VIEWSTAMP REPLICATION: ORDERING UPDATES

44

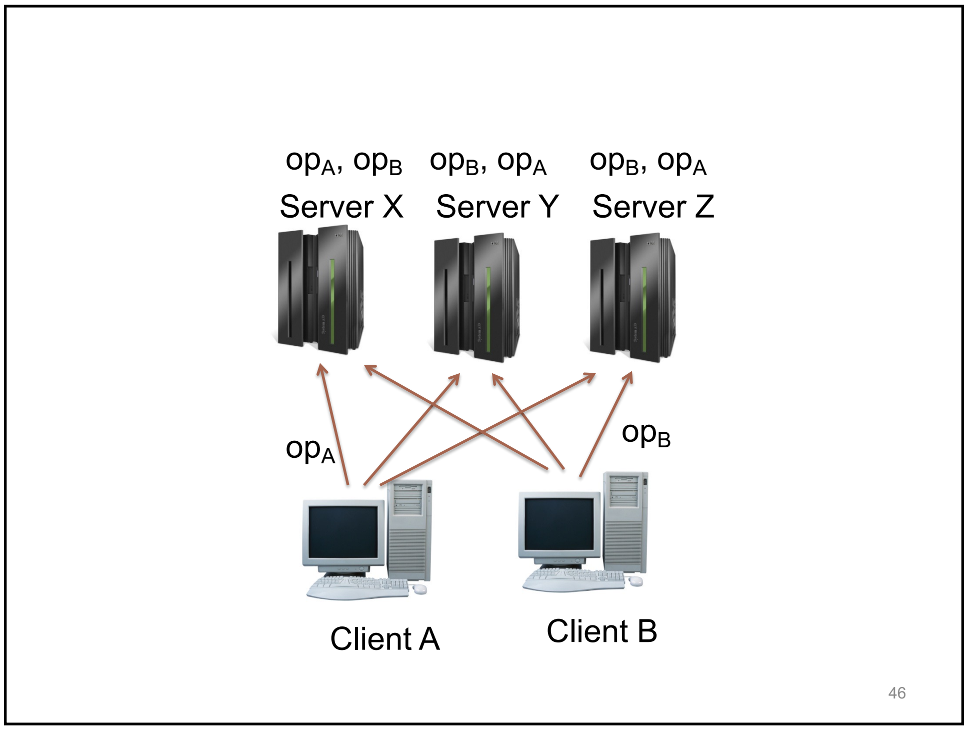
44

Quorum Consensus

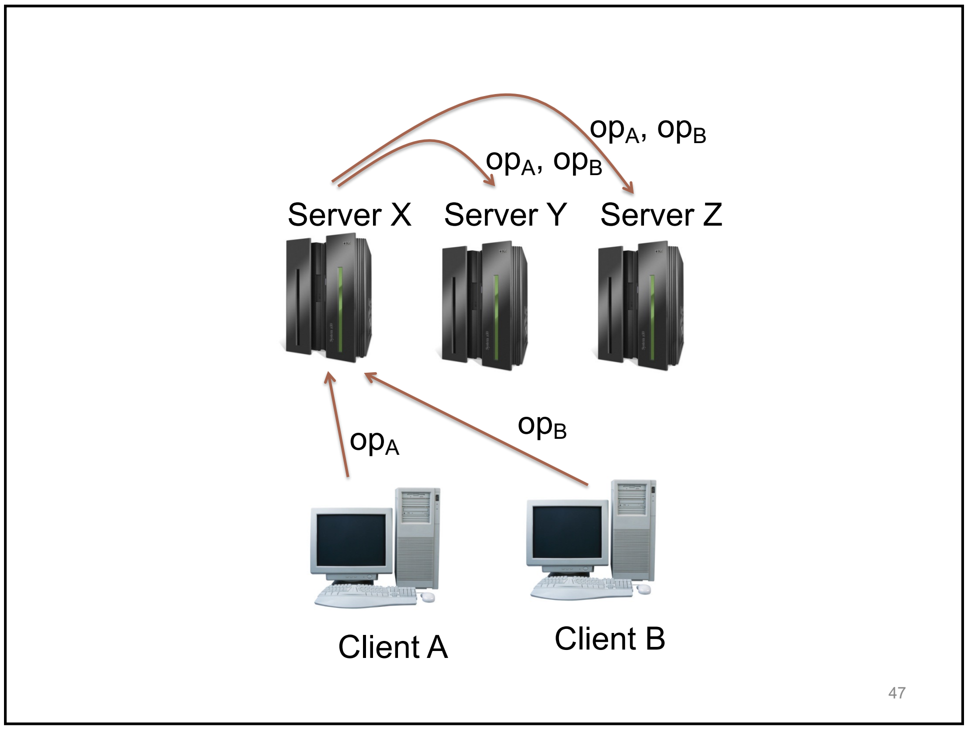
- Crash-stop failures
- Requires 2F+1 replicas
  - Operations must intersect for at least one replica
  - Want availability for both reads and writes
  - Read and write quorums of F+1 nodes

45

45


op_A, op_B
op_B, op_A
op_B, op_A
Server X
Server Y
Server Z
op_A
op_B
Client A
Client B
46

46


op_A, op_B
op_A, op_B
Server X
Server Y
Server Z
op_A
op_B
Client A
Client B
47

47

Viewstamp Replication

- Primary-backup
- System moves through a sequence of views
  - – Primary runs the protocol
  - – Replicas do a view change if it fails

48

48

Replica state

- A replica id i (between 0 and N-1)
  - – Replica 0, replica 1, …
- A view number v#, initially 0
- Primary is the replica with id

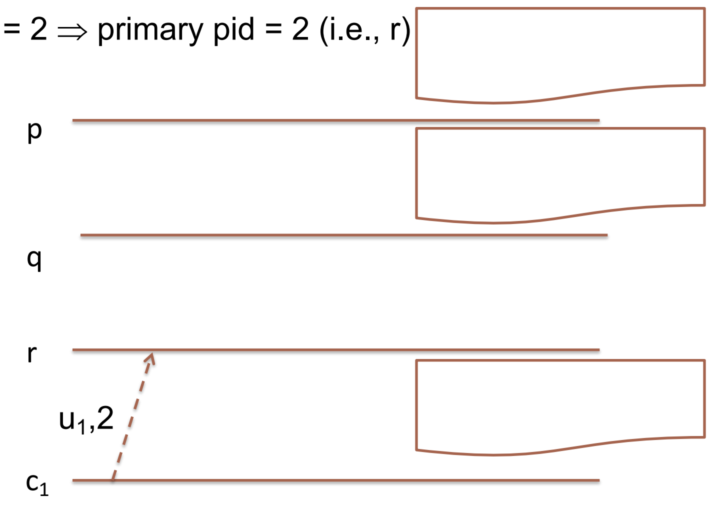
  i = v# mod N
- A log of <op, op#, status> entries
  - – Status = prepared or committed

49

49

Client knows current view

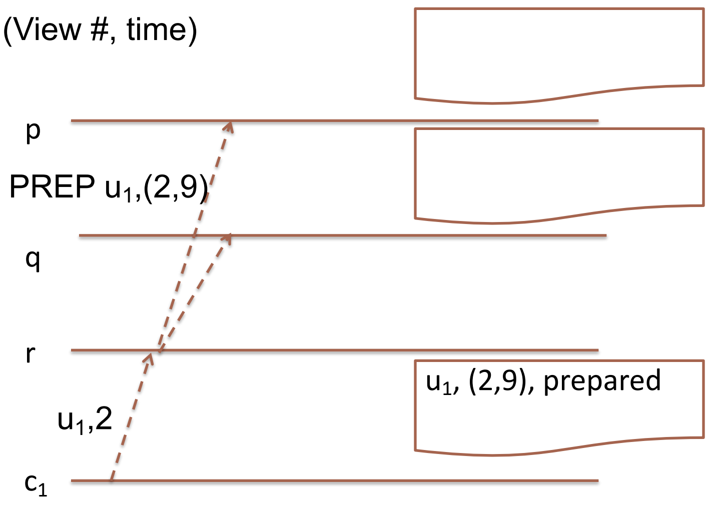
View # = 2 $\Rightarrow$ primary pid = 2 (i.e., r)



50

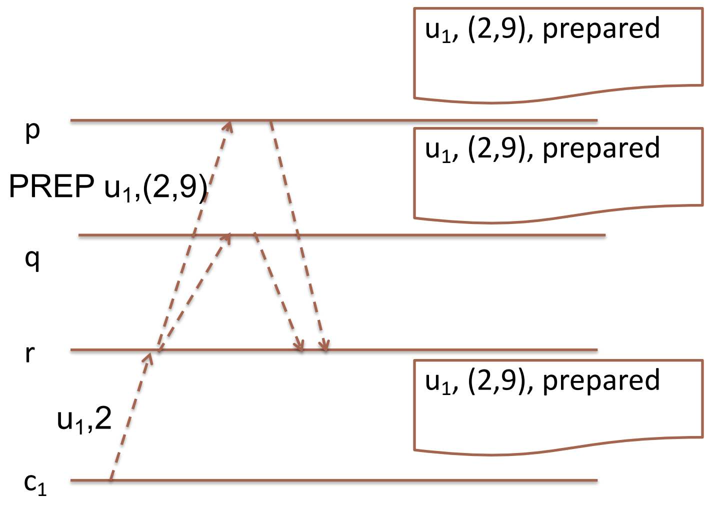
Primary chooses logical timestamp for update, e.g. LT(u_1)=9

Op # = (View #, time)



51

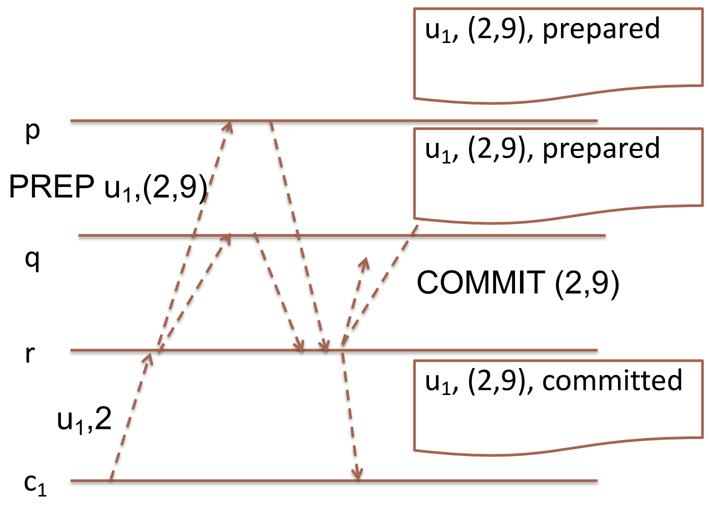
Primary waits for ≥ f replicas to respond (≥ f+1 total)



52

52

Client notified immediately after acks



53

53

VIEWSTAMP REPLICATION: VIEW CHANGE

54

54

View Changes

- Used to mask primary failures
- Replicas monitor the primary
- Replica requests next primary to do a view change

55

55

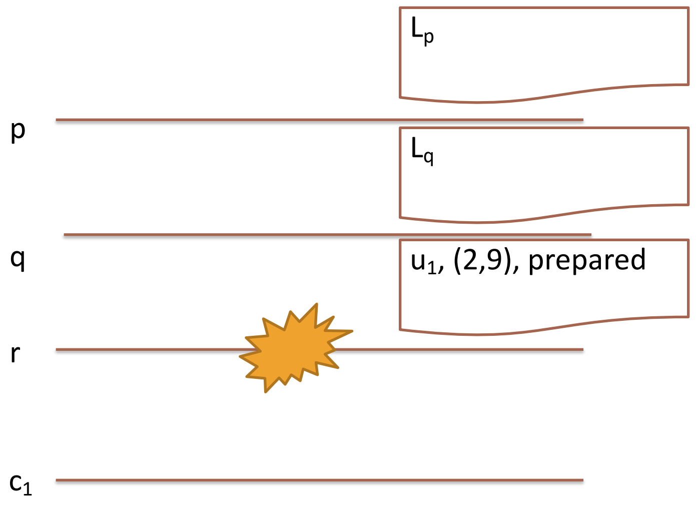
Correctness Requirement

- Operation order must be preserved by a view change
- For operations that are visible
  - executed by server
  - client received result
- An operation can be visible if it prepared at f+1 replicas
  - this is the commit point

56

56

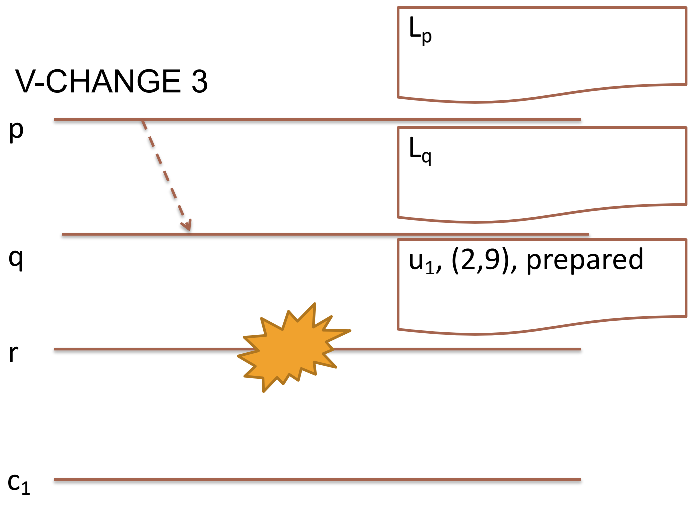
Primary r has crashed after assigning LT(u_1)=9



57

57

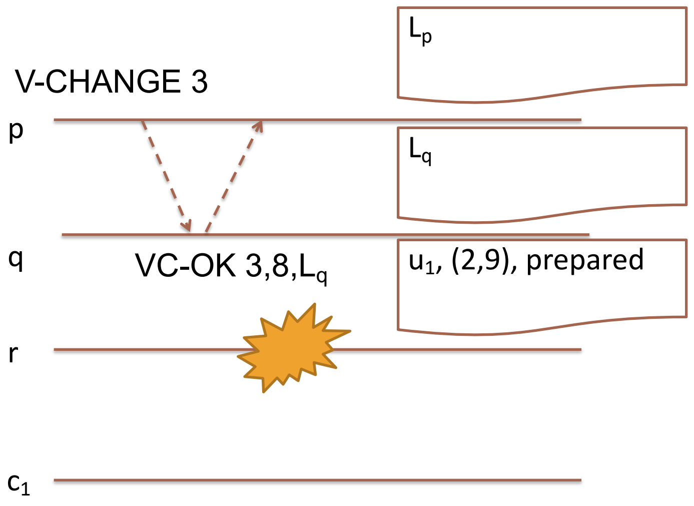
View # increments (to 3), new primary is p, start view change



58

58

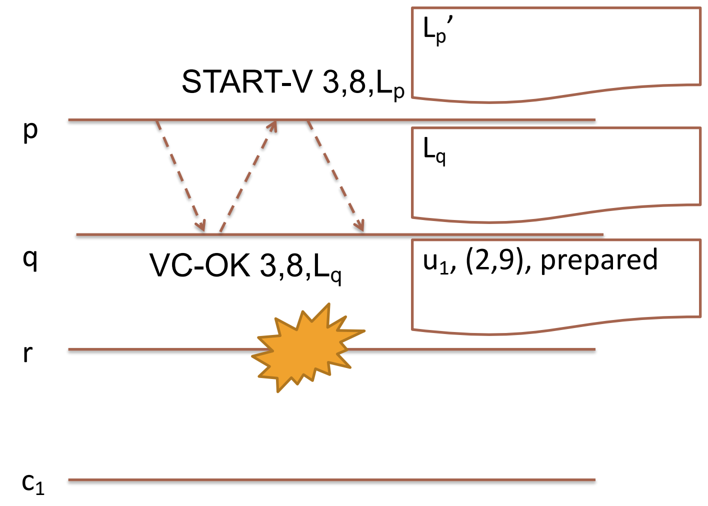
Replica q acks view change, returns last op# it knew of and its log (L_q contains any operations that p never heard of)



59

59

Primary p lets replicas know (via its log L_p') of ops that replicas were never informed of before old primary crashed



60

60

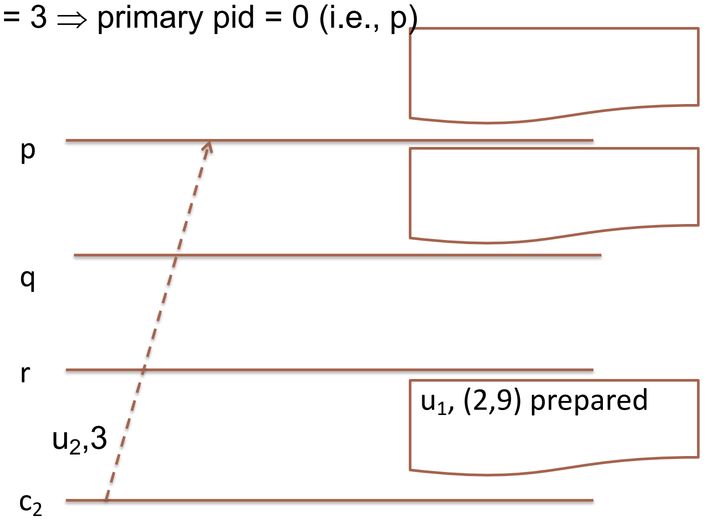
View Change

- New primary may not know of all updates from old primary
- New primary asks replicas for their logs
- Any committed operation was acked by a quorum, so must be in log of a surviving replica
  - Primary takes the max of the logs returned
  - That log has most recent updates

61

Second client c_2 knows current view #

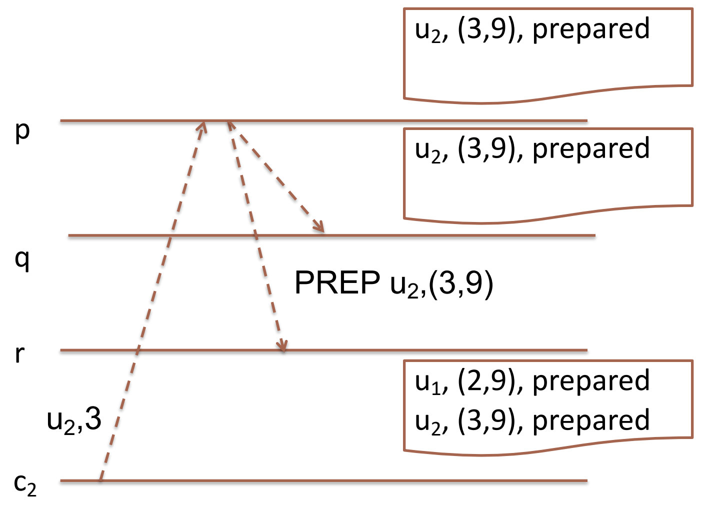
View# = 3 $\Rightarrow$ primary pid = 0 (i.e., p)



62

62

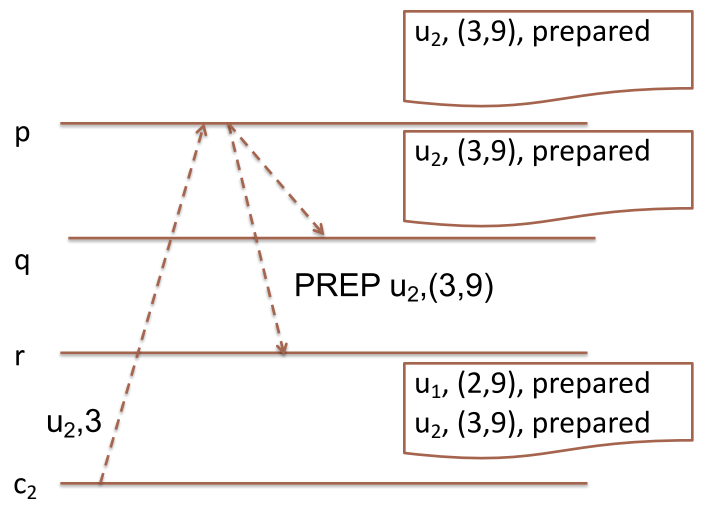
Primary p gets the other replicas to prepare to commit u_2



63

63

Viewstamp (3,9) avoids confusion between u_1 and u_2 at r (Replica r will not think p is committing u_1)



64

64

Persistent State

- Voting protocol: votes must survive failure
  - Save queue of pending updates on disk
- Viewstamp: primary can respond to client without recording commit on disk
  - View change: recover commitment from logs of surviving replicas
- Only need to persist state after view change
  - So if we crash and recover, we know view# when we crashed
  - Even that unnecessary with more expensive recovery protocol

65

Summary

- Primary-Backup
  - Order updates at the primary
  - Order preserved by view change
- Quorum Consensus
- Split brain?
- What about FLP?

66

66