L20: Replicated state machines with Paxos

Sam Madden

6.033 Spring 2014

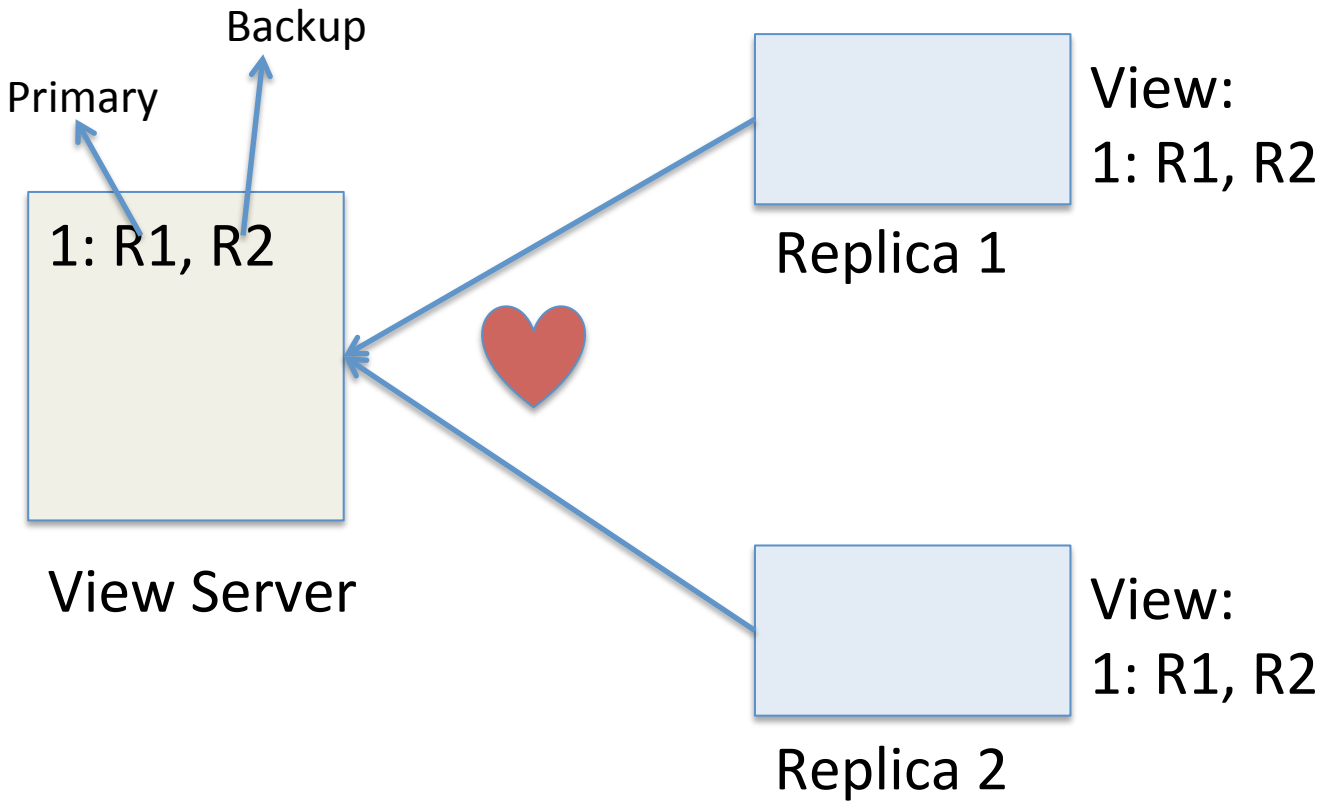
Overall Goal

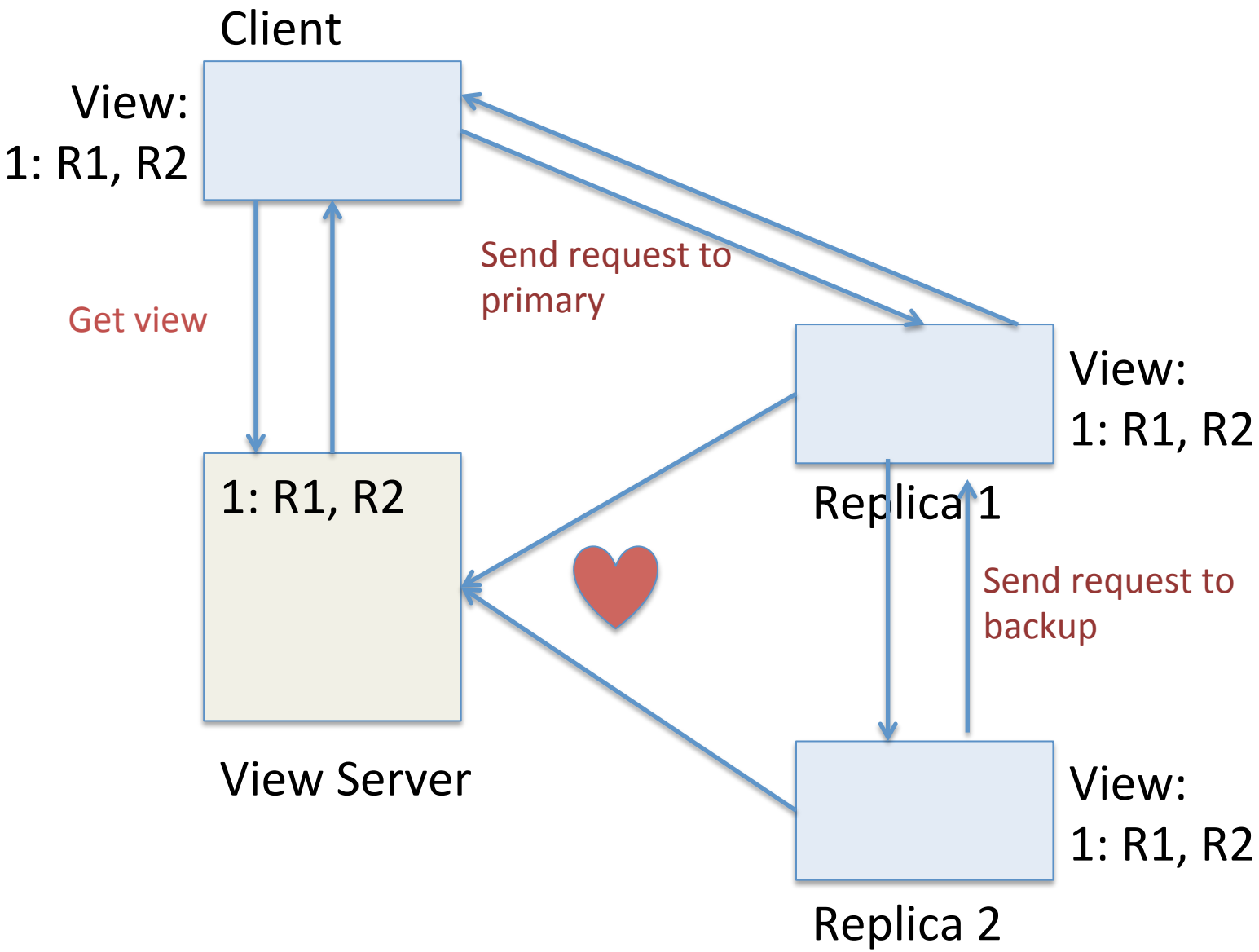
Building a stateful server (e.g., database) that remains *available* in the presence of node failures

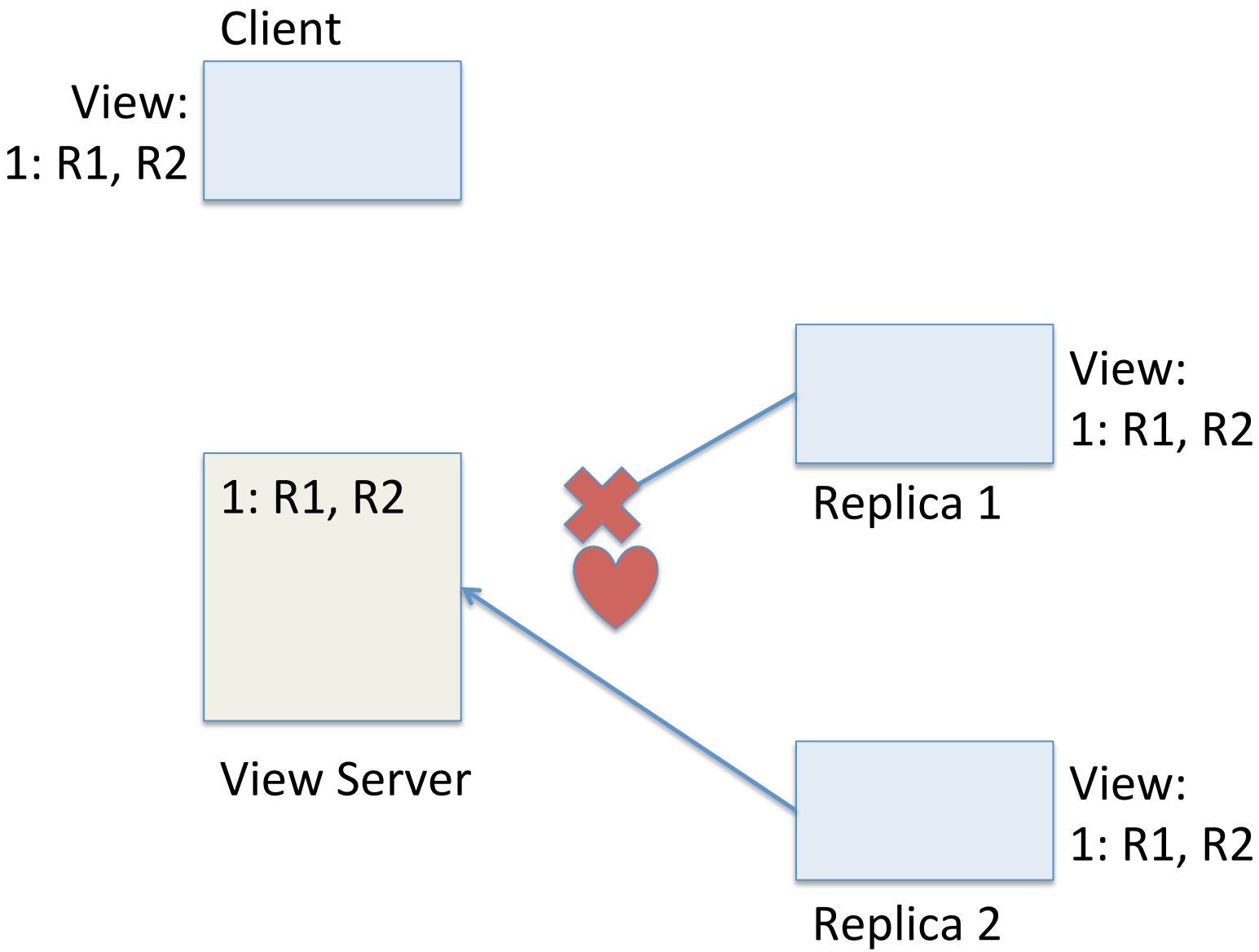
Last time: Replicated State Machine (RSM)

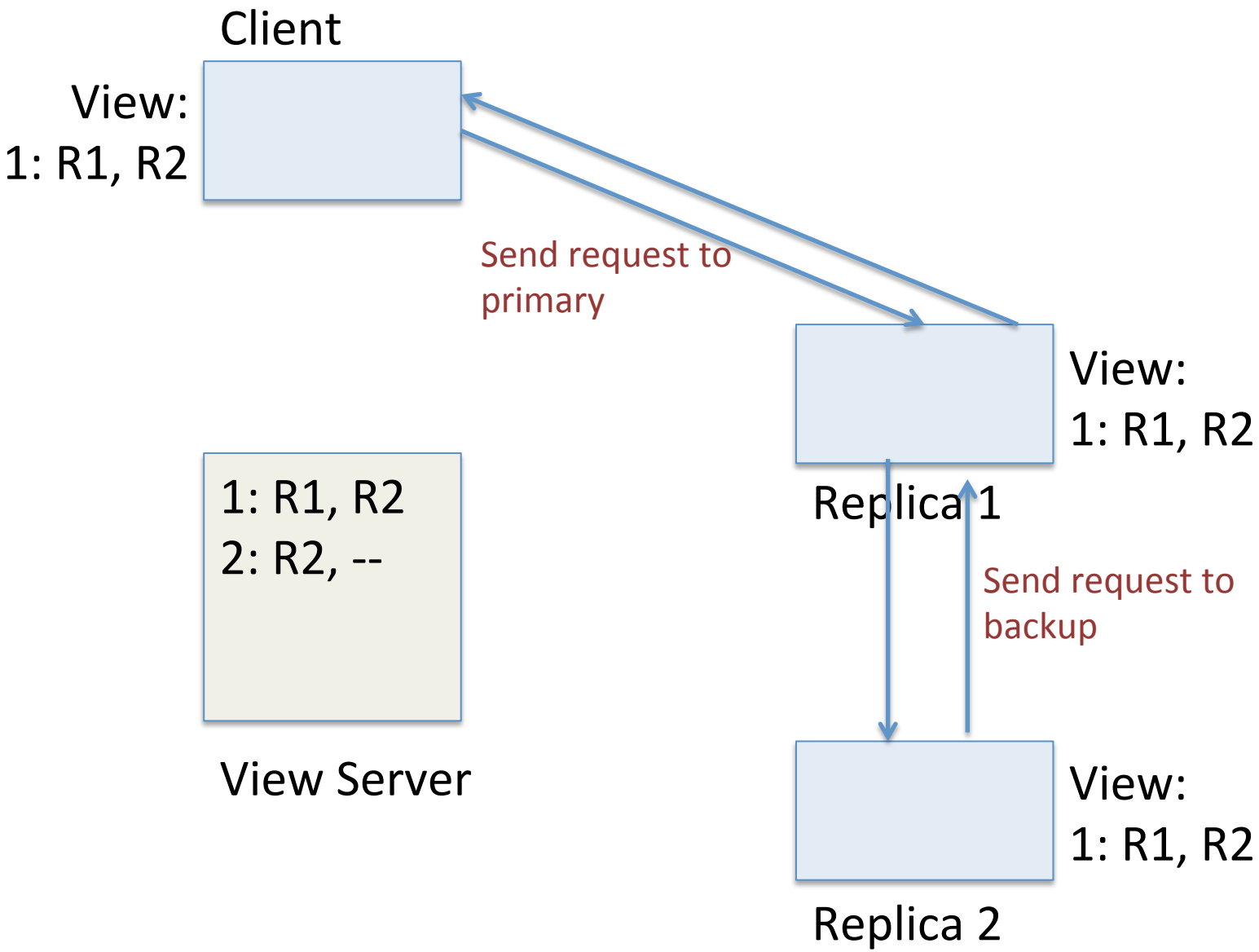
- Key idea: send all replicas same commands in same order
  - This will keep them in sync
- Idea: make one node a primary (coordinator) to order all transactions and send to others
- What if primary fails? Introduce a *view server* that keeps track of current primary / replica

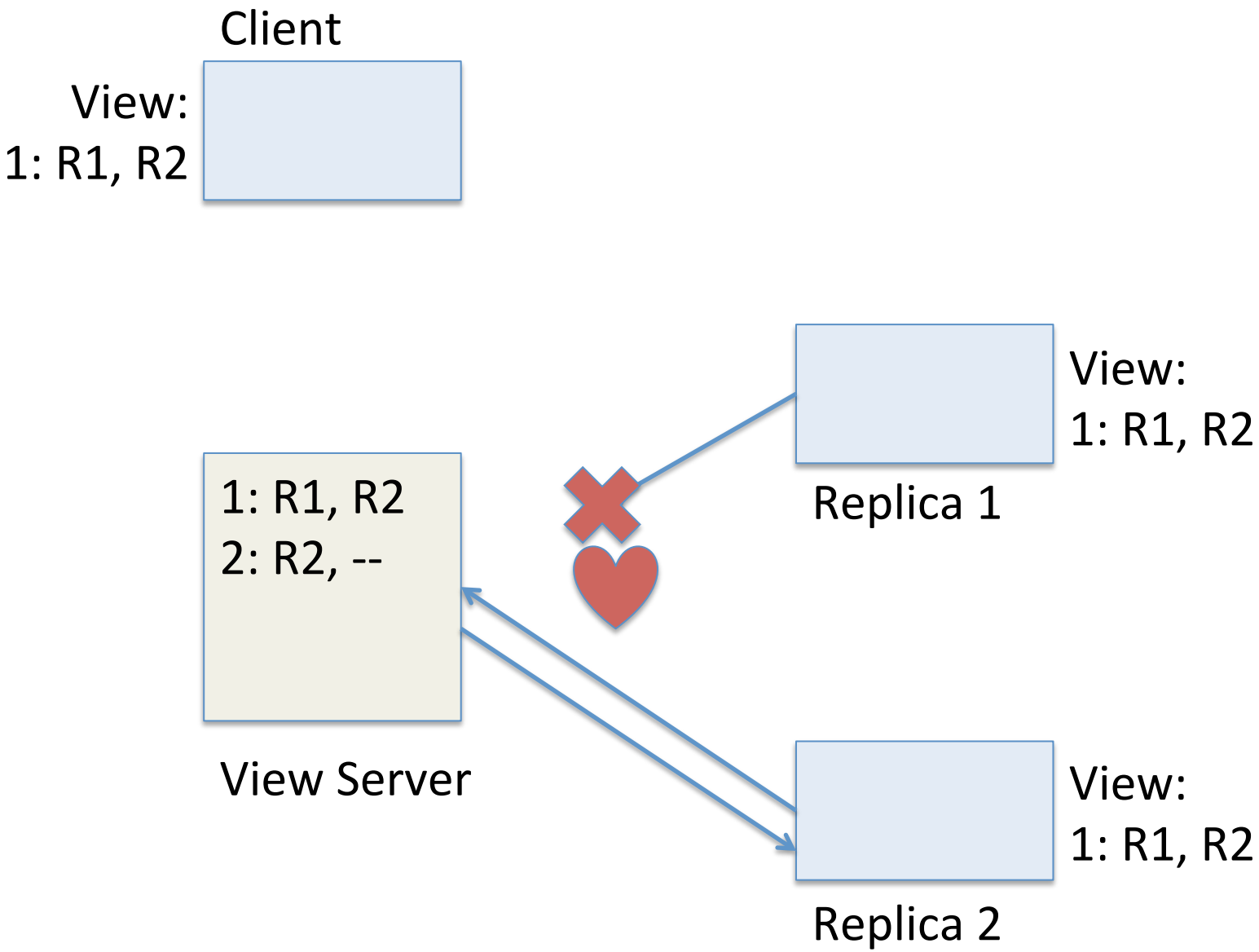
RSM w/ View Server Example

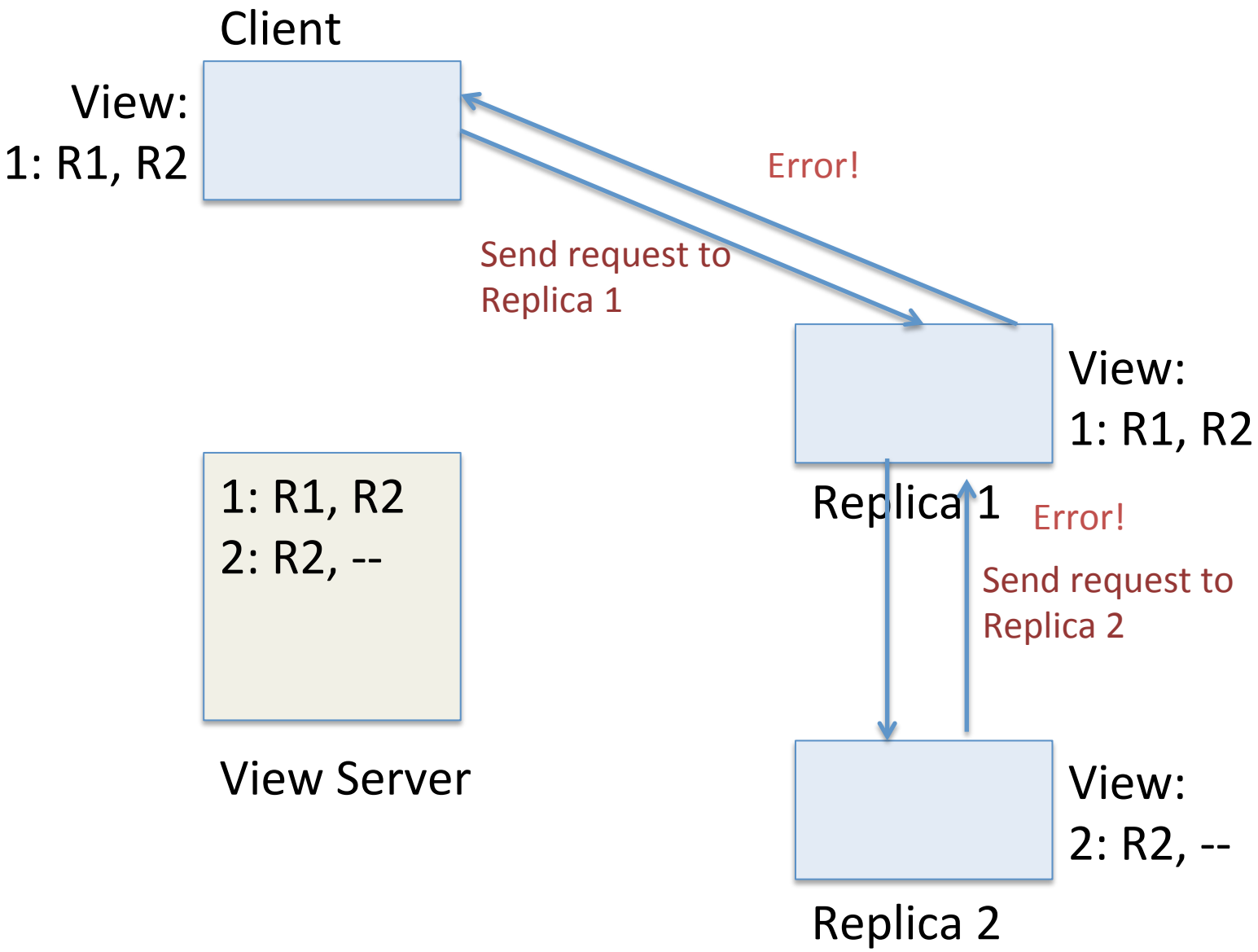



Client
View:
1: R1, R2
Get view
Send request to
primary
View:
1: R1, R2
Replica 1
1: R1, R2
Send request to
backup
View Server
View:
1: R1, R2
Replica 2


Client
View:
1: R1, R2
1: R1, R2
View Server
Replica 1
View:
1: R1, R2
Replica 2
View:
1: R1, R2

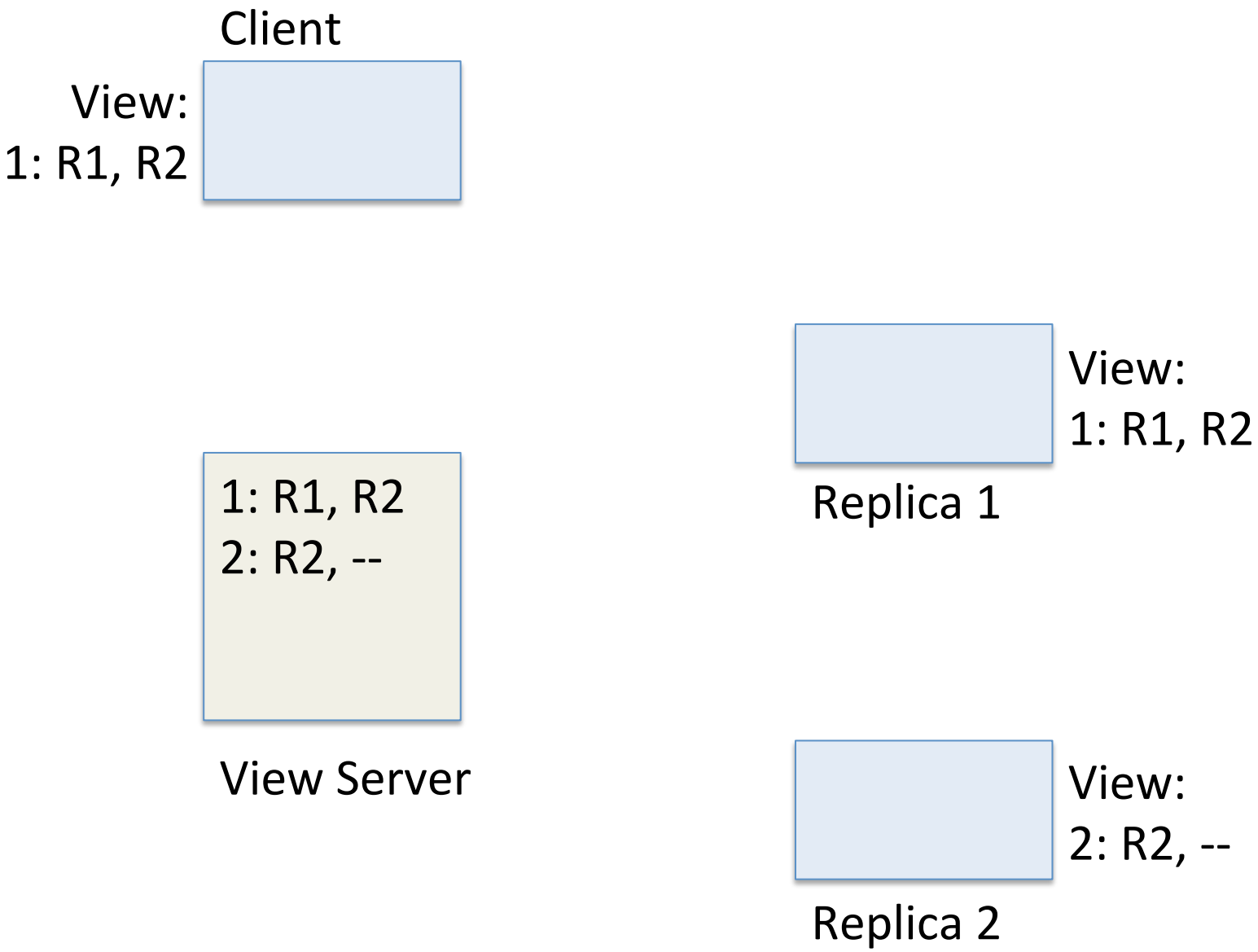

Client
View:
1: R1, R2
Send request to
primary
View:
1: R1, R2
Replica 1
1: R1, R2
2: R2, --
Send request to
backup
View Server
View:
1: R1, R2
Replica 2

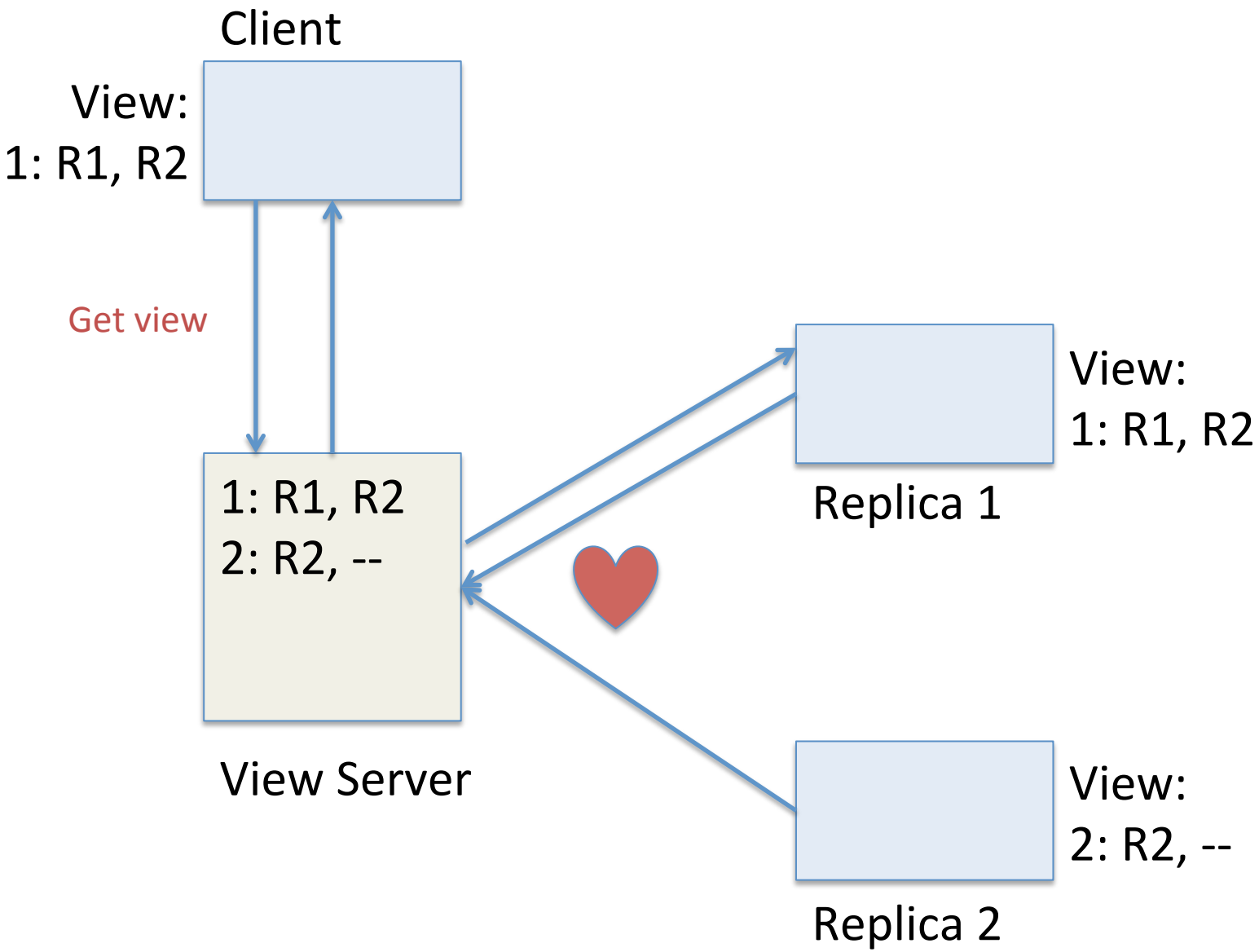

Client
View:
1: R1, R2
View:
1: R1, R2
Replica 1
1: R1, R2
2: R2, --
View Server
View:
1: R1, R2
Replica 2

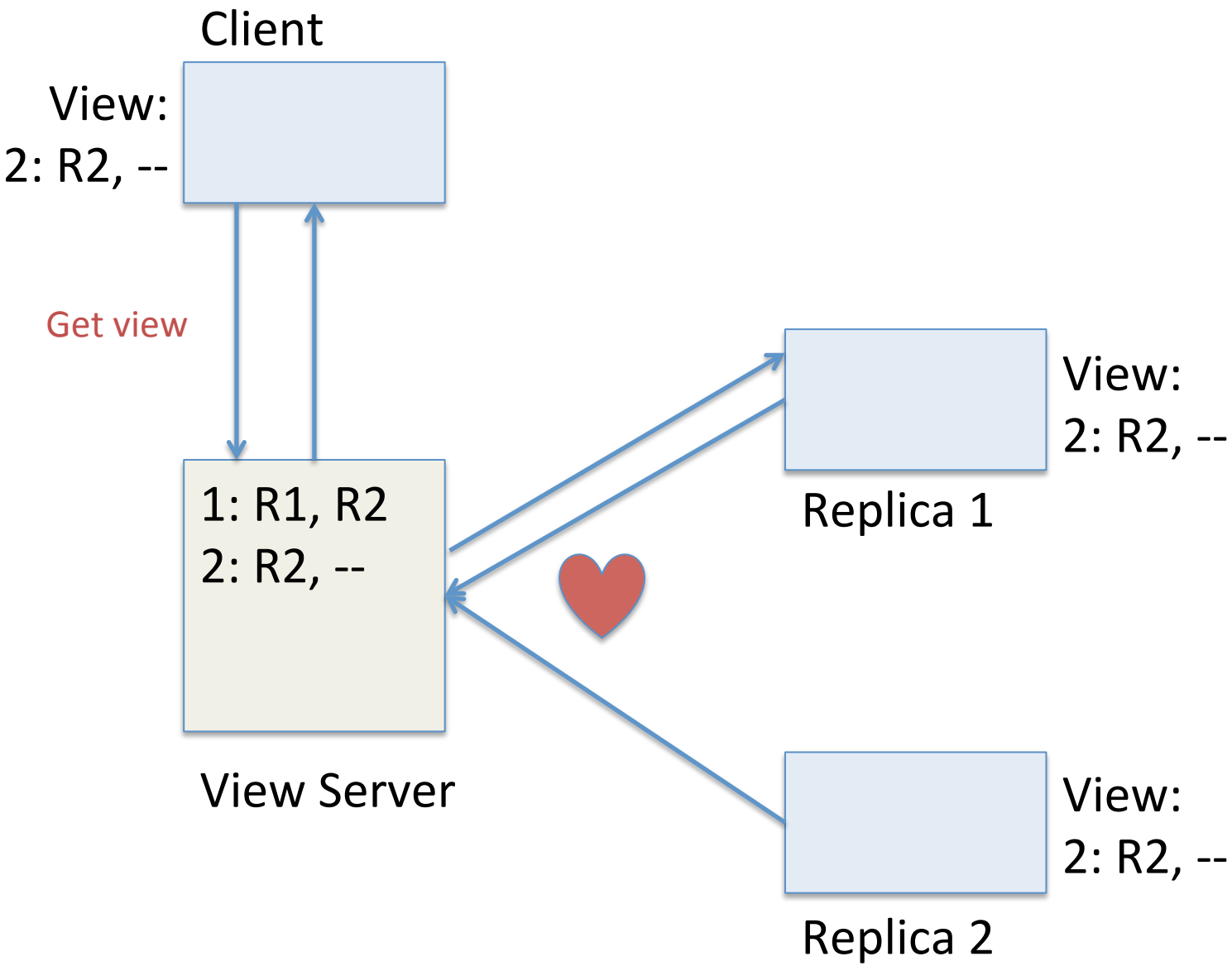


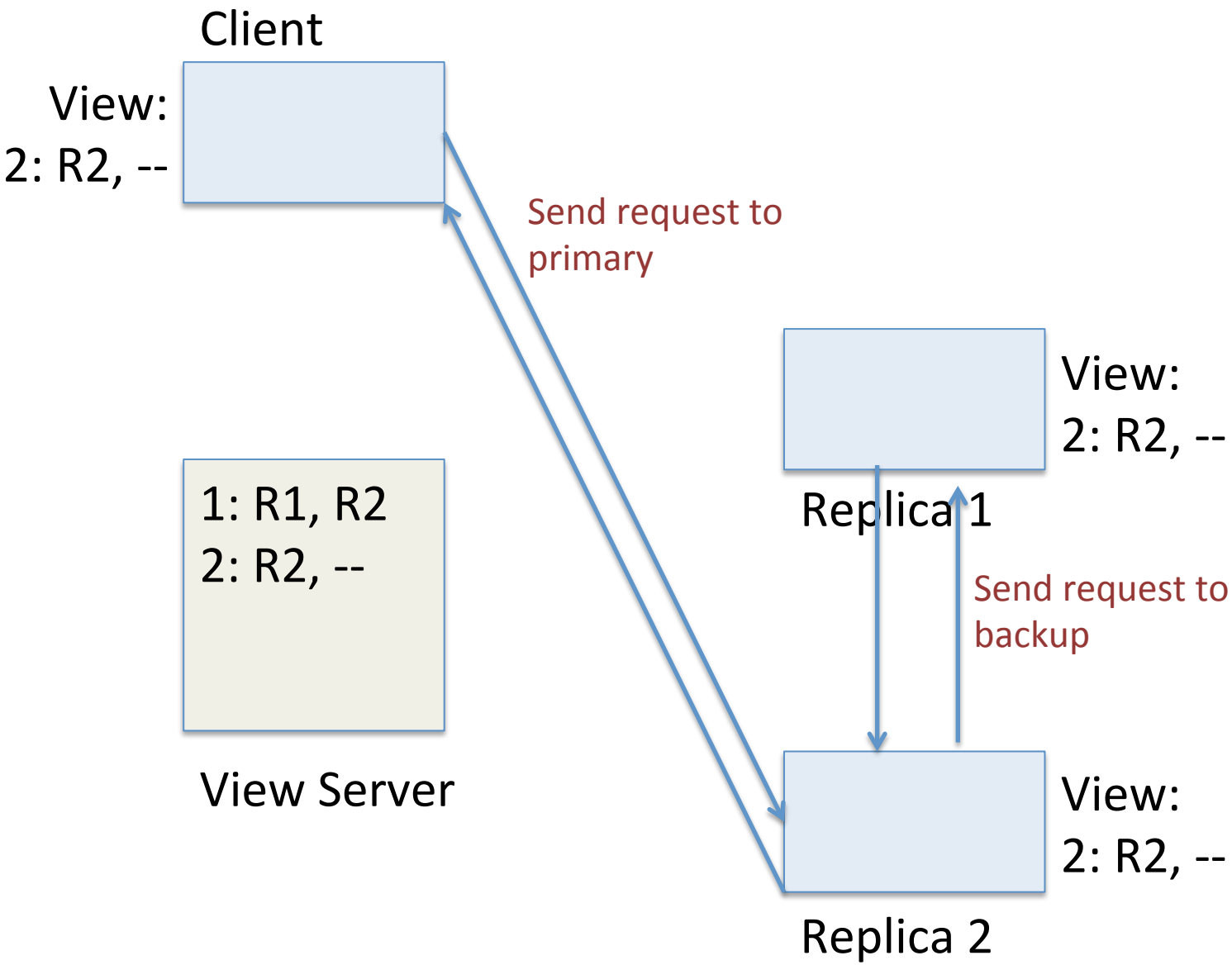
Moment R2 learns about new view it becomes active

Replica 2 refuses to process request


Client
View:
1: R1, R2
1: R1, R2
2: R2, --
View Server
View:
1: R1, R2
Replica 1
View:
2: R2, --
Replica 2


Client
View:
1: R1, R2
Get view
1: R1, R2
2: R2, --
View Server
View:
1: R1, R2
Replica 1
View:
2: R2, --
Replica 2


Client
View:
2: R2, --
Get view
1: R1, R2
2: R2, --
View Server
View:
2: R2, --
Replica 1
View:
2: R2, --
Replica 2


Client
View:
2: R2, --
Send request to
primary
View:
2: R2, --
Replica 1
1: R1, R2
2: R2, --
Send request to
backup
View Server
View:
2: R2, --
Replica 2

Problem: How to make view server fault tolerant

Paxos Goal

N >= 3 nodes, trying to *agree* about some value (e.g., the next view server for the RSM protocol)

Paxos properties

- All nodes agree on a value, despite node failures, network failures, delays
  - E.g., X is the next operation to execute
  - E.g., Y is the next primary
- Fault tolerant: succeeds if fewer than N/2 nodes fail
  - Liveness not guaranteed
- Assumption: nodes are fail-stop

Setup

Servers each run 3 processes:

Proposer – proposes new values

Acceptor – accepts (or rejects) proposals

Learner – client waiting for new value

Paxos rule

- If a majority of nodes in an earlier proposal number accepted a value, later proposals must accept the same value

- State maintained by acceptor:
  - Np: largest proposal seen in prepare
  - Na: largest proposal seen in accept
  - Va: value accepted for proposal Na

- State must be persistent across reboot

Propose(V):
choose unique N, > Np
send **Prepare**(N) to acceptors
if **Prepare_OK**(Na, Va) from majority:
V' = Va with highest Na, or V if none
send **Accept**(N, V') to acceptors
if **Accept_OK**(N) from majority:
send **Decided**(V') to learners

Paxos

Proposer

Prepare(N):
if N > Np:
log Np = N
reply **Prepare_OK**(Na, Va)

Accept(N, V):
if N ≥ Np:
log Na = N, log Va = V
reply **Accept_OK**(Na, Va)

- **Np**: largest proposal seen in prepare
- **Na**: largest proposal seen in accept
- **Va**: value accepted for proposal Na

Acceptor

Propose(V):
choose unique N, > Np
send **Prepare**(N) to acceptors
if **Prepare_OK**(Na, Va) from majority:
V' = Va with highest Na, or V if none
send **Accept**(N, V') to acceptors
if **Accept_OK**(N) from majority:
send **Decided**(V') to learners

Paxos

Proposer

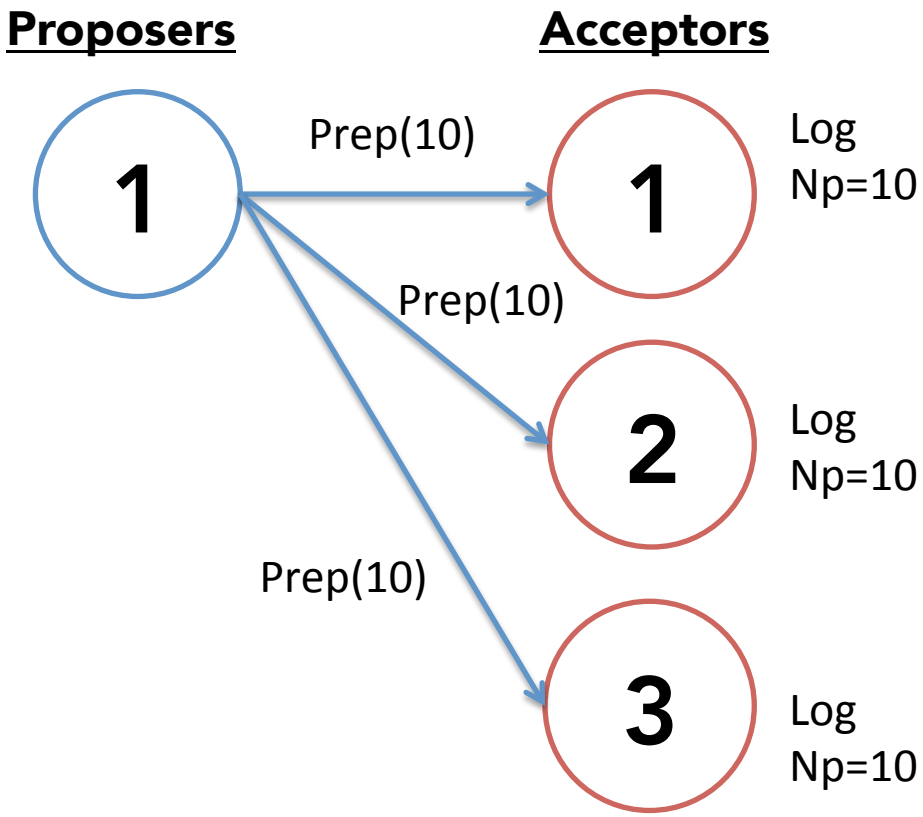
Prepare(N):
if N > Np:
log Np = N
reply **Prepare_OK**(Na, Va)

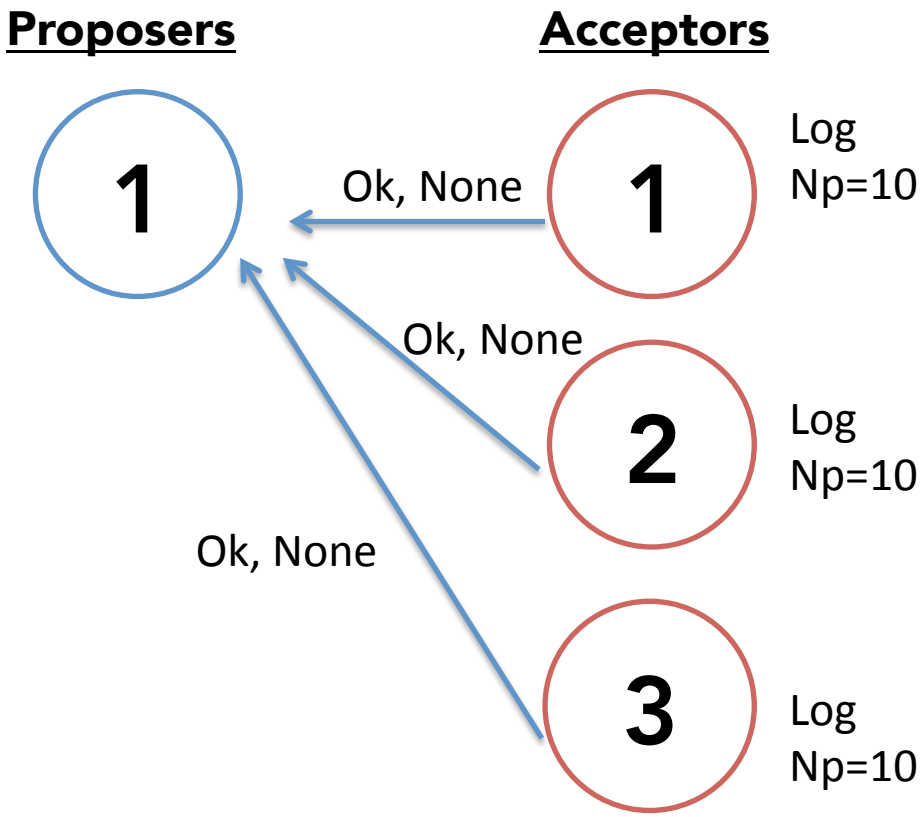
Accept(N, V):
if N ≥ Np:
log Na = N, log Va = V
reply **Accept_OK**(Na, Va)

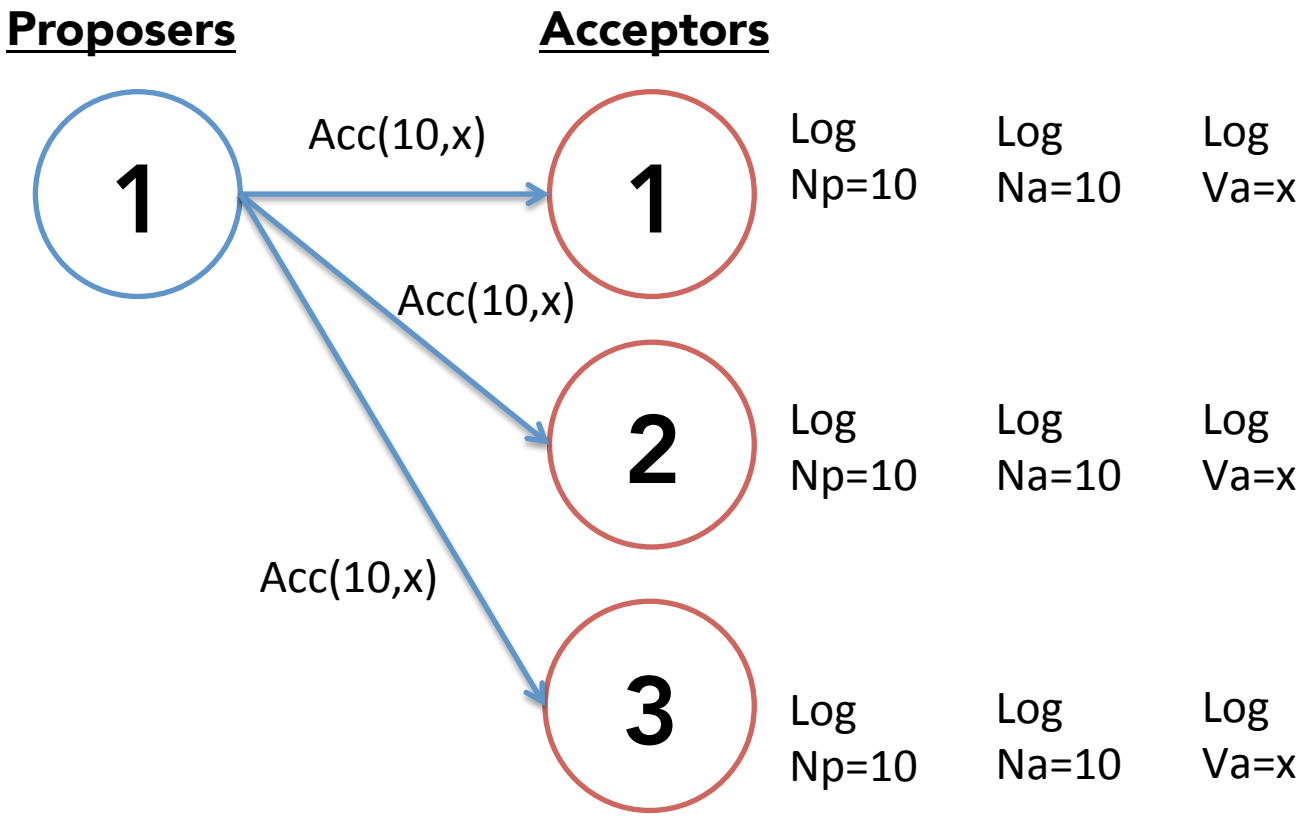
- **Np**: largest proposal seen in prepare
- **Na**: largest proposal seen in accept
- **Va**: value accepted for proposal Na

Acceptor

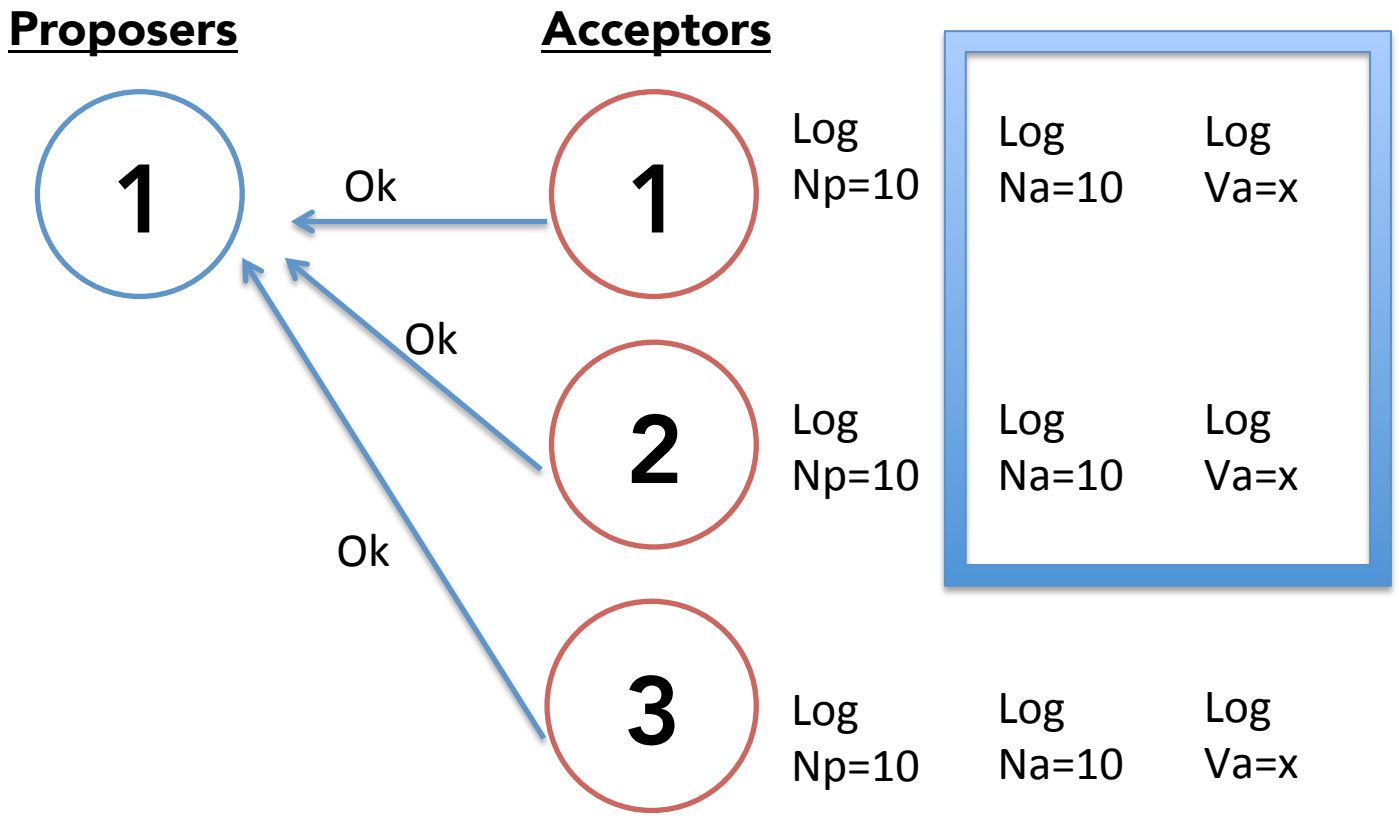
Example 1


Proposers
Acceptors
1
Prep(10)
1
Log
Np=10
Prep(10)
2
Log
Np=10
Prep(10)
3
Log
Np=10

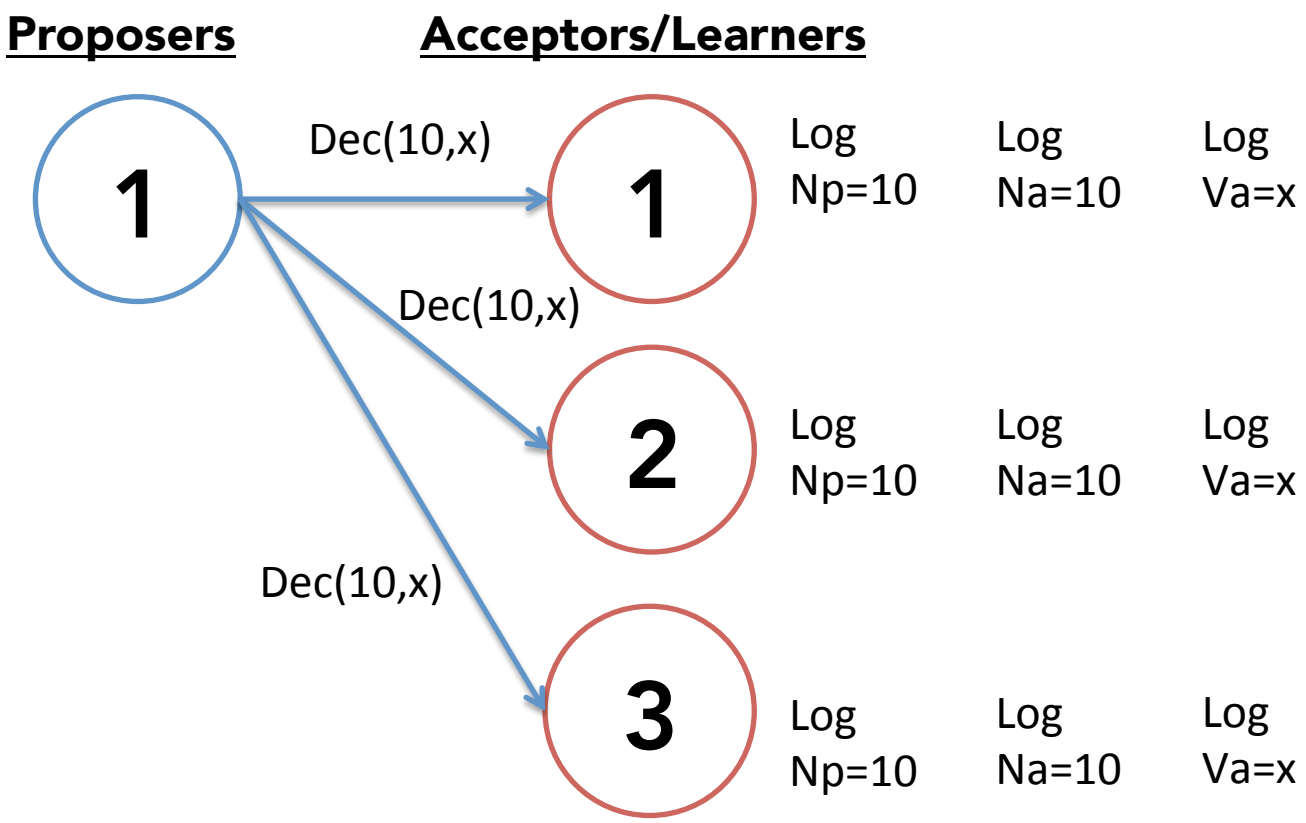

Proposers
Acceptors
1
1
Ok, None
Log
Np=10
Ok, None
2
Log
Np=10
Ok, None
3
Log
Np=10


Proposers
Acceptors
1
Acc(10,x)
1
Log
Np=10
Log
Na=10
Log
Va=x
Acc(10,x)
2
Log
Np=10
Log
Na=10
Log
Va=x
Acc(10,x)
3
Log
Np=10
Log
Na=10
Log
Va=x

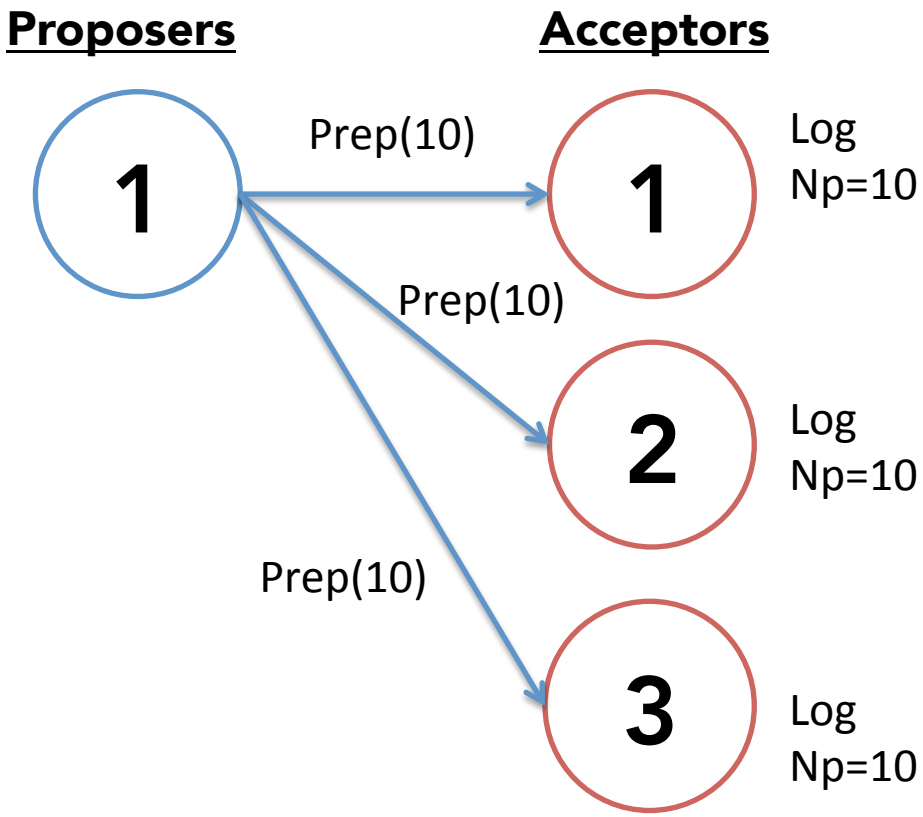
Commit point when majority of acceptors log value

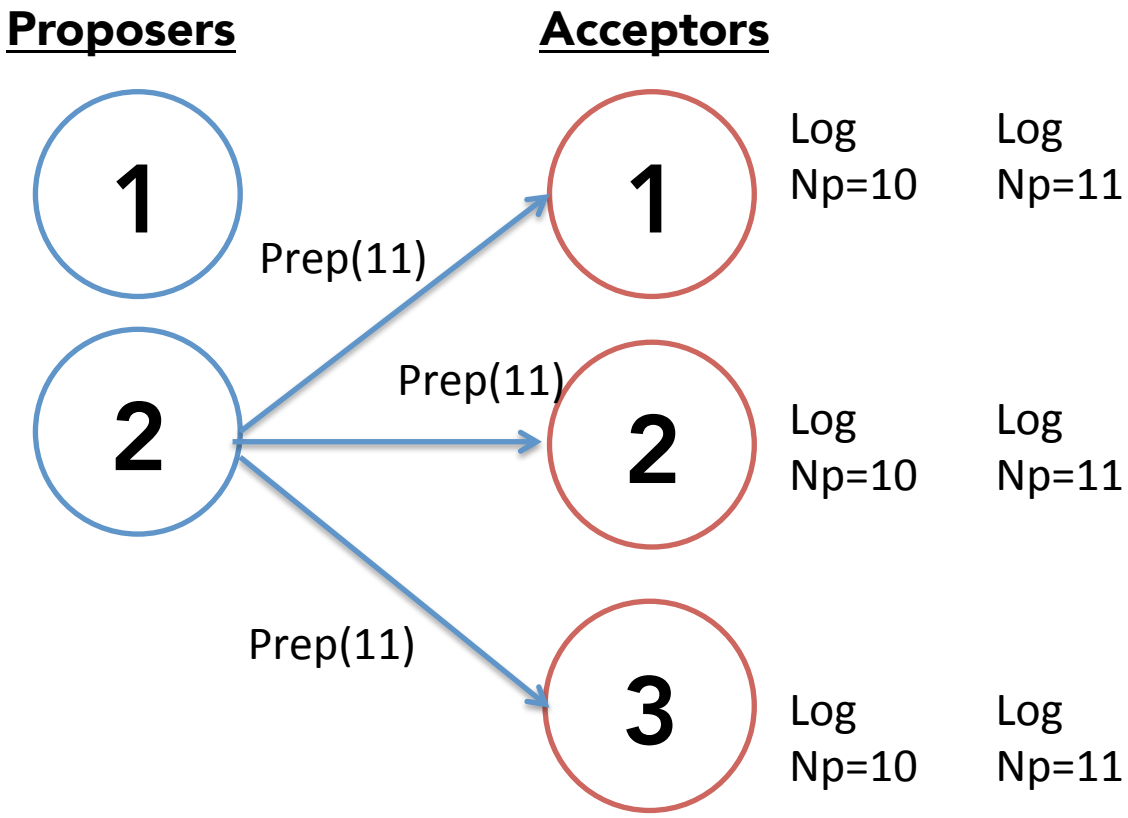


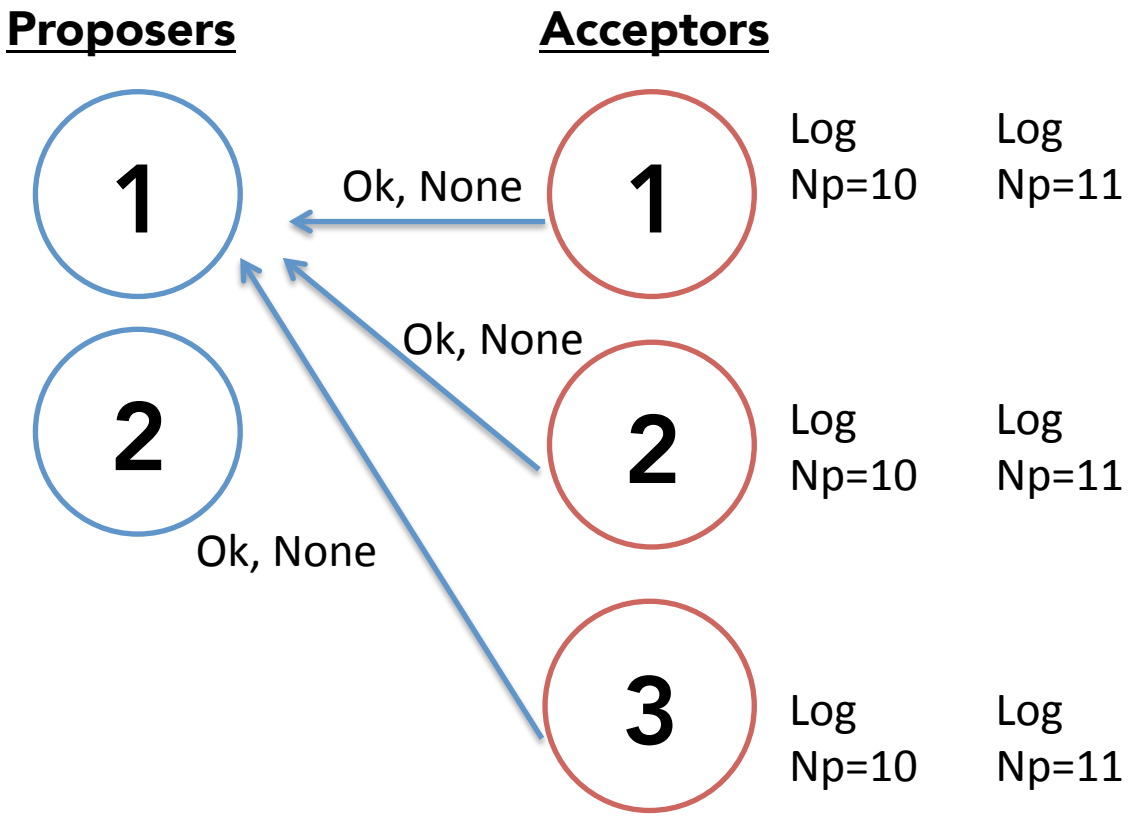
A1—3 can share with learners

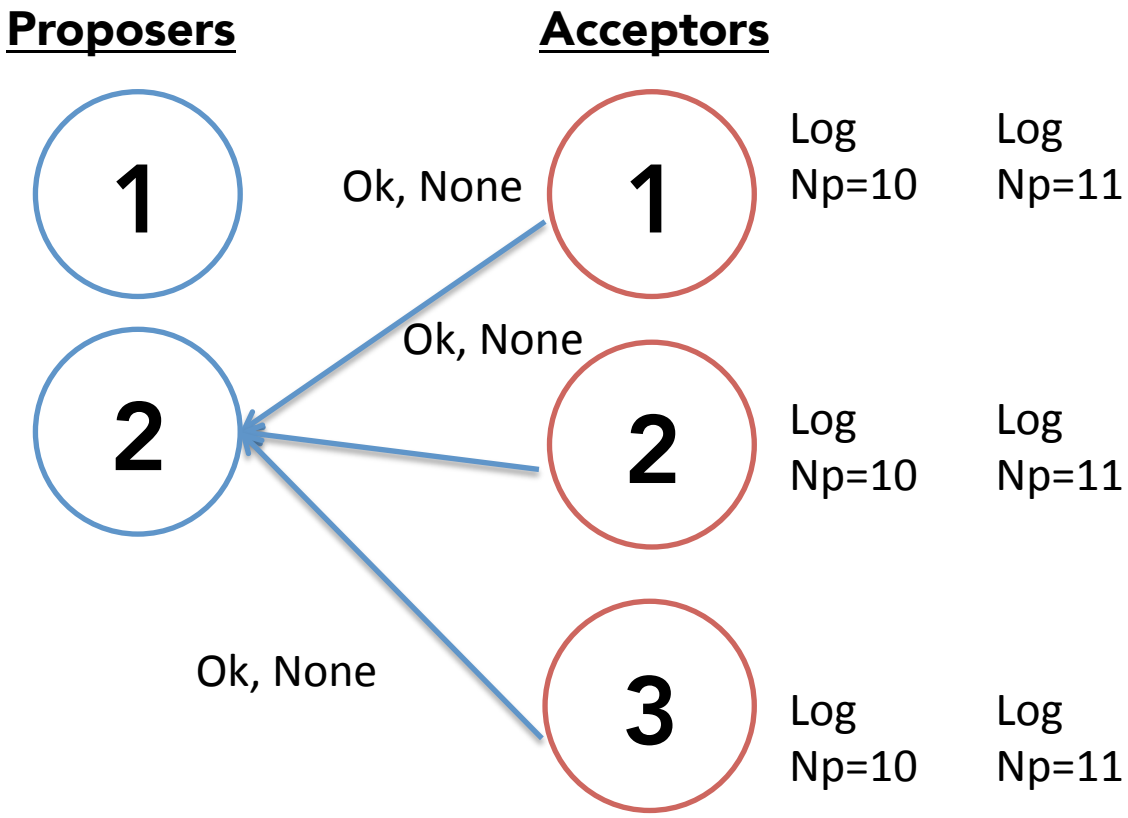


Example 2

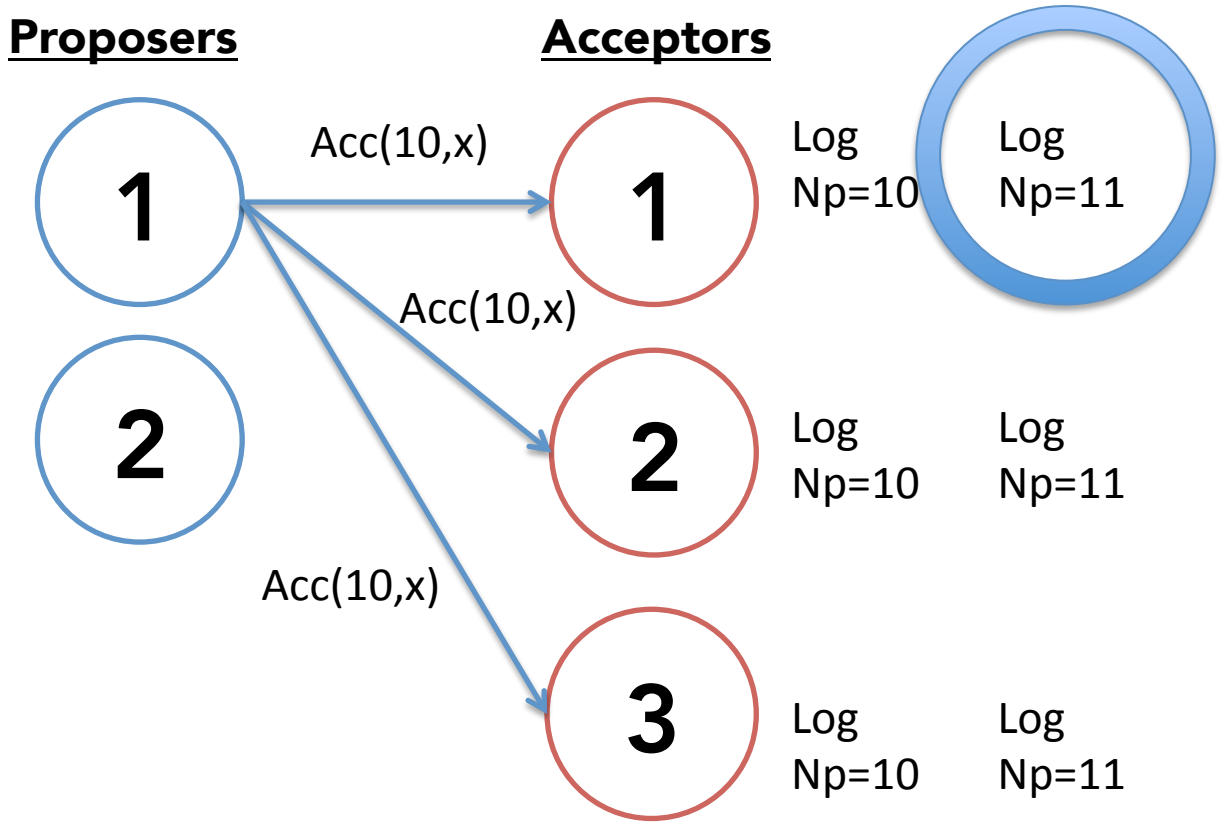

Proposers
Acceptors
1
Prep(10)
1
Log
Np=10
Prep(10)
2
Log
Np=10
Prep(10)
3
Log
Np=10


Proposers
Acceptors
1
2
1
2
3
Prep(11)
Prep(11)
Prep(11)
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11


Proposers
Acceptors
1
2
1
2
3
Ok, None
Ok, None
Ok, None
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11


Proposers
Acceptors
1
2
1
2
3
Ok, None
Ok, None
Ok, None
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11
Log
Np=10
Log
Np=11

P1 attempts commit first, fails

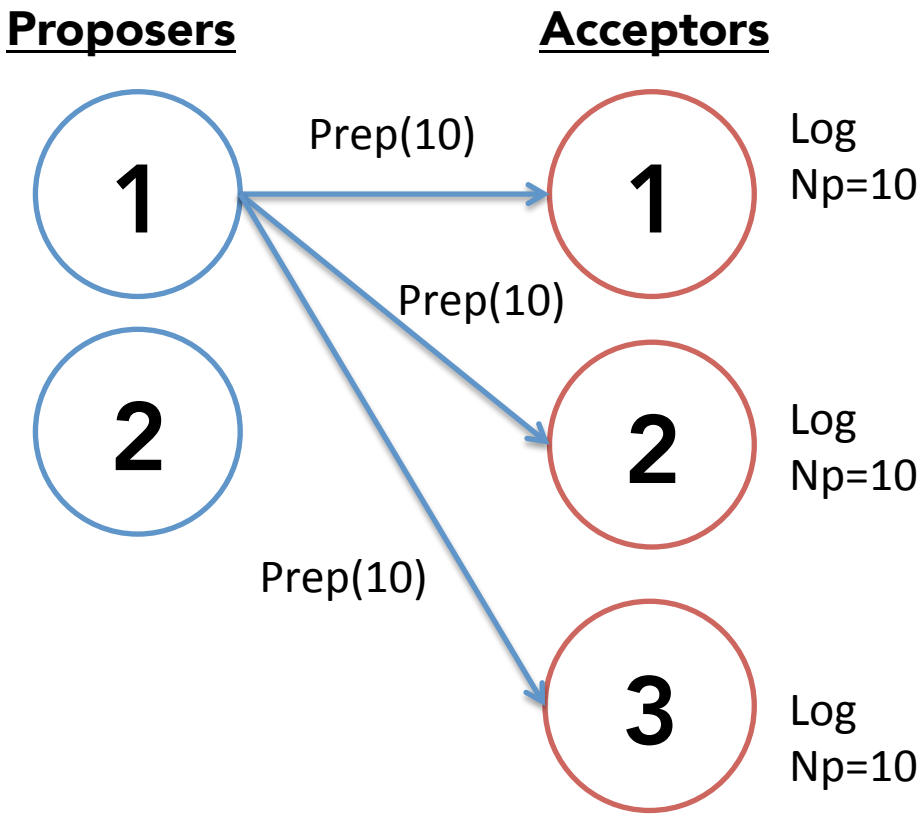


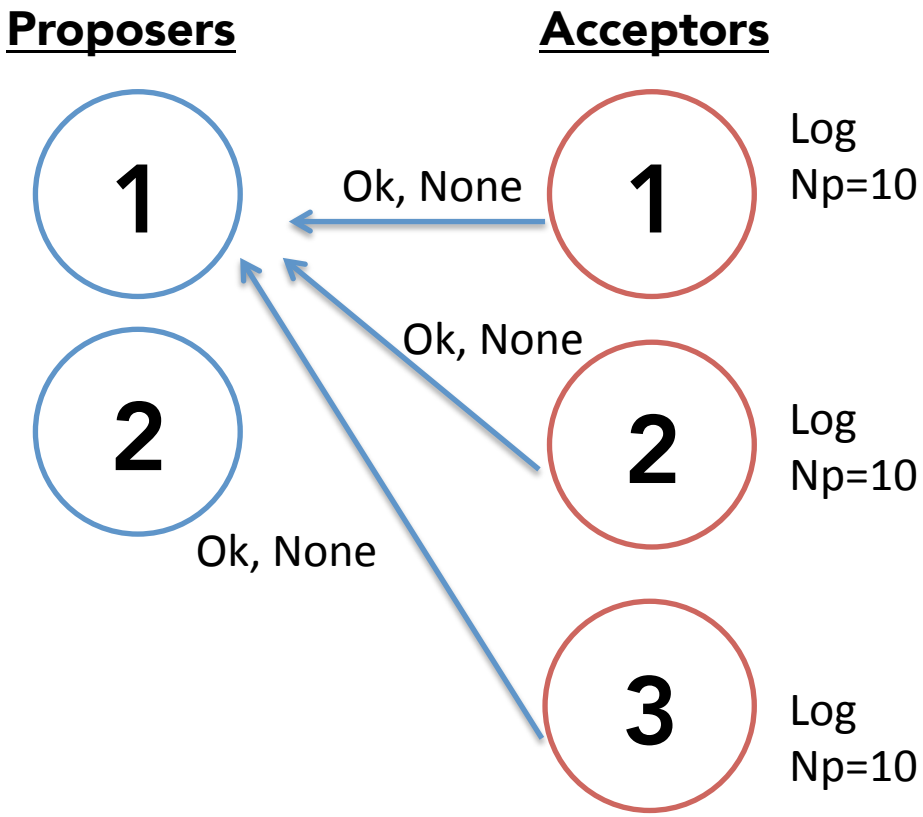
Rejected because Np > 10

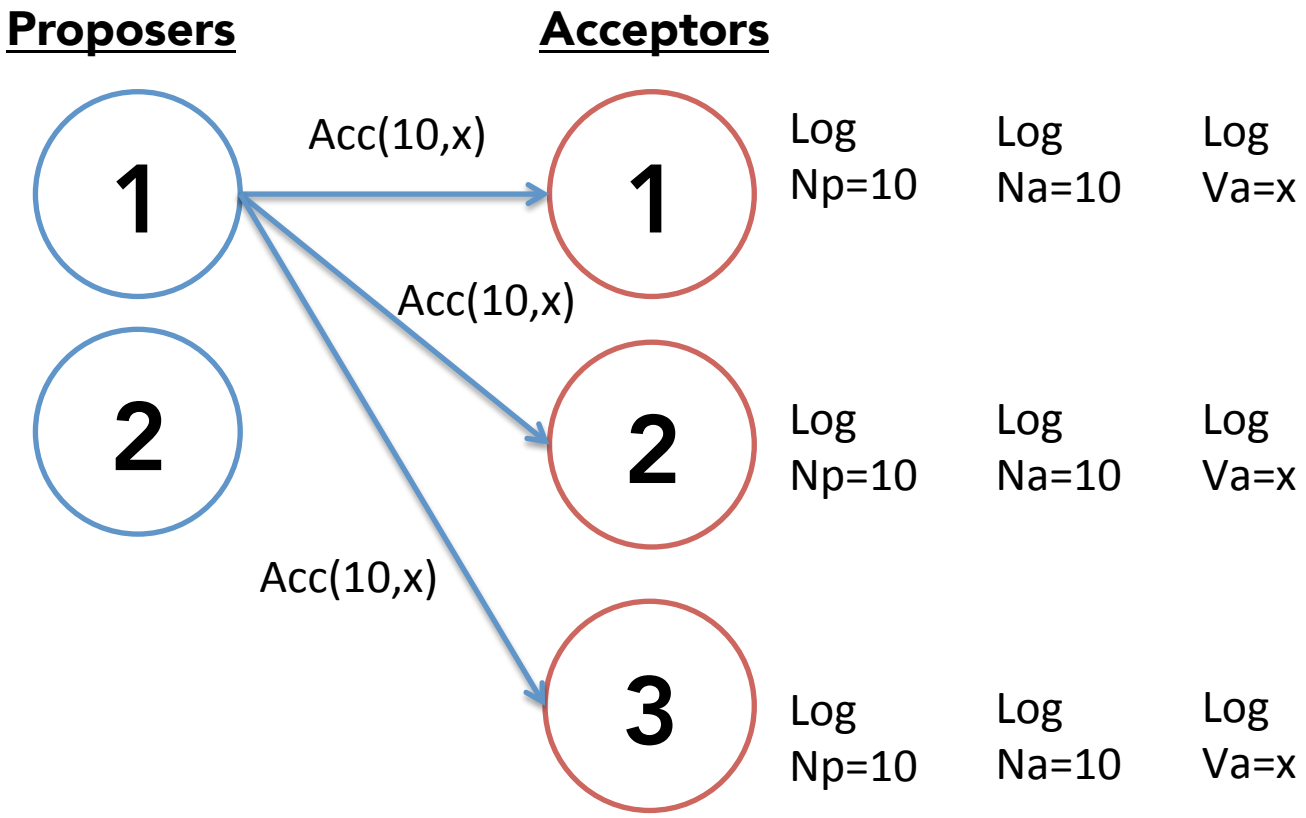
(This is why we must record Np)

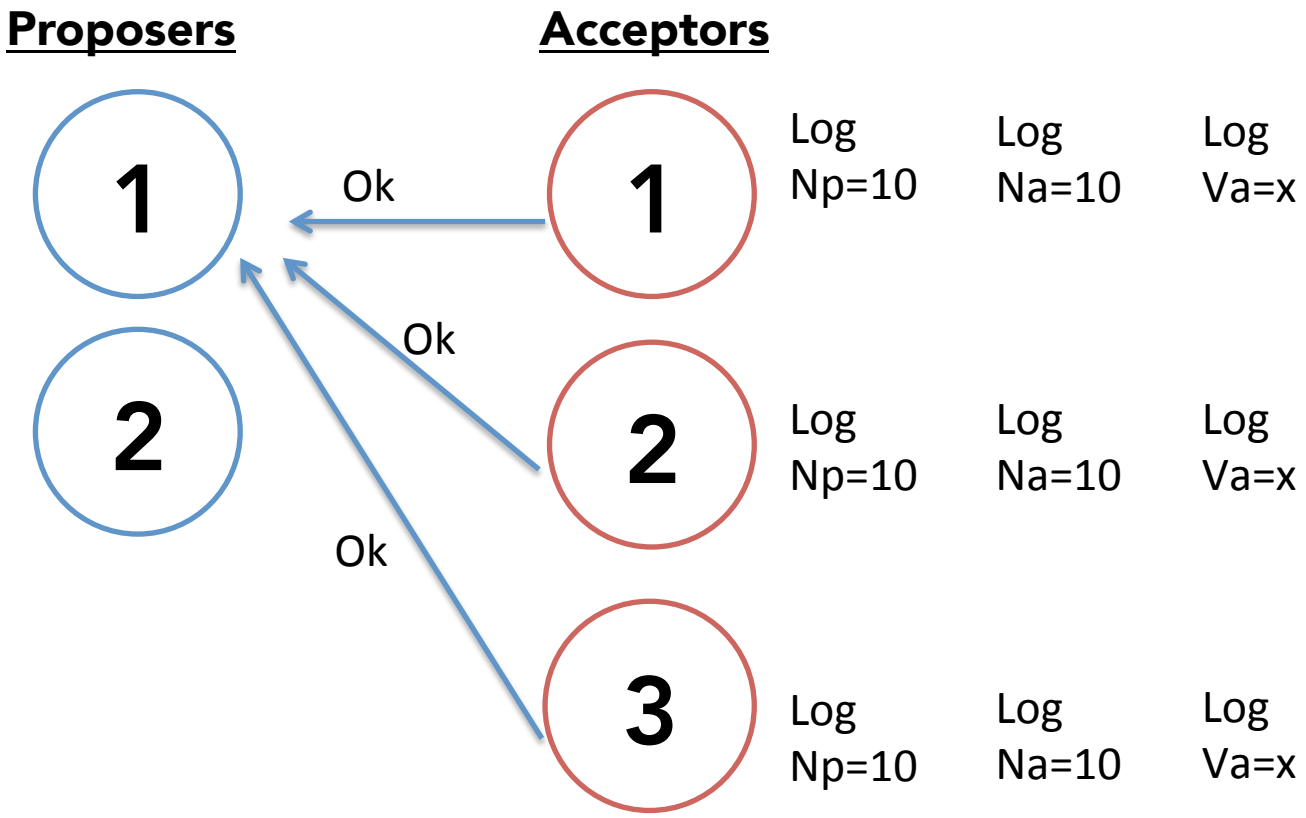
Decided Message Omitted for Brevity

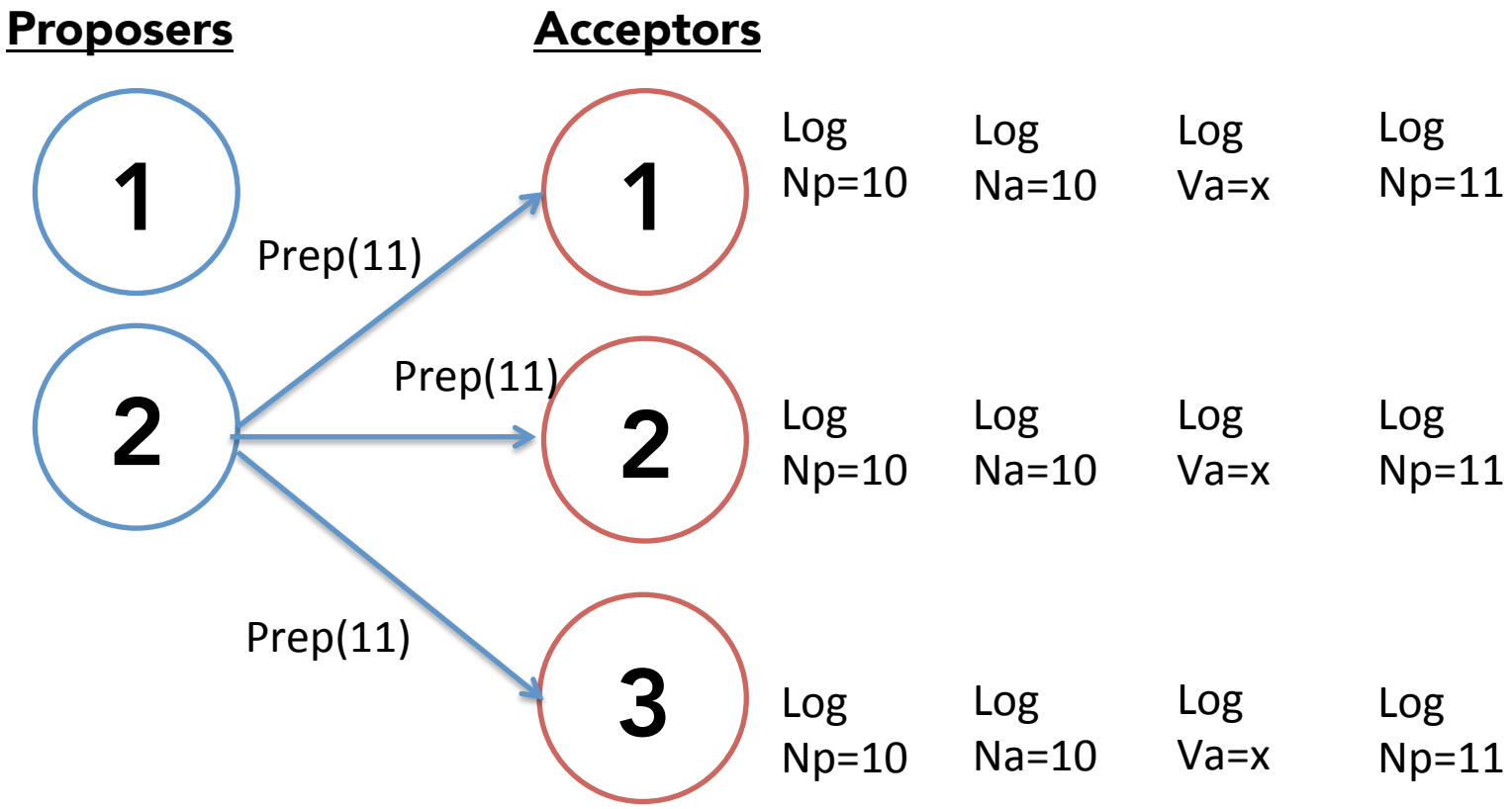
Example 3


Proposers
Acceptors
1
2
Prep(10)
Prep(10)
Prep(10)
1
2
3
Log
Np=10
Log
Np=10
Log
Np=10

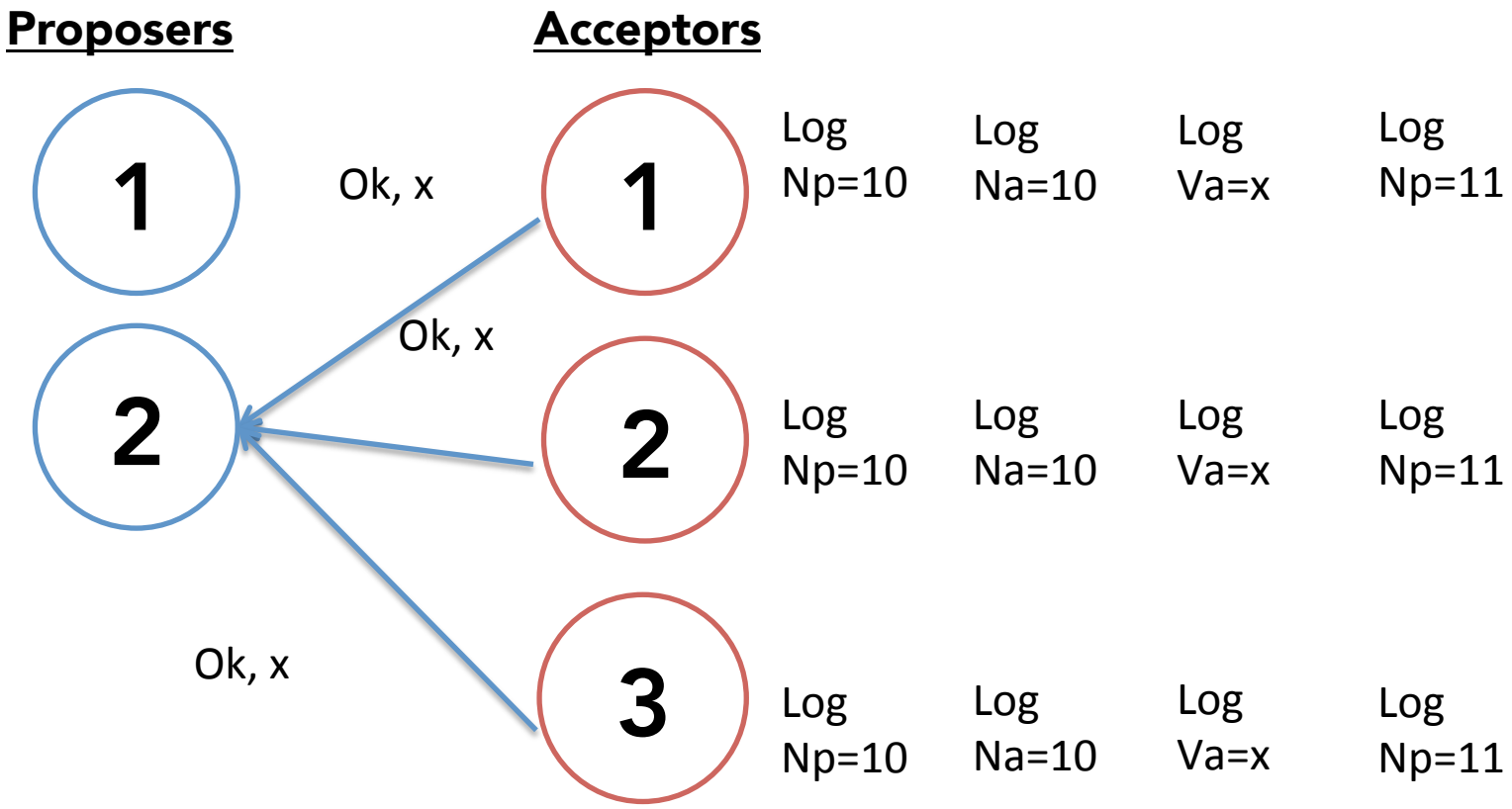

Proposers
Acceptors
1
2
1
2
3
Ok, None
Ok, None
Ok, None
Log
Np=10
Log
Np=10
Log
Np=10


Proposers
Acceptors
1
2
1
2
3
Acc(10,x)
Acc(10,x)
Acc(10,x)
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x


Proposers
Acceptors
1
2
1
2
3
Ok
Ok
Ok
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x

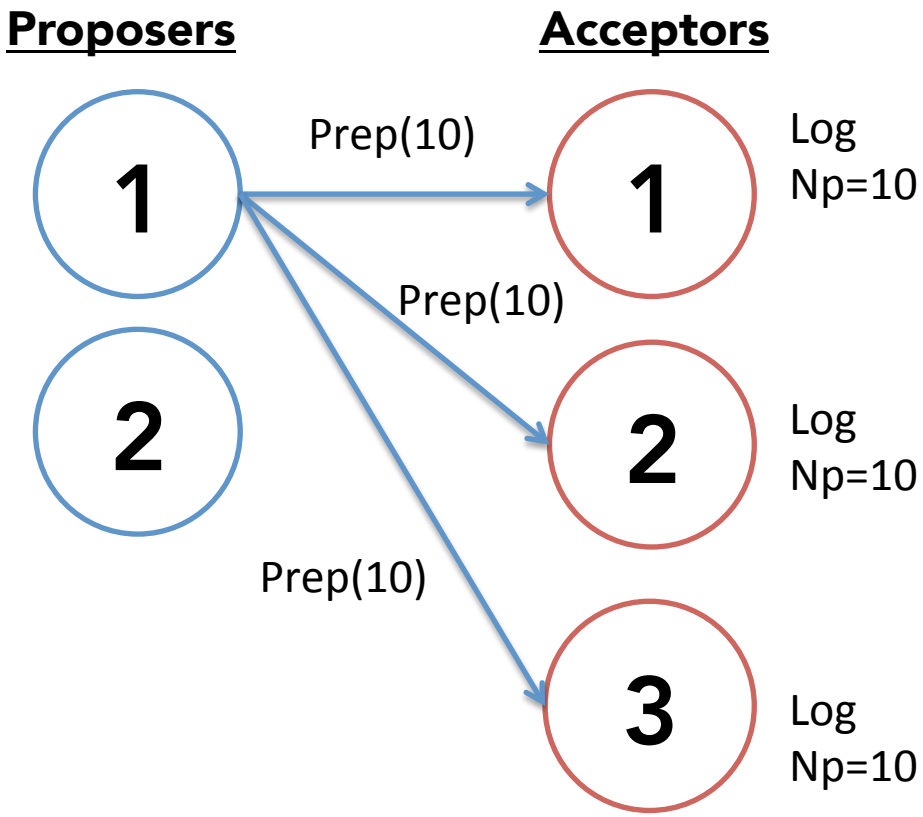

Proposers
Acceptors
1
2
1
2
3
Prep(11)
Prep(11)
Prep(11)
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=11
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=11
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=11

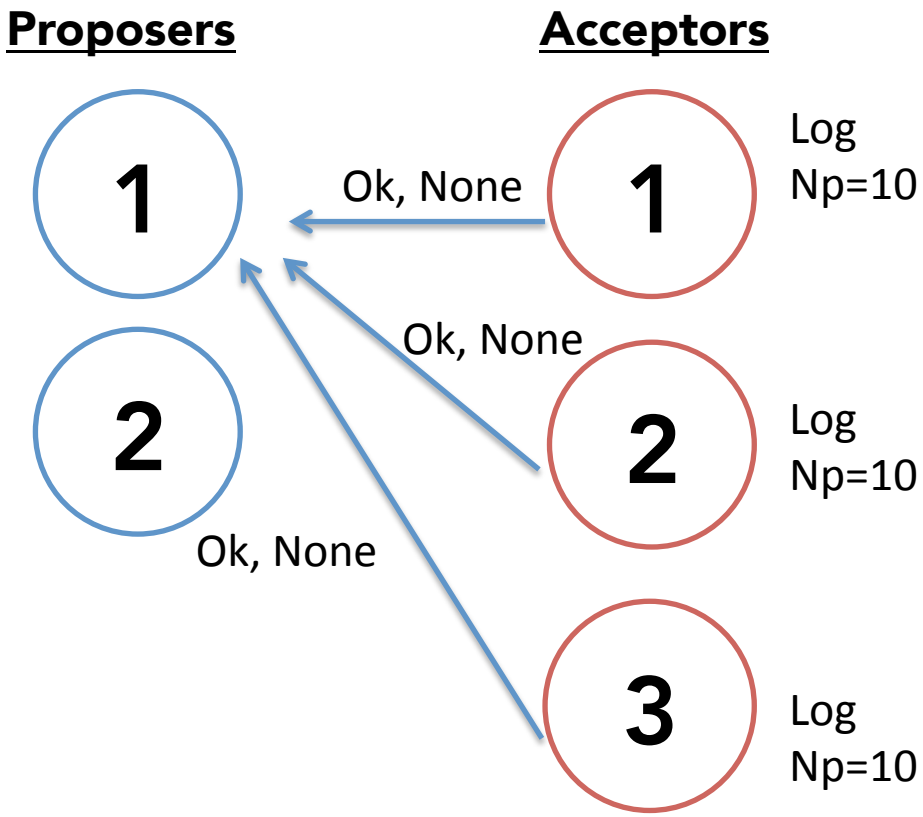
New proposer learns previously committed value

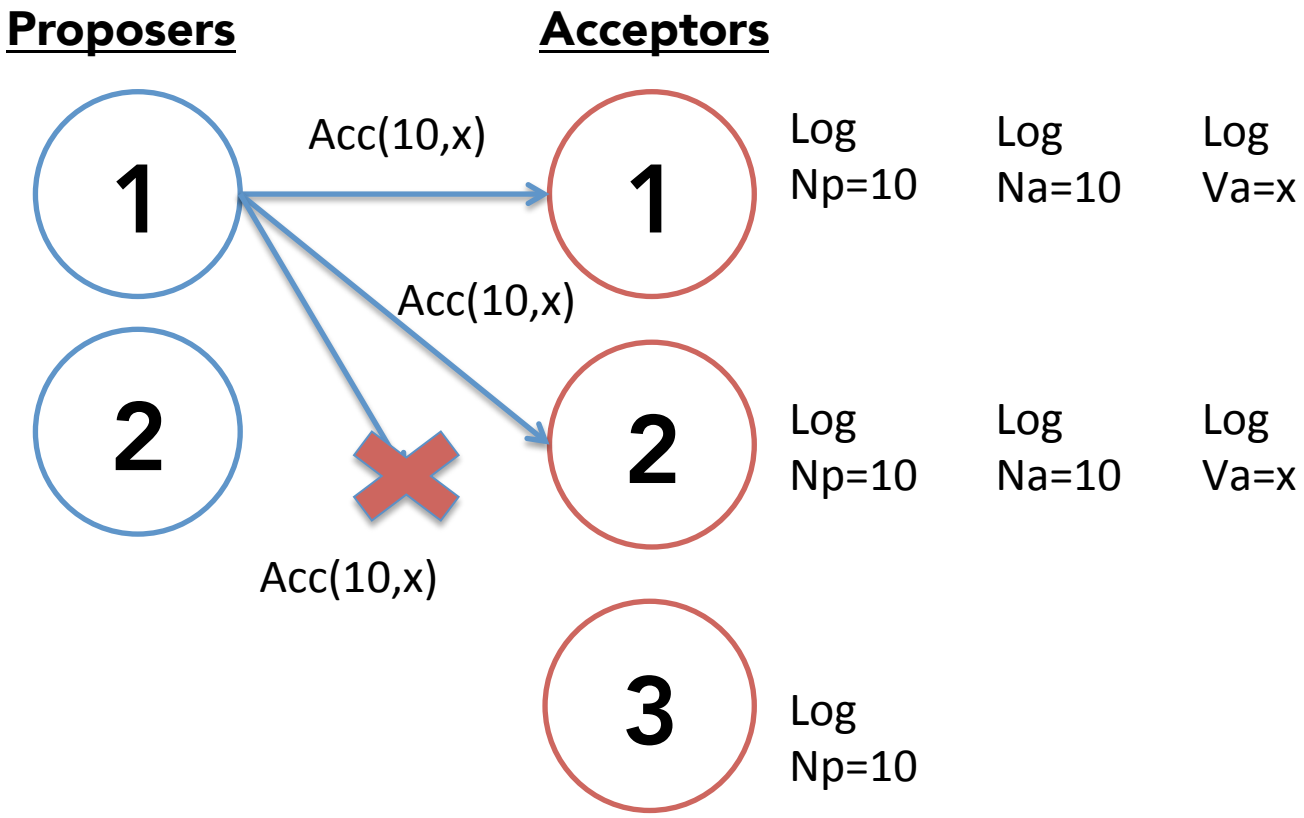


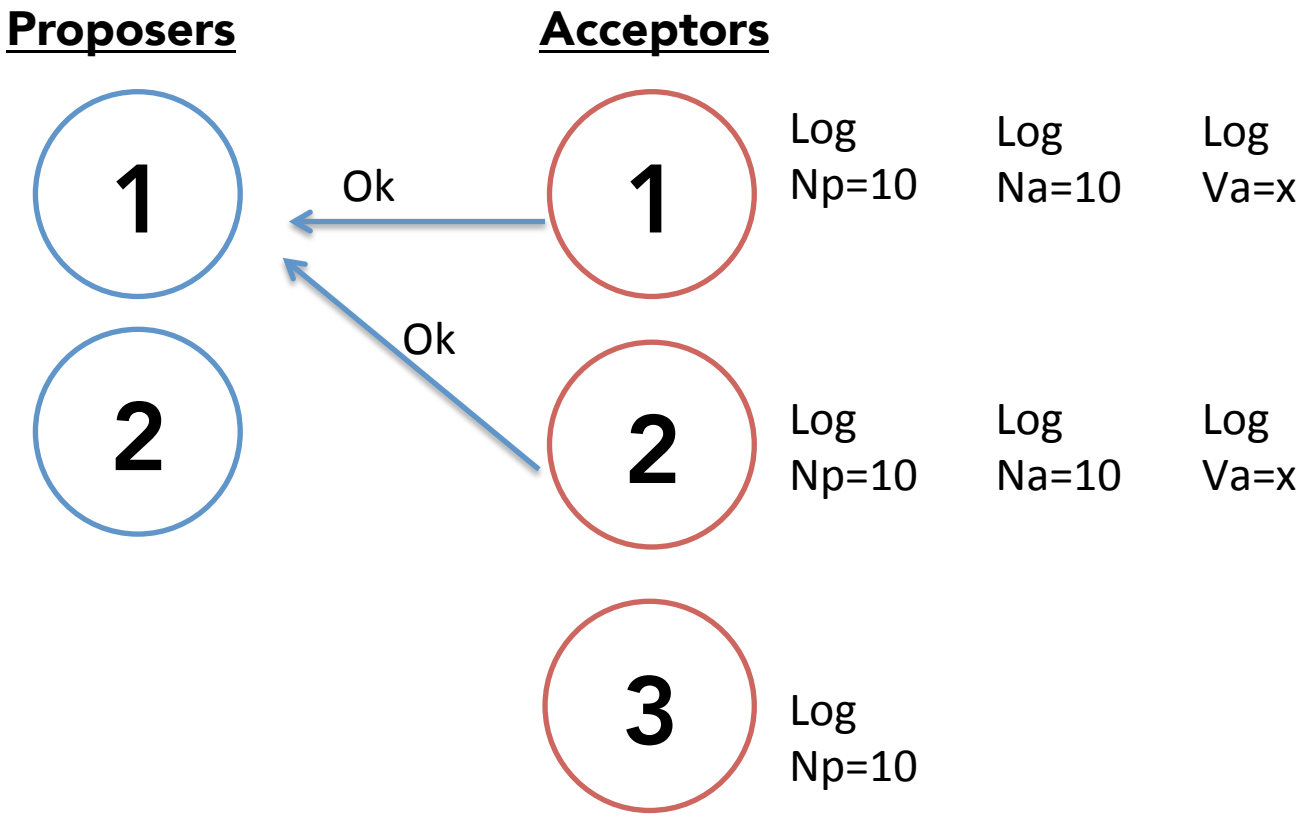
Decided Message Omitted for Brevity

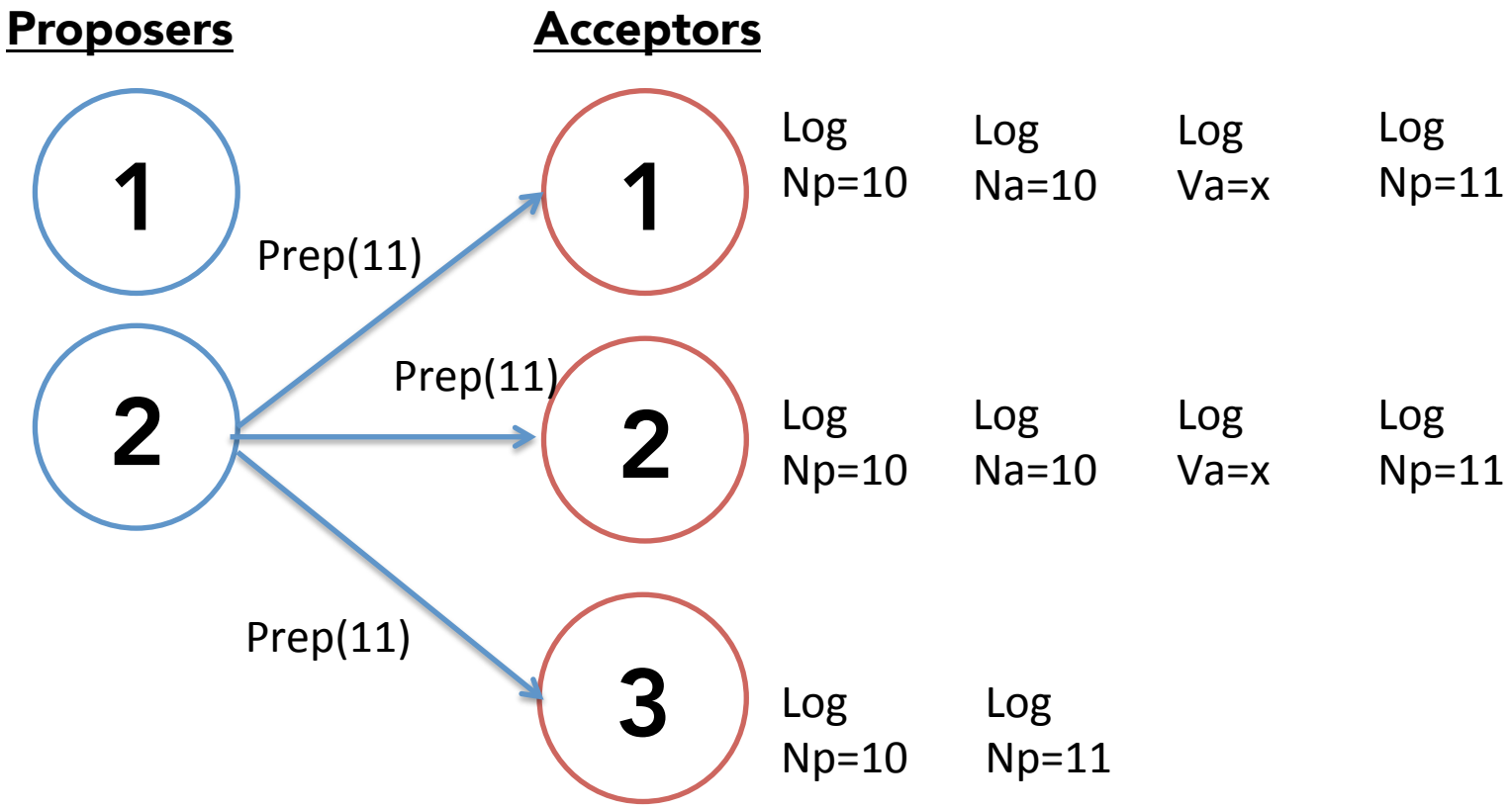
Example 4


Proposers
Acceptors
1
2
Prep(10)
Prep(10)
Prep(10)
1
2
3
Log
Np=10
Log
Np=10
Log
Np=10

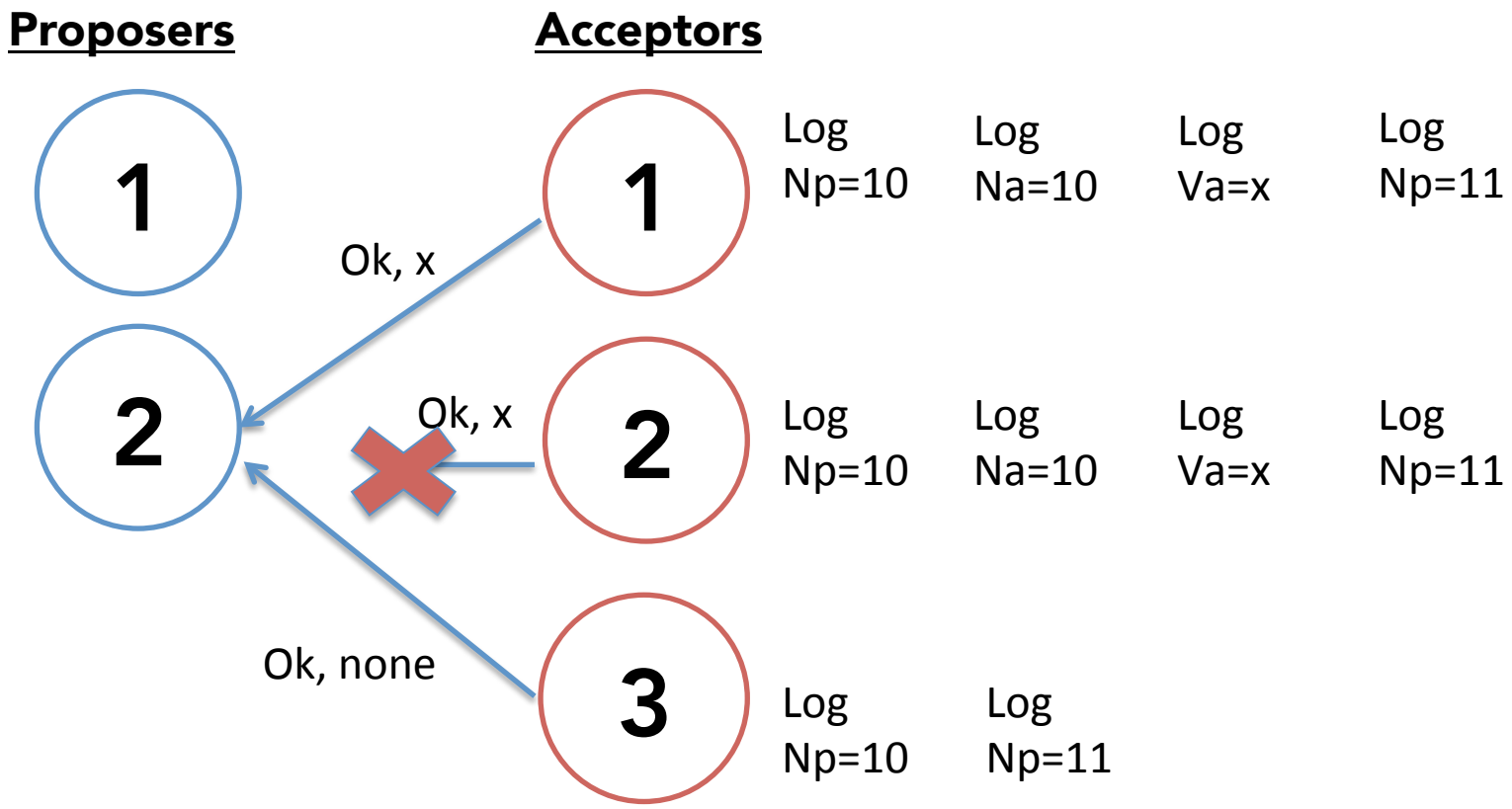

Proposers
Acceptors
1
2
1
2
3
Ok, None
Ok, None
Ok, None
Log
Np=10
Log
Np=10
Log
Np=10


Proposers
Acceptors
1
2
1
2
3
Acc(10,x)
Acc(10,x)
Acc(10,x)
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10

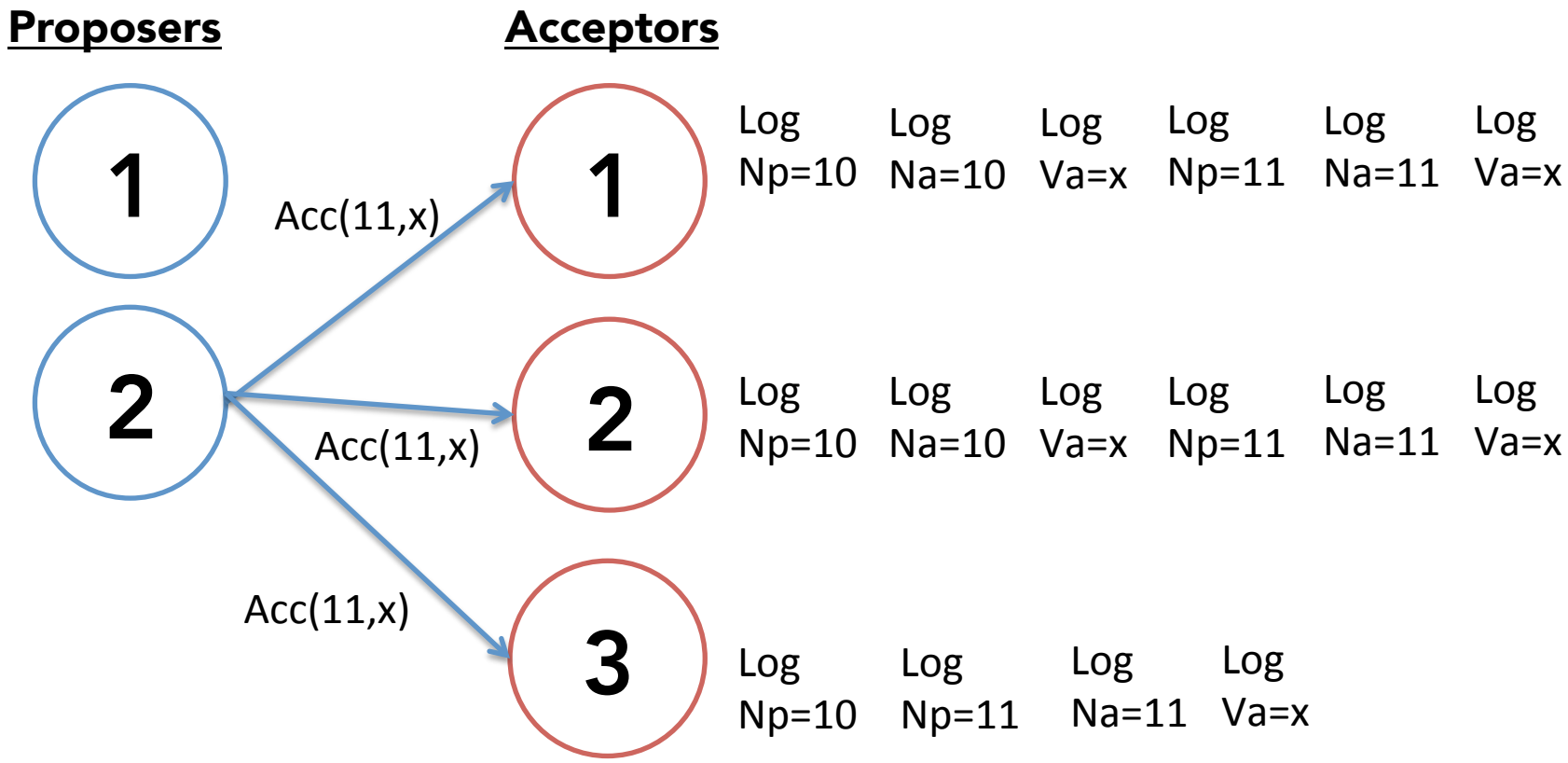

Proposers
Acceptors
1
2
Ok
Ok
1
2
3
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=10


Proposers
Acceptors
1
2
1
2
3
Prep(11)
Prep(11)
Prep(11)
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=11
Log
Np=10
Log
Na=10
Log
Va=x
Log
Np=11
Log
Np=10
Log
Np=11

P2 learns value is x; has to propagate it even if hears from non-majority



P2 learns committed value is x, tells A3



Decided Message Omitted for Brevity

Summary

- Consistency: single-copy semantics

- Replicated state machines provide single-copy
  - Key issue: agreeing on order of operations
  - Hard case: network partition

- Paxos allows replicas to reach consensus, in presence of machine and network failures
  - Widely used in practice [Chubby, ZooKeeper, etc.]