantine Fault Tolerance (PBFT)

CS 539 / ECE 526

Distributed Algorithms
Paxos Summary

• Most widely known/used and first practical crash fault tolerant protocol
  – Replication, psync, f < n/2 crash
  – Leader-based, quorum intersection, lock ranking
PBFT

- Most widely known/used and first practical Byzantine fault tolerant protocol
  - Replication, psync, $f < n/3$ Byzantine
  - Leader-based, quorum intersection, lock ranking
  - Independently developed from Paxos by Castro and Liskov in 1999, but share many key concepts

- We will modify Paxos into PBFT
  - What obviously go wrong with Byzantine faults?
Paxos Protocol

• Leader (replica k % n) sends (new-view, k)
• Others reply with (status, k, x_{lck}, k_{lck})
• Leader (propose, x, k) where x is the highest locked value among the f+1 status
• Others (vote, x, k) and lock (x, k)
• Leader (success, x, k); Others commit x
Challenges for Byzantine Paxos

• Leader may equivocate (e.g., double-propose)

• Byzantine nodes can make false “claims”
  – “Previous leader is not making progress.”
  – “I am locked on value x with rank k.”
  – “x is the highest locked value I have seen.”
PBFT Steady State

- Leader (proposes, x, k), replicas (vote1, x, k)
- Upon n-f (vote1, x, k), lock (x, k) and send (vote2, x, k)
- Upon n-f (vote2, x, k), commit x

Replica 1 (Leader)
Replica 2
Replica 3
Replica 4

propose
vote1
vote2

n = 3f+1
PBFT Steady State

- Two rounds of all-to-all voting
- When a replica locks, it has a certificate, i.e., \(2f+1\) signed \((\text{vote}_1, x, k)\) from distinct replicas

- Can still use all-leader-all voting
  - But no longer strictly better than all-to-all
  - Leader must forward certs, so fewer (linear) but longer (linear) msgs, still \(O(n^2)\) bits in total
    - ... unless using threshold sig, down to linear bits
- All-to-all voting does not need sigs in steady state (important at the time, but less important today)
PBFT Safety and Liveness

• Safety within view: quorum intersection
  – Two quorums of 2f+1 intersect at f+1 → there cannot be two proposals both certified

• Safety across views: hard part (later)

• Liveness: honest leader during synchrony
PBFT View Change

- Every “claim” needs to be “backed up” by signed msgs from sufficiently many replicas
  - New leader cannot step up at will
  - Replica reported locks need certificates
  - Leader’s claimed highest lock needs proof
PBFT View Change

• If suspecting leader $k-1$, send $(\text{blame}, k-1)$ to all

• New leader sends $(\text{new-view}, k, \{(\text{blames}, k-1)\})$

• Replicas send $(\text{status}, k, x_{\text{lck}}, k_{\text{lck}}, \{(\text{vote1}, x_{\text{lck}}, k_{\text{lck}})\})$ to leader $k$

• Leader sends $(\text{propose}, x, k, \{(\text{status}, k, ..., \{\text{vote1}, ...\})\})$ where $(x, k)$ is the highest locked value among $2f+1$ status msgs
PBFT View Change

- **Blame** and **status** can be sent together
- **new-view** and **propose** can then also be merged
- But it may aid understanding to treat them separately
Safety Across Views

• One replica commits $x$ in view $k$
  $\rightarrow$ $2f+1$ replicas voted and locked $(x, k)$
  $\rightarrow$ $f+1$ of them are honest
  $\rightarrow$ Leader $k+1$ presents $2f+1$ status (locks), must include one $(x, k)$, which is highest
  $\rightarrow$ Leader $k+1$ re-proposes $x$. No other value can be voted or locked in view $k+1$
  $\rightarrow$ Leader $k+2$ presents (status) locks, at least one $(x, k)$, still highest, re-proposes $x$
  $\rightarrow$ ......
PBFT Efficiency

• Steady state: 3 rounds, $O(n^2)$ communication
• View change: 2 (4) rounds
• View change communication?
  – n-to-n  *blames* of size $O(1)$
  – 1-to-n  *new-view* of size $O(n)$
  – n-to-1  *status* of size $O(n)$ (since they contain certs)
  – 1-to-n  *propose* of size $O(n^2)$ (contains n status)

  – Total: $O(n^2)$ msgs and $O(n^3)$ bits
PBFT Original Notation

• Original notation FYI:
  – blame + status = view-change
  – new-view + propose = preprepare
  – vote1 = prepare
  – vote2 = commit
PBFT Summary

• Most widely known/used and first practical Byzantine fault tolerant protocol
  – Replication, psync, f < n/3 Byzantine
  – Leader-based, quorum intersection, lock ranking
  – $O(n^2)$ steady state, $O(n^3)$ view change

• We skipped many subtle details (e.g., multi-slot is quite tricky)

• Many improvements, active research area
  – Most significant: Linear View Change [Kwon, 2014]
Linear View Change (LVC)

• If suspecting leader k-1, send \((\text{blame, } k-1)\) to all
• New leader sends \((\text{new-view, } k, \{(\text{blames, } k-1)\})\)
• Others send leader \((\text{status, } k, x_{\text{lck}}, k_{\text{lck}}, \{(\text{vote1, } x_{\text{lck}}, k_{\text{lck}})\})\)
• Leader sends \((\text{propose, } x, k, \{(\text{vote1, } x_{\text{lck}}, k_{\text{lck}})\})\) where \((x, k)\) is the highest locked value among 2f+1 status
  – Leader is not using 2f+1 signed status to back up its proposal
  – Why is this safe?
Linear View Change (LVC)

- If suspecting leader k-1, send \((\text{blame, k-1})\) to all
- New leader sends \((\text{new-view, k, \{(blames, k-1)\}})\)
- Others send leader \((\text{status, k, x_{lck}, k_{lck}, \{(vote1, x_{lck}, k_{lck})\}})\)
- Leader sends \((\text{propose, x, k, \{(vote1, x_{lck}, k_{lck})\}})\) where \((x, k)\) is the **highest locked value** among \(2f+1\) status
  - Leader is not using \(2f+1\) signed status to back up its proposal
  - Why is this safe? Safe if others do not blindly believe the leader
- A replica refuses to vote if it has a higher lock than the certificate in the leader’s **propose** msg!
Safety Across Views with LVC

• One replica commits $x$ in view $k$
  $\rightarrow$ 2$f+1$ replicas voted and locked ($x$, $k$)
  $\rightarrow$ $f+1$ of them are honest
  $\rightarrow$ If leader $k+1$ proposes $x' \neq x$, it cannot show a certificate as high as ($x$, $k$)
  $\rightarrow$ At most 2$f$ votes for $x'$ in view $k+1$, not a cert
  $\rightarrow$ If leader $k+2$ proposes $x'' \neq x$, it cannot show a certificate as high as ($x$, $k$)
  $\rightarrow$ ……
LVC Efficiency

• View change: 2 (4) rounds

• View change communication?
  – n-to-n **blames** of size O(1)
  – 1-to-n **new-view** of size O(n)
  – n-to-1 **status** of size O(n) (contain cert)
  – 1-to-n **propose** of size O(n) (contains cert)
  – Total view change communication in bits: O(n²)

• Why is it called **Linear** View Change then?
  – With threshold signatures, cert is O(1)
  – With static view-change schedule (e.g., every epoch), can skip **blame** and **new-view** in some cases
PBFT Summary

- Most widely known/used and first practical Byzantine fault tolerant protocol
  - Replication, psync, $f < n/3$ Byzantine
  - Leader-based, quorum intersection, lock ranking
- Steady state: 3 rounds, $O(n^2)$ communication
  - 5 rounds, $O(n)$ communication with all-leader-all voting and threshold signature
- View change: 2 rounds, $O(n^3)$ communication
  - $O(n^2)$ communication with Tendermint view change
  - $O(n)$ communication further adding threshold sig and static view-change schedule