Paxos

Yang Wang

Paxos and consensus

- **Termination**: Every correct process decides some value when there are no asynchronous events.
- **Validity**: If all processes propose the same value v, then all correct processes decide v.
- **Integrity**: Every correct process decides at most one value, and if it decides some value v, then v must have been proposed by some process.
- **Agreement**: Every correct process must agree on the same value.
- Validity and integrity are trivial in fail-stop model. Let's focus on termination and agreement.

Paxos

Leslie Lamport

2013 Turing Award Winner

The Part Time Parliament

2012 SIGOPS Hall of Fame Award

Paxos

Barbara Liskov

2008 Turing Award Winner

Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems.

2012 SIGOPS Hall of Fame Award

The paper introduces a replication protocol very similar to what is now known as Paxos. That protocol has become the standard for consistent, fault-tolerant state-machine replication, and is widely used in data centers to keep the state consistent despite failures and reconfiguration.

Primary backup and Paxos

- Primary backup
  - The system asks a primary to decide the next request, but if the primary is not responding, the system does not know how to proceed.

- Paxos' first key idea
  - A decision is made if it is agreed by a majority of replicas.

Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority of replicas.
- It can guarantee "agreement". Why?

Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority of replicas.
- It can guarantee "agreement". Why?
- How many replicas does Paxos need?

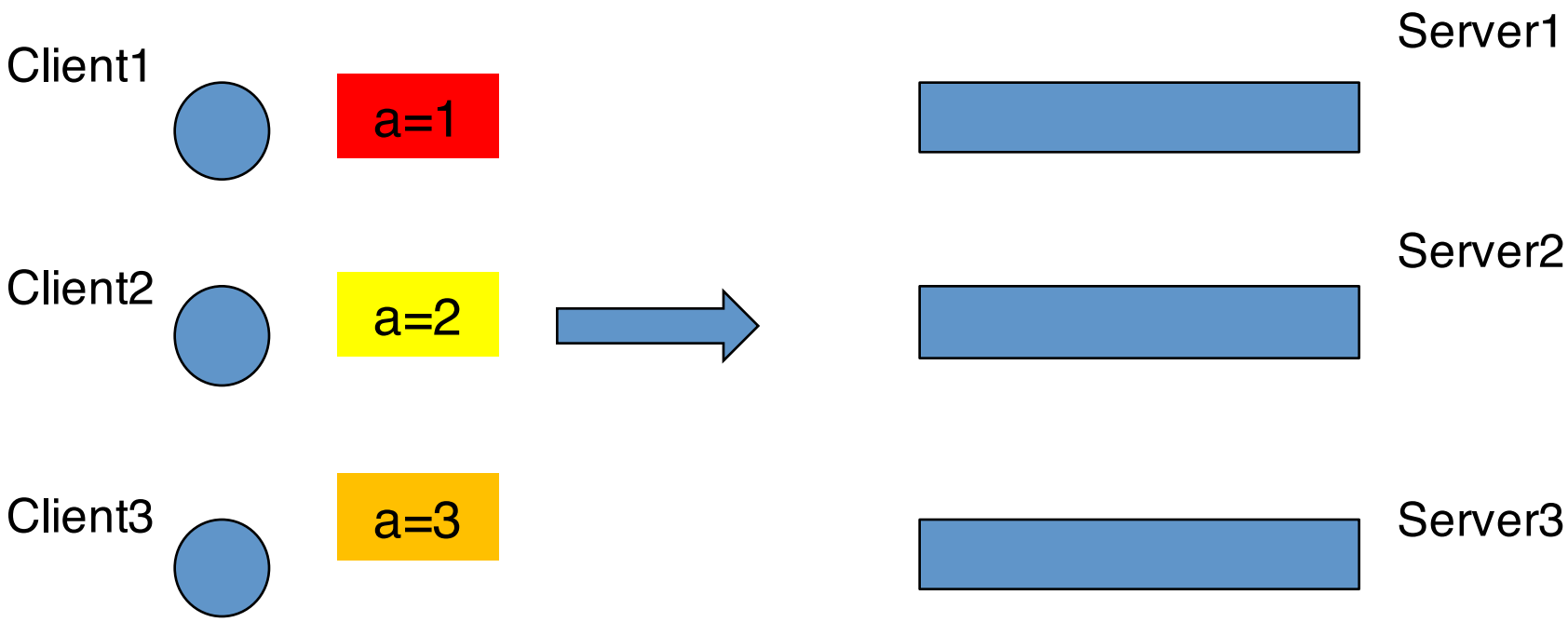
Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority of replicas.
- It can guarantee "agreement". Why?
- How many replicas does Paxos need?
  - $N-f>=N/2$, so $N>=2f+1$

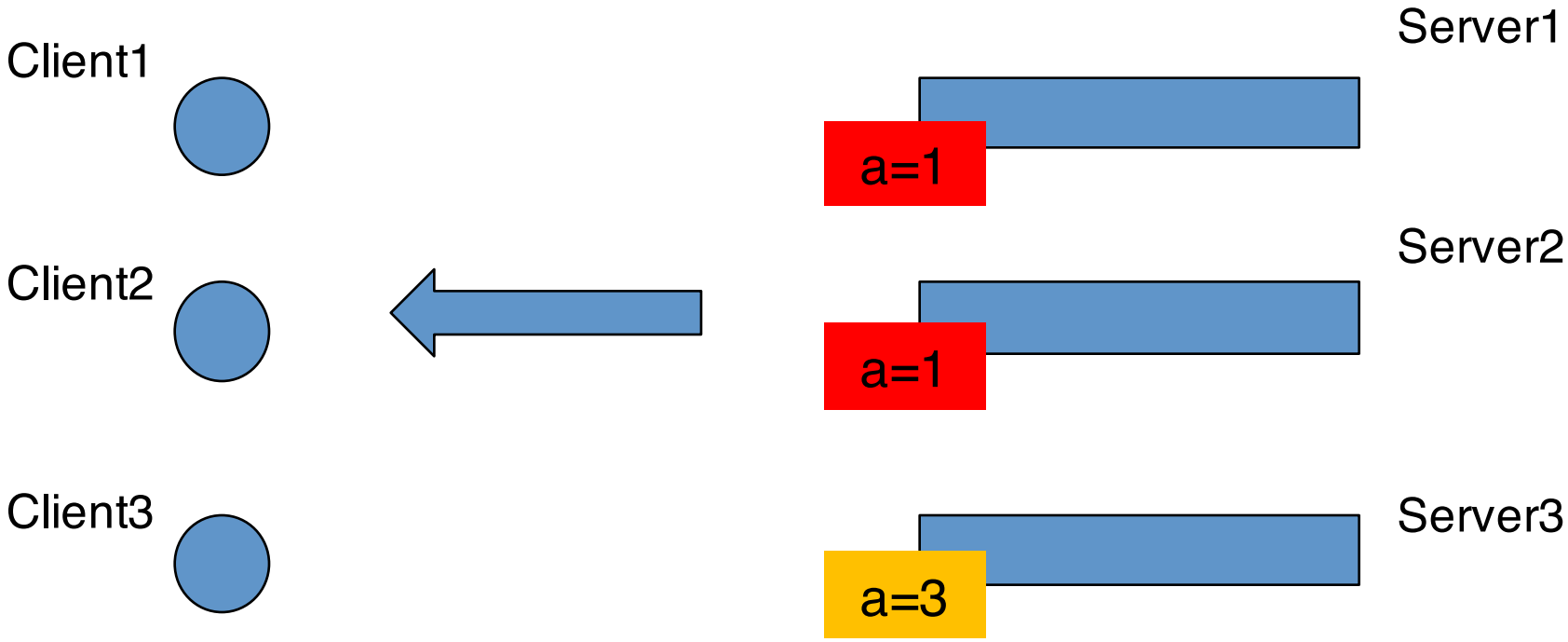
Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority of replicas.
- It can guarantee "agreement". Why?
- How many replicas does Paxos need?
  - $N-f>=N/2$, so $N>=2f+1$
- Does it guarantee "termination" when there are no asynchronous events?

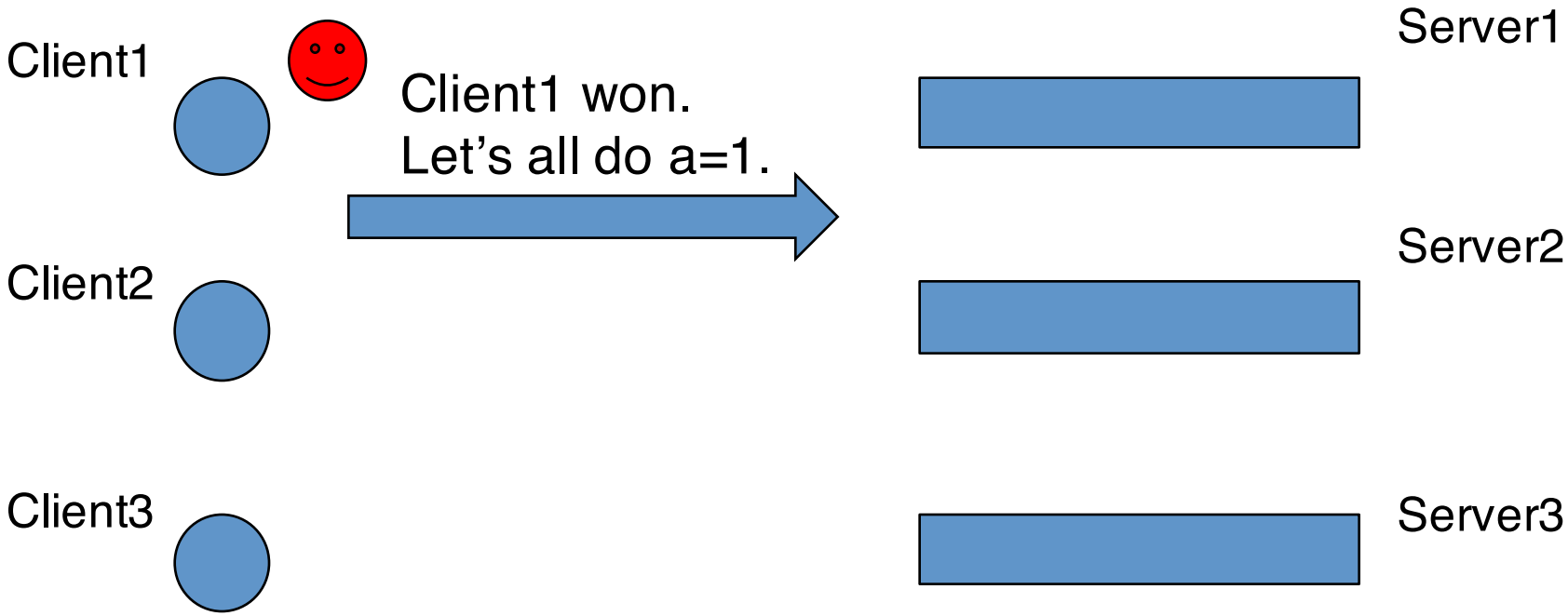
Start from a naïve protocol



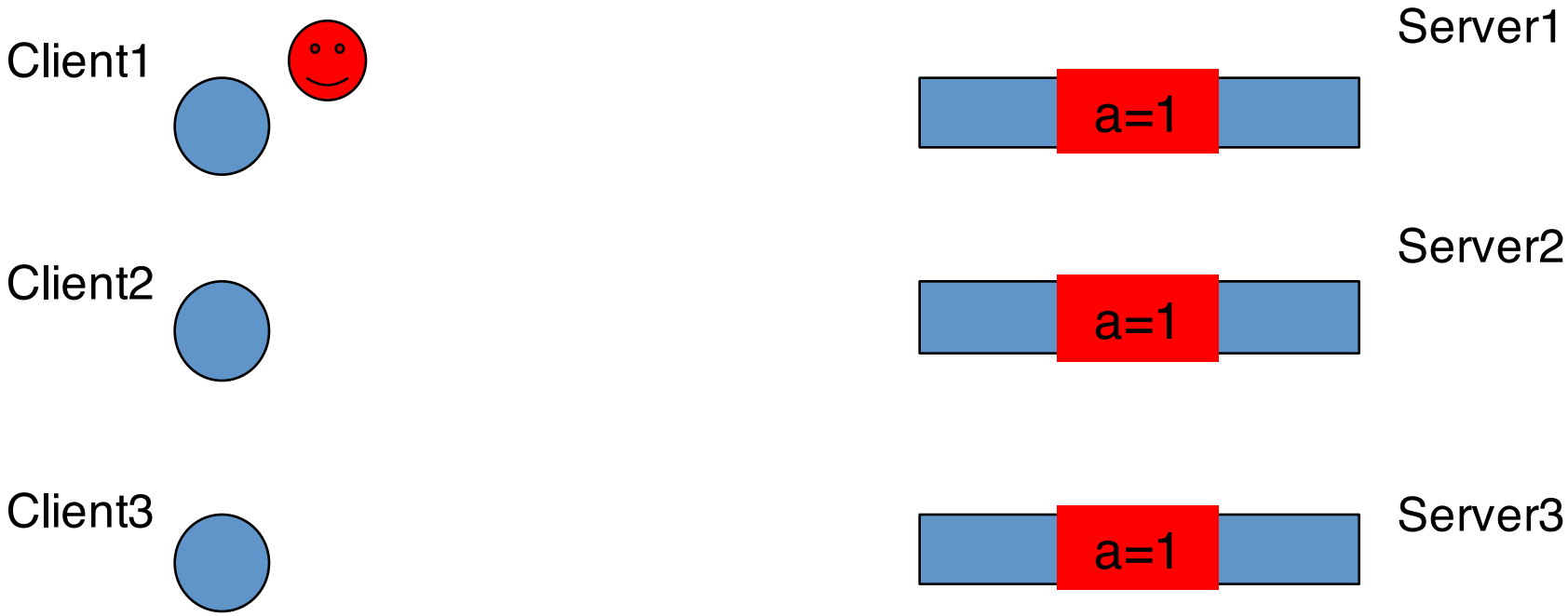
Start from a naïve protocol



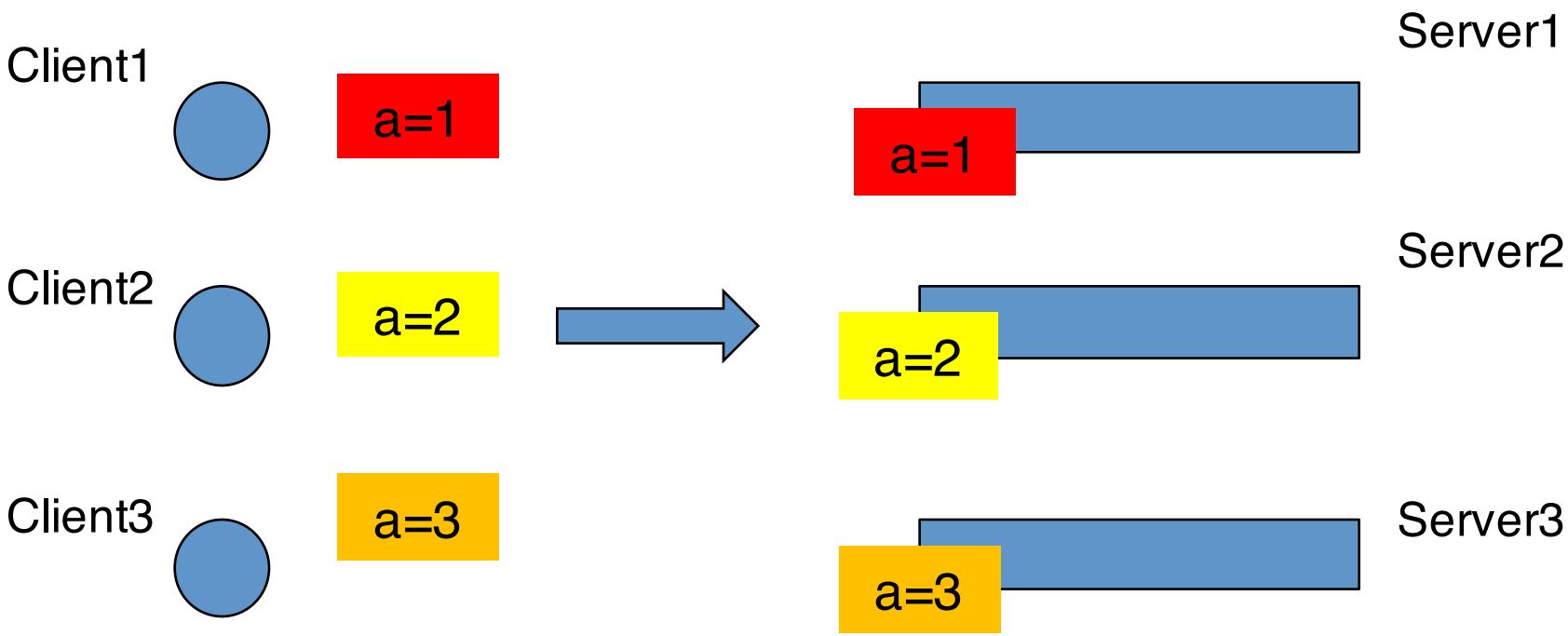
Start from a naïve protocol



Start from a naïve protocol

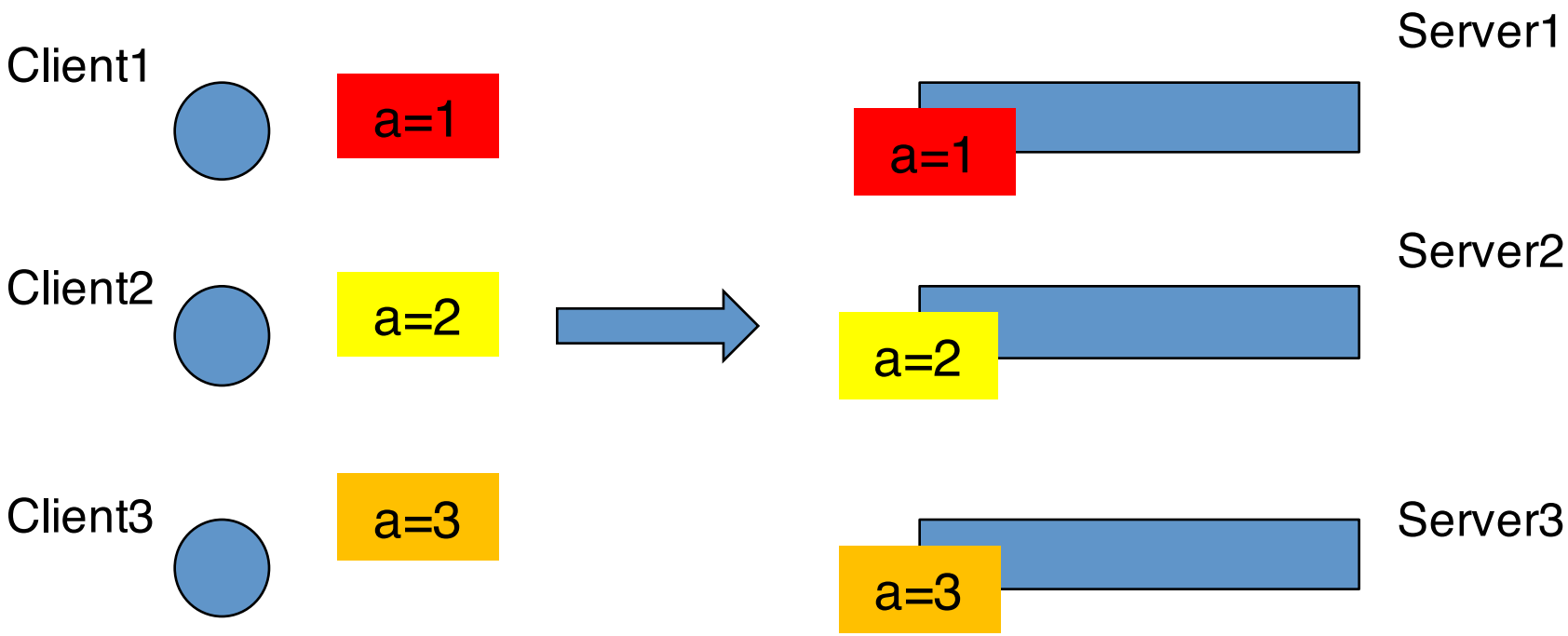


However …



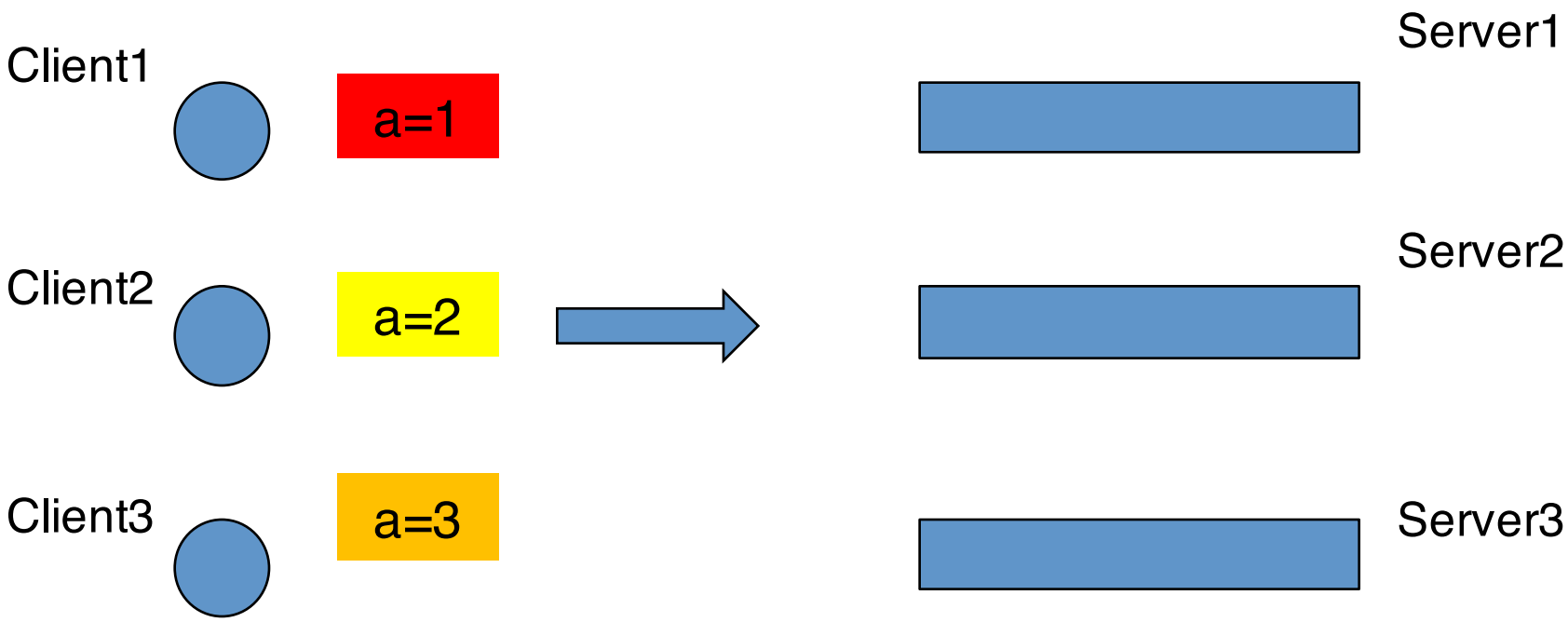
It may happen that no one gets a majority of votes. In this case, the protocol cannot terminate, even when there are no asynchronous events.

However …

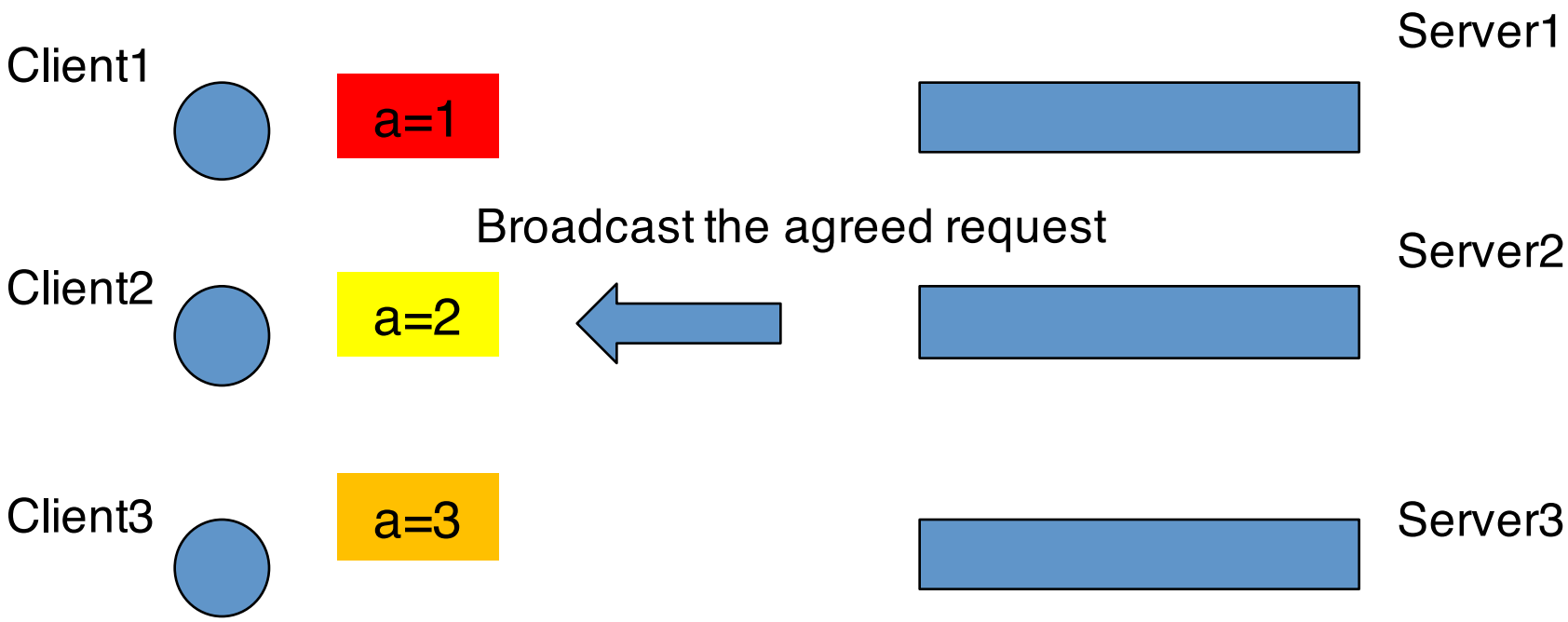


It seems that we cannot expect the system to reach agreement in one round.

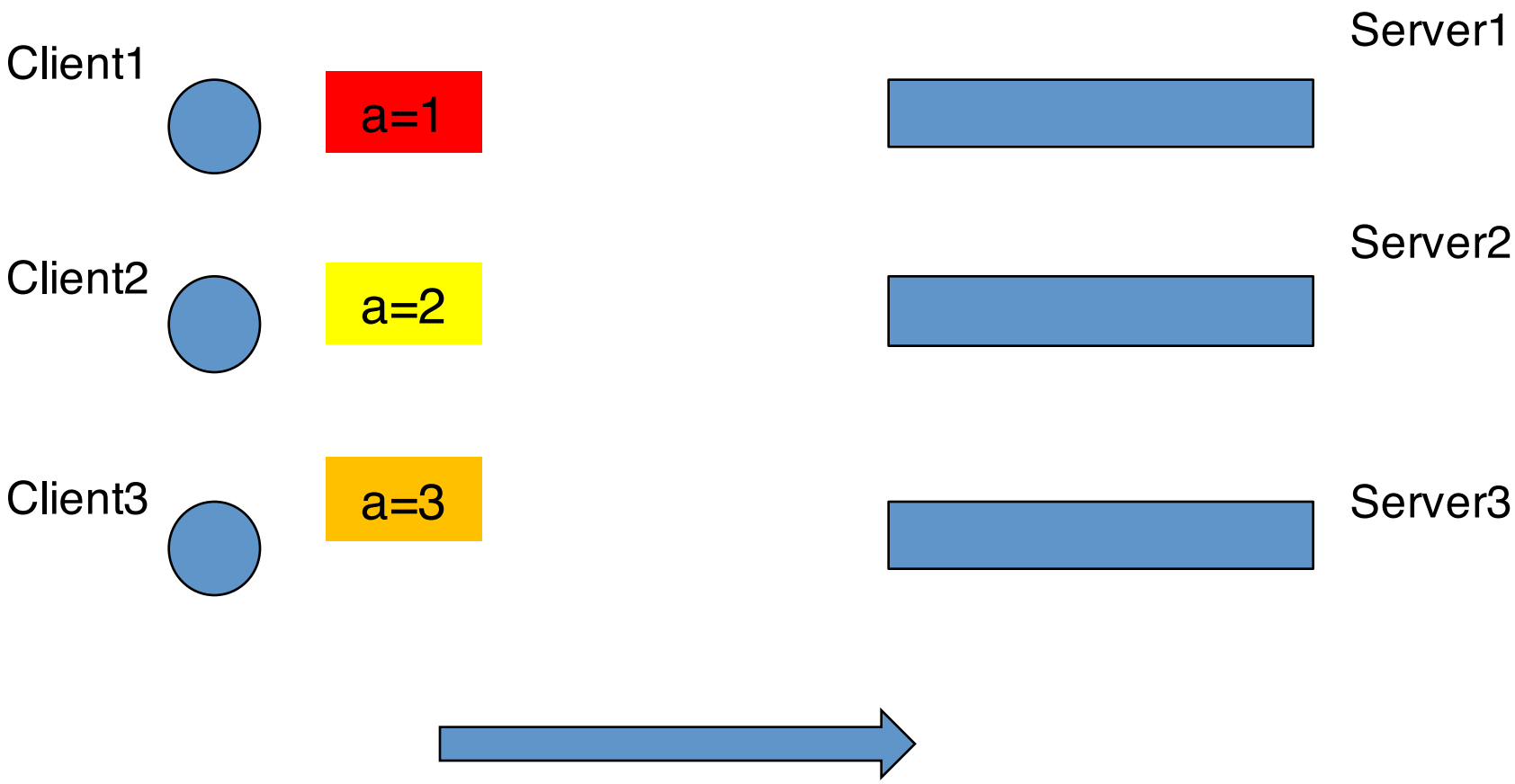
Try another protocol



Try another protocol

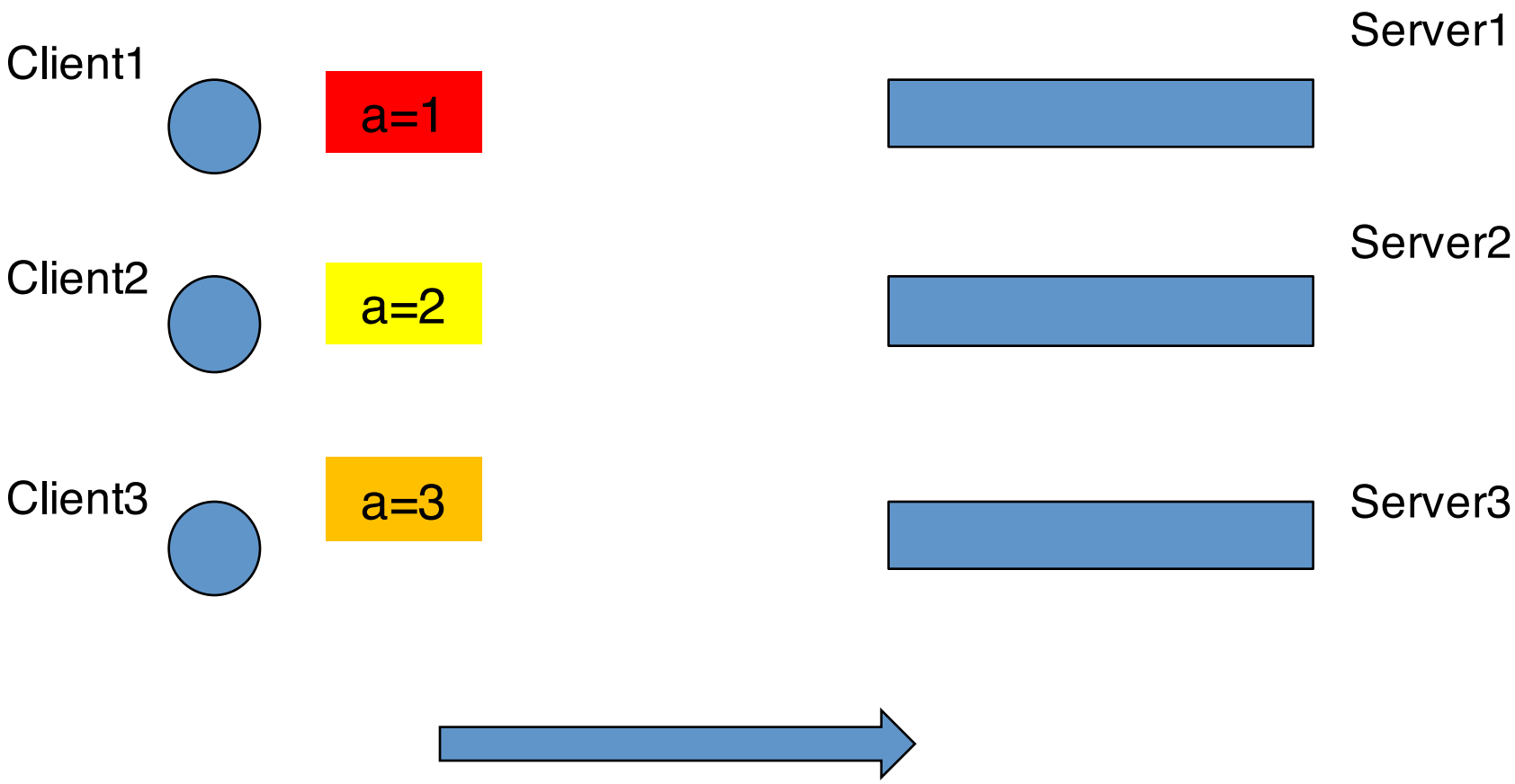


Try another protocol



If any request gets a majority of votes, ask servers to execute it.
If not, the client resends its request.

Try another protocol



If any request gets a majority of votes, ask servers to execute it.
If not, the client resends its request.

Any problems?

Try another protocol

- Problem 1: may not terminate
  - If unlucky, it may happen that no one gets a majority of votes in every round

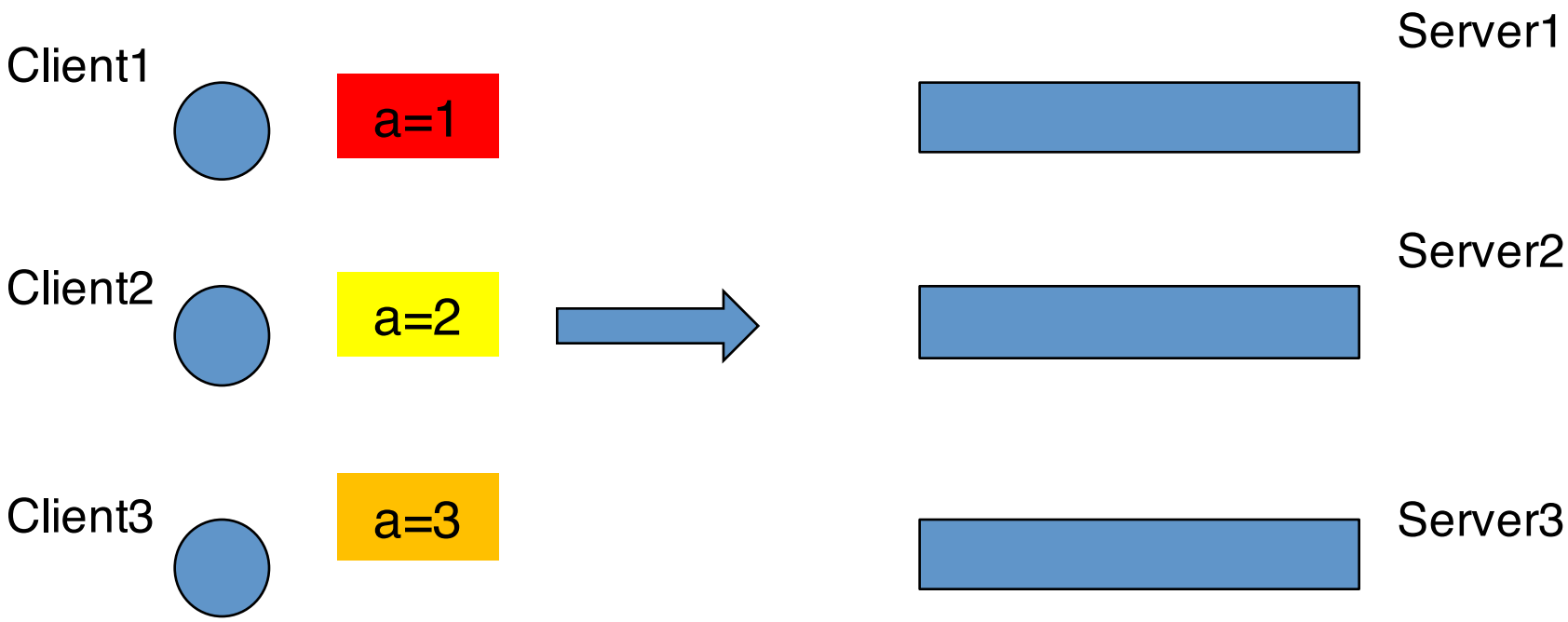
Try another protocol

- Problem 1: may not terminate
  - If unlucky, it may happen that no one gets a majority of votes in every round
  - Solution: a client should sleep a while before resending (random and exponentially growing sleeping time)

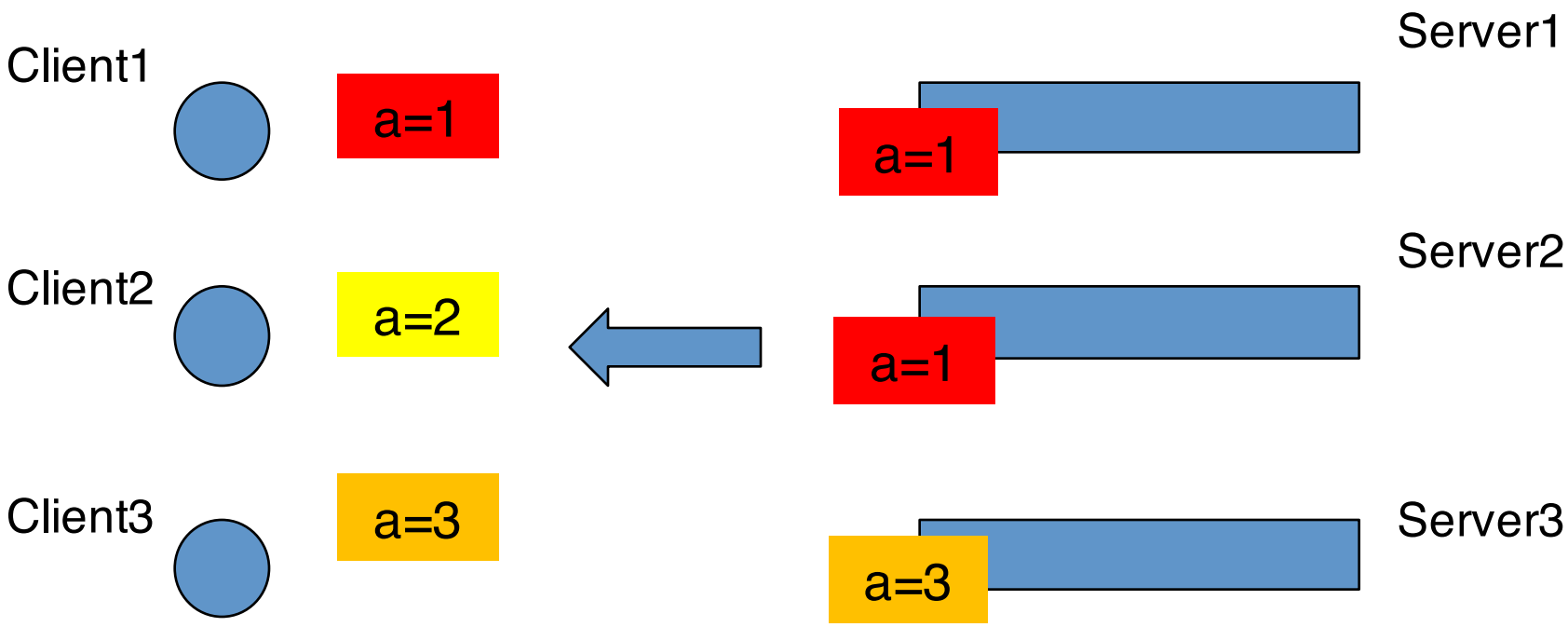
Try another protocol

- Problem 1: may not terminate
  - If unlucky, it may happen that no one gets a majority of votes in every round
  - Solution: a client should sleep a while before resending (random and exponentially growing sleeping time)
- Problem 2: may violate agreement

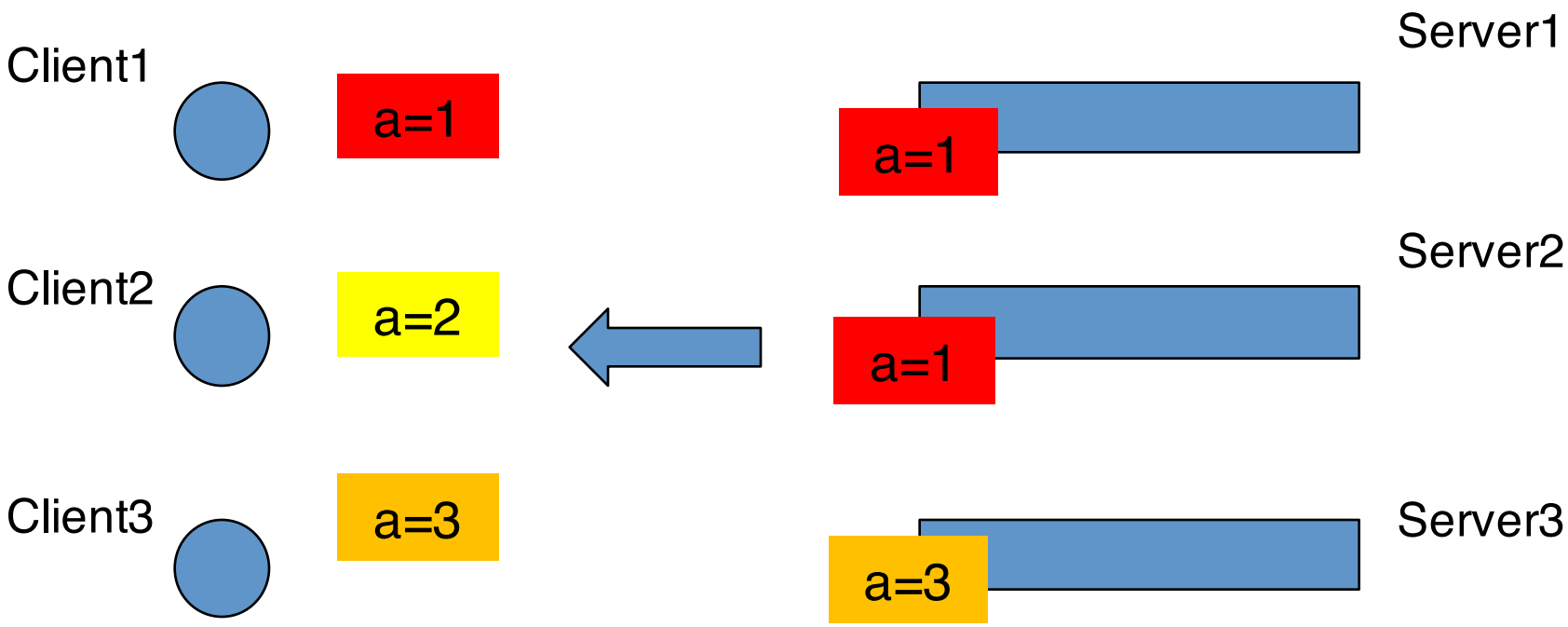
Example of violating agreement



Example of violating agreement

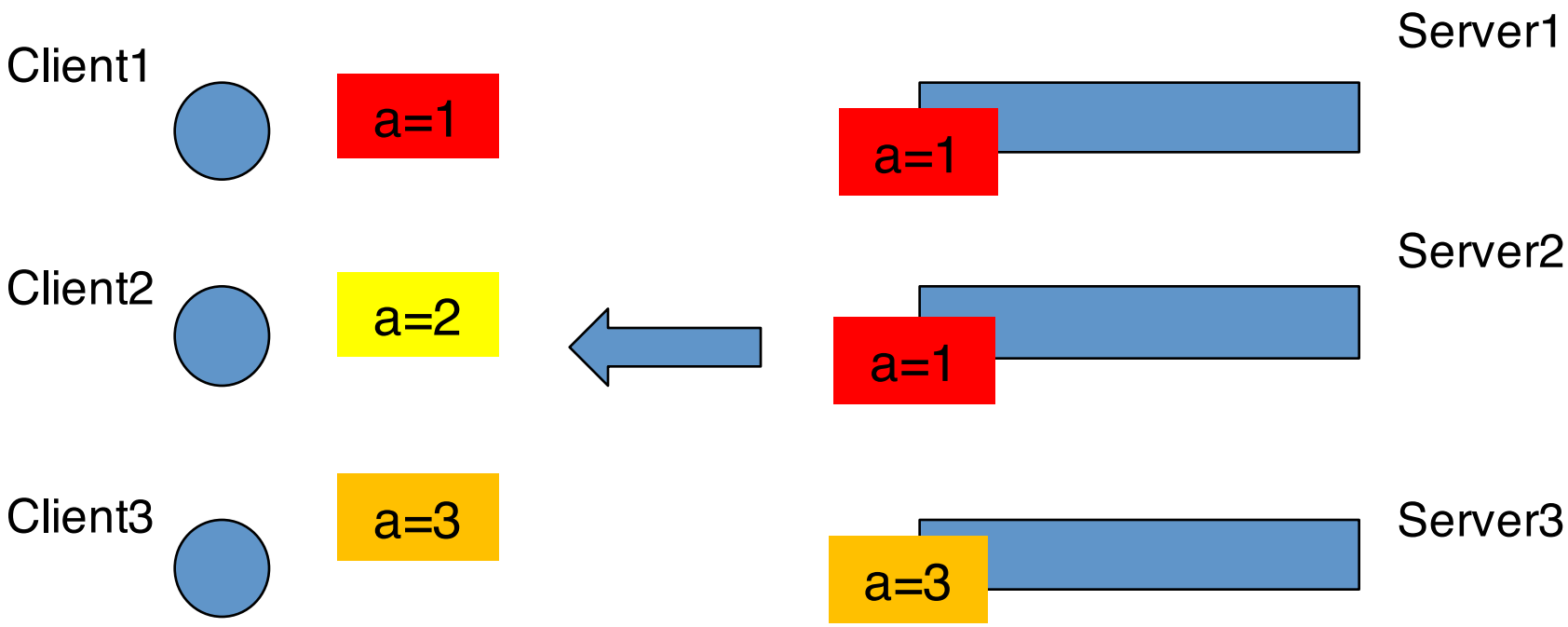


Example of violating agreement



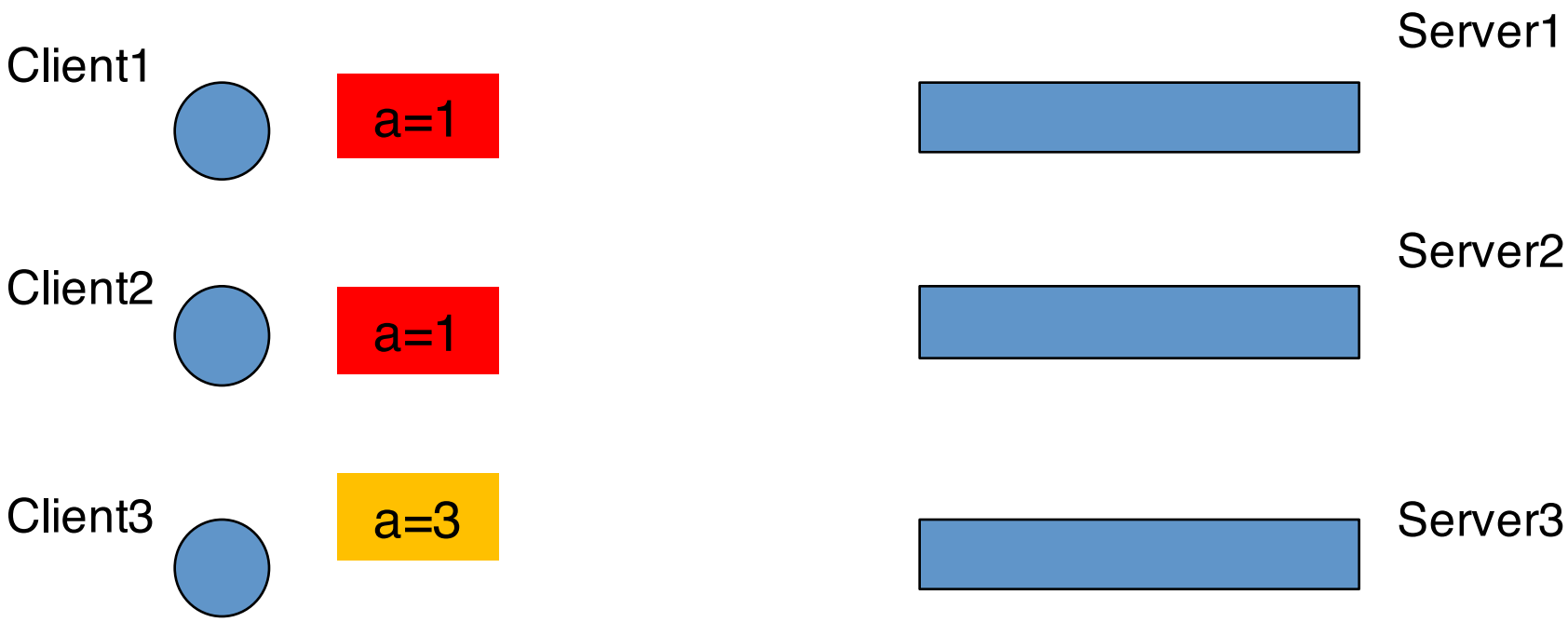
Because of potential failures, a client cannot wait for all servers to respond. It must take actions when f+1 servers respond.

Example of violating agreement



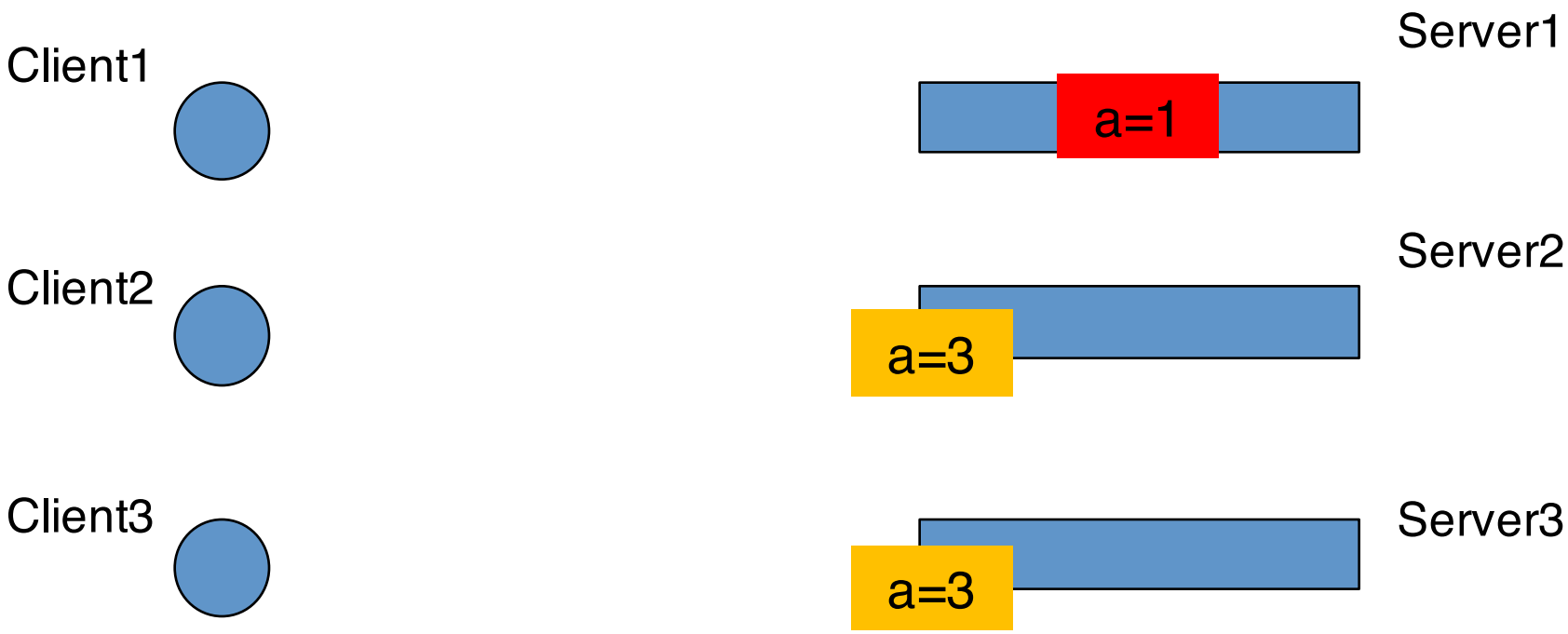
Suppose client1 and client2 got responses from server1 and server2
Suppose client3 got responses from server2 and server3

Example of violating agreement



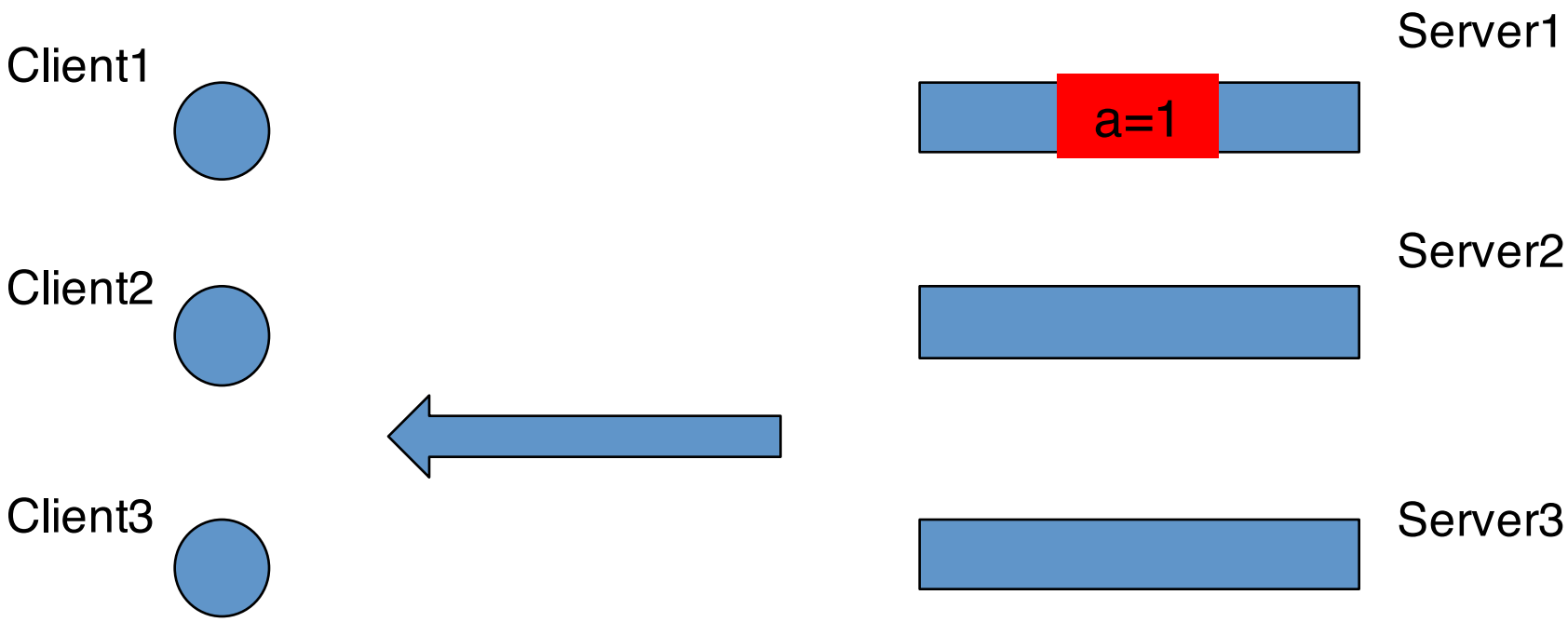
Suppose client1 and client2 got responses from server1 and server2
Suppose client3 got responses from server2 and server3
Client1 and client2 will ask servers to “execute a=1”
Client3 will resend “a=3”

Example of violating agreement



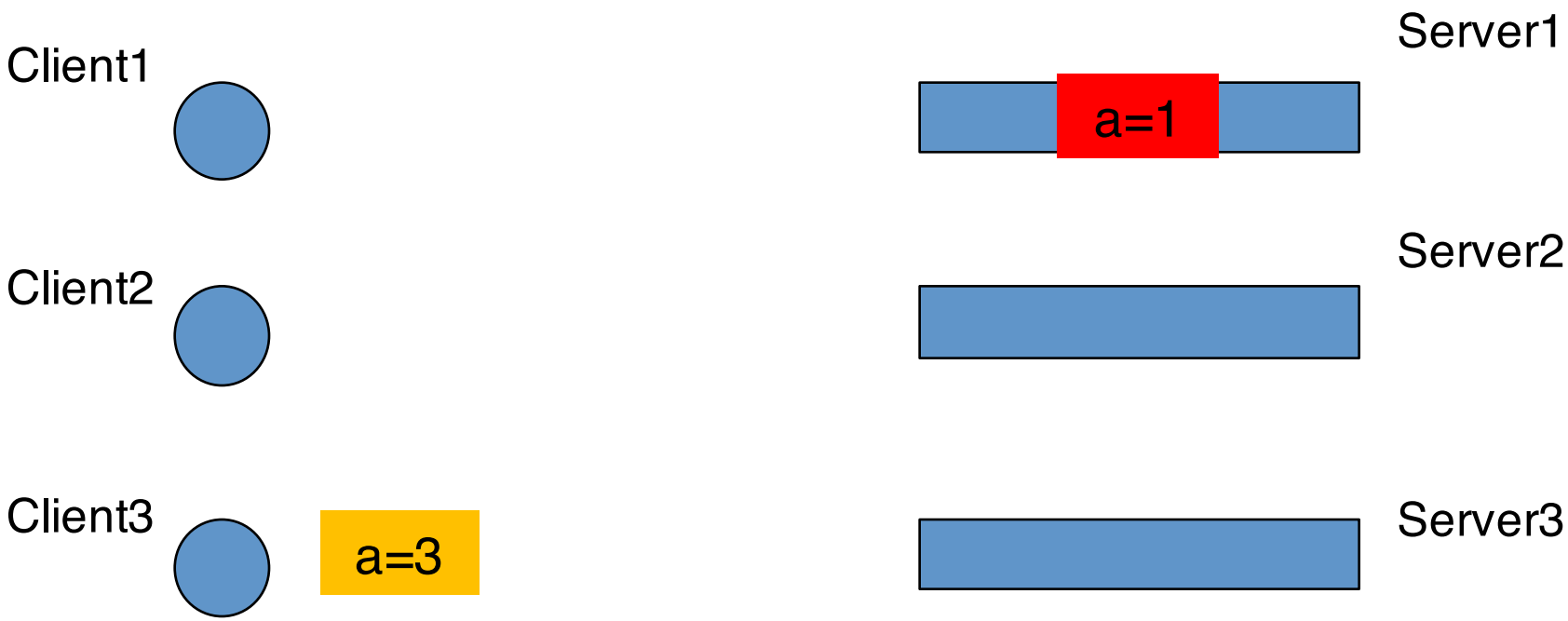
Suppose server1 receives “execute a=1”
Suppose server2 and server3 receive “propose a=3” but miss “execute a=1” because of a long network delay

Example of violating agreement



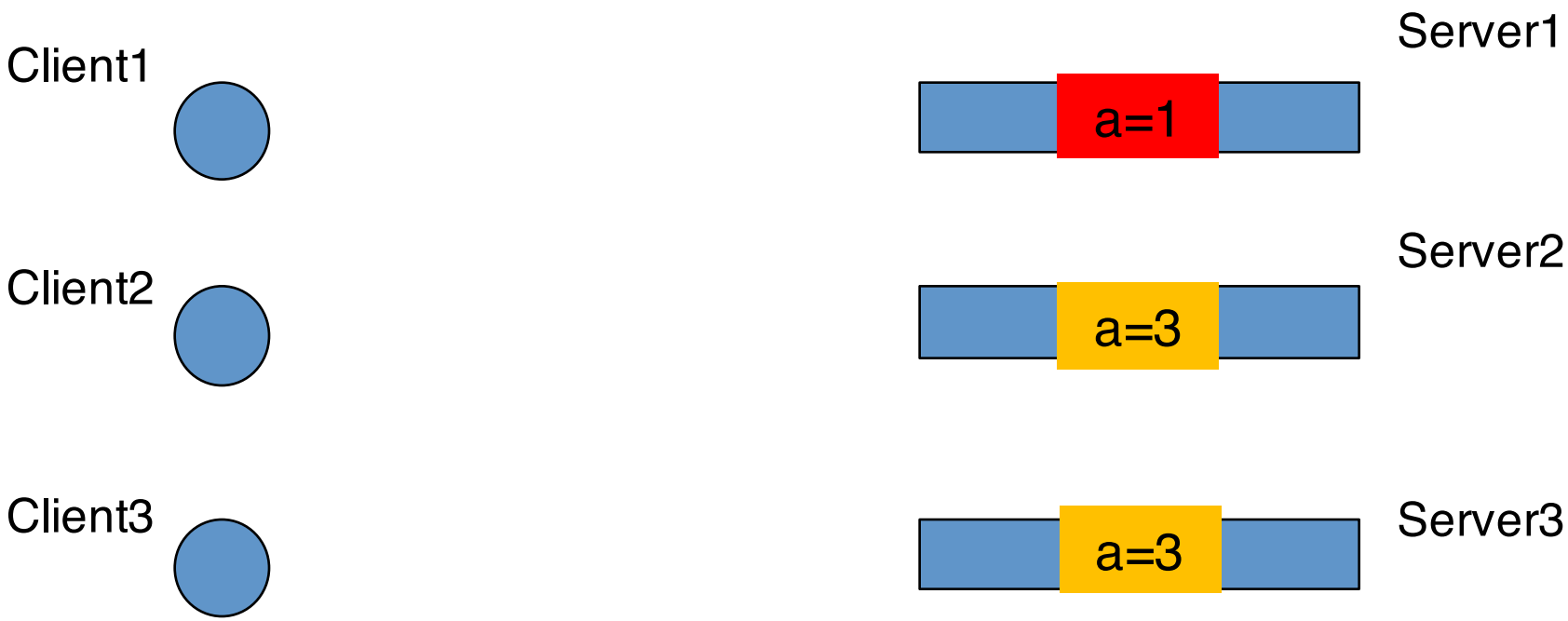
Suppose server1 receives “execute a=1”
Suppose server2 and server3 receive “propose a=3” but miss “execute a=1” because of a long network delay
Server2 and server3 will agree on “a=3”

Example of violating agreement



Suppose server1 receives “execute a=1”
Suppose server2 and server3 receive “propose a=3” but miss “execute a=1” because of a long network delay
Server2 and server3 will agree on “a=3”
Client3 believes it wins the and asks all servers to “execute a=3”

Example of violating agreement



Violating agreement ……
Problem: servers may fail, so a client cannot expect to know status of all servers. Message delivery can have an arbitrary delay.

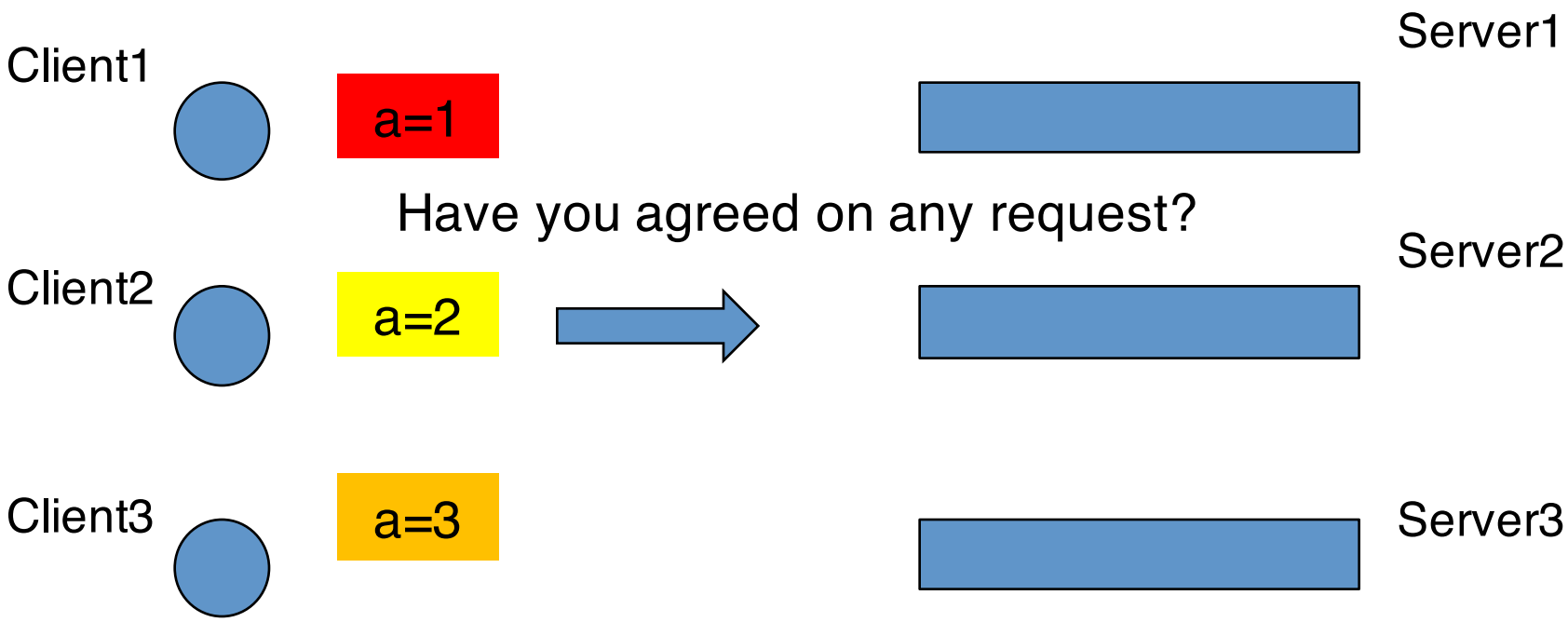
Summarize results so far

- Paxos' first key idea
  - A decision is made if it is agreed by a majority.
- Problem: it may happen that no proposal can get a majority of votes
  - We need to use multiple rounds
  - A server may have to change what it agrees on
  - This is dangerous …….
  - If already agreed by a majority, then no one should make or agree on a different proposal

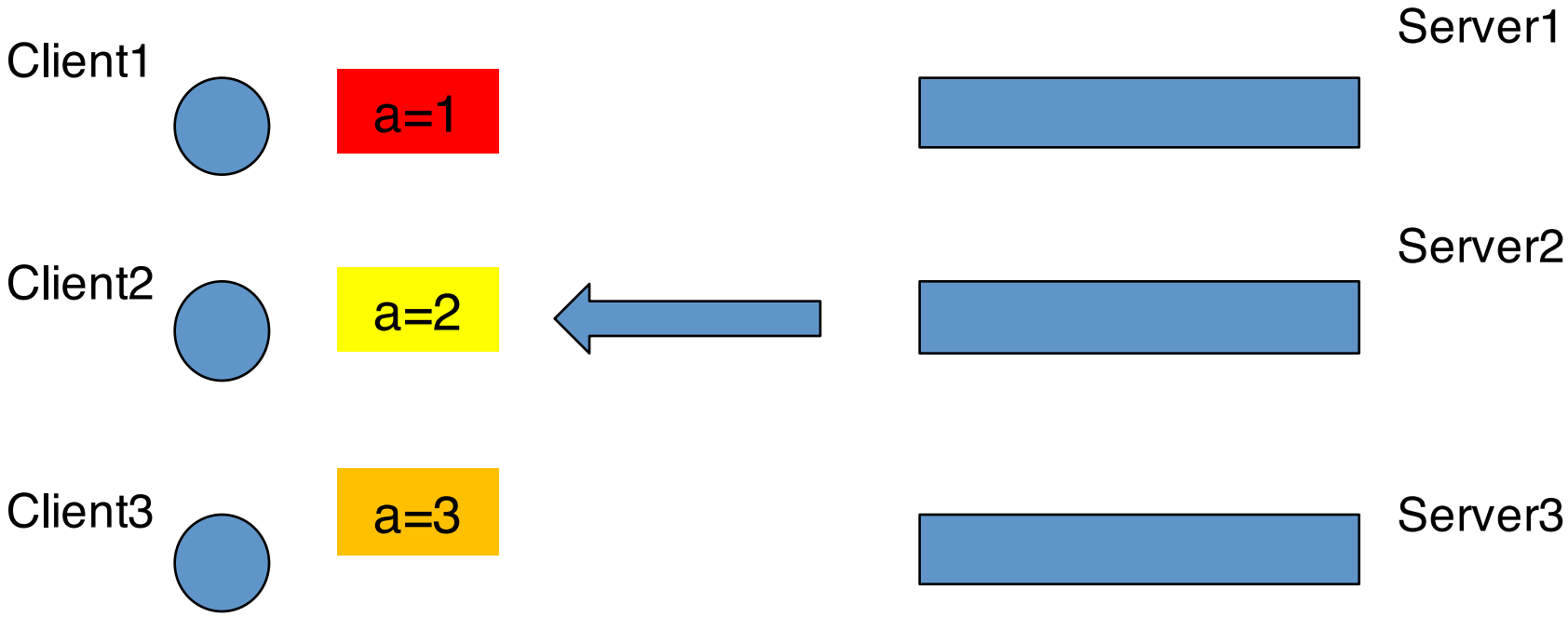
Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority.
- Paxos' second key idea
  - Before a client makes a proposal, it must ask servers to understand what has been agreed

Improving the protocol

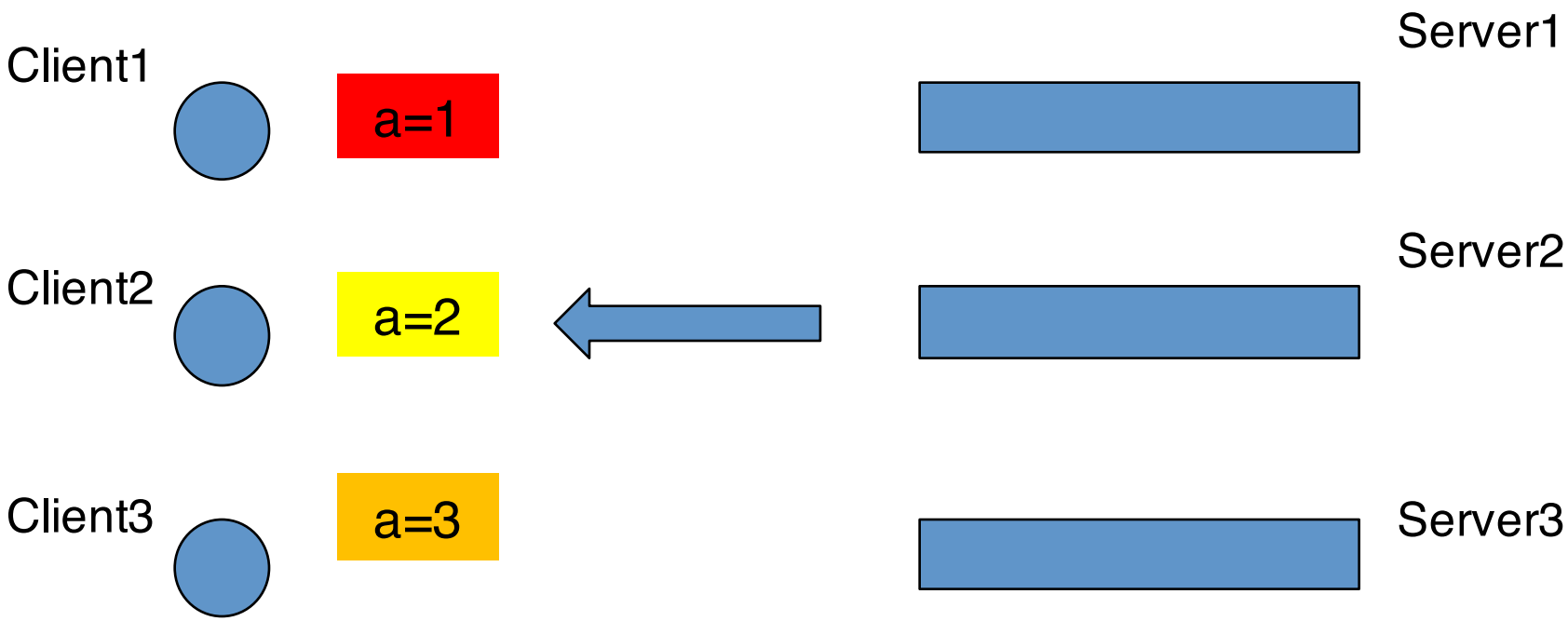


Improving the protocol



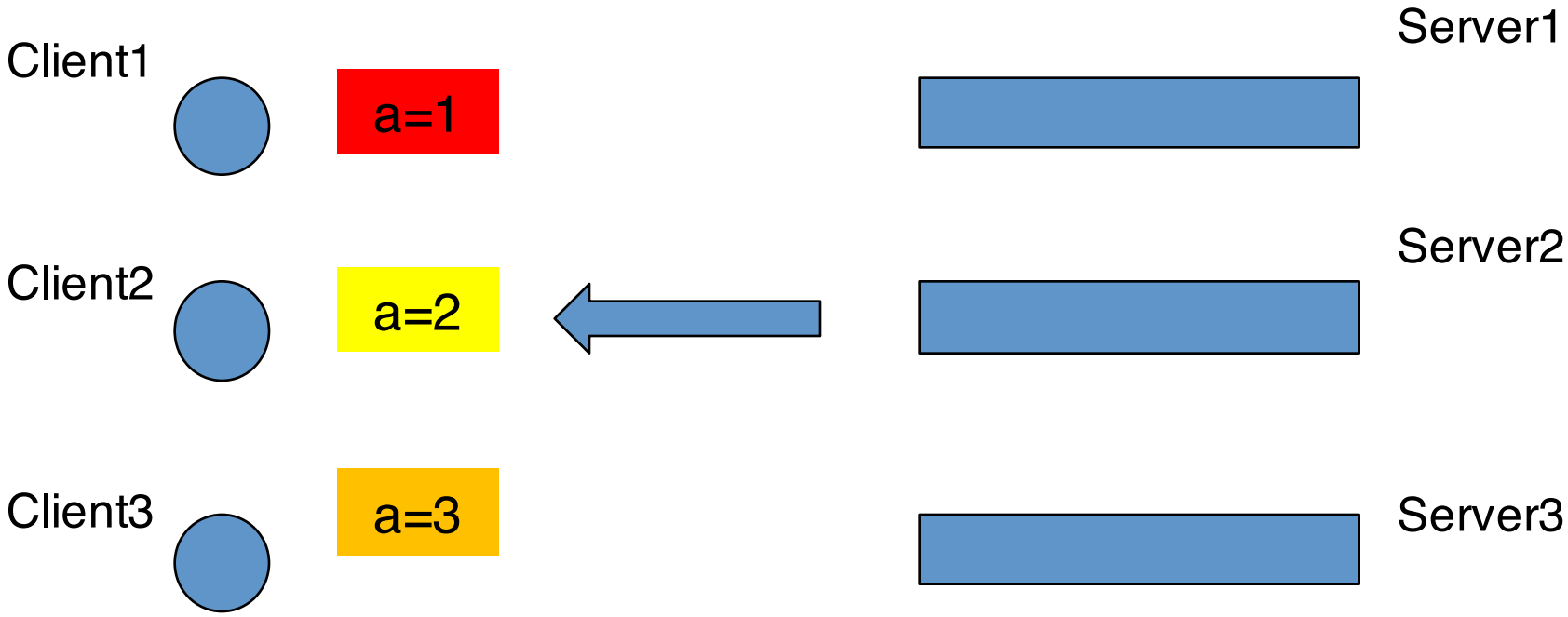
Remember a client can only expect f+1 responses.

Improving the protocol



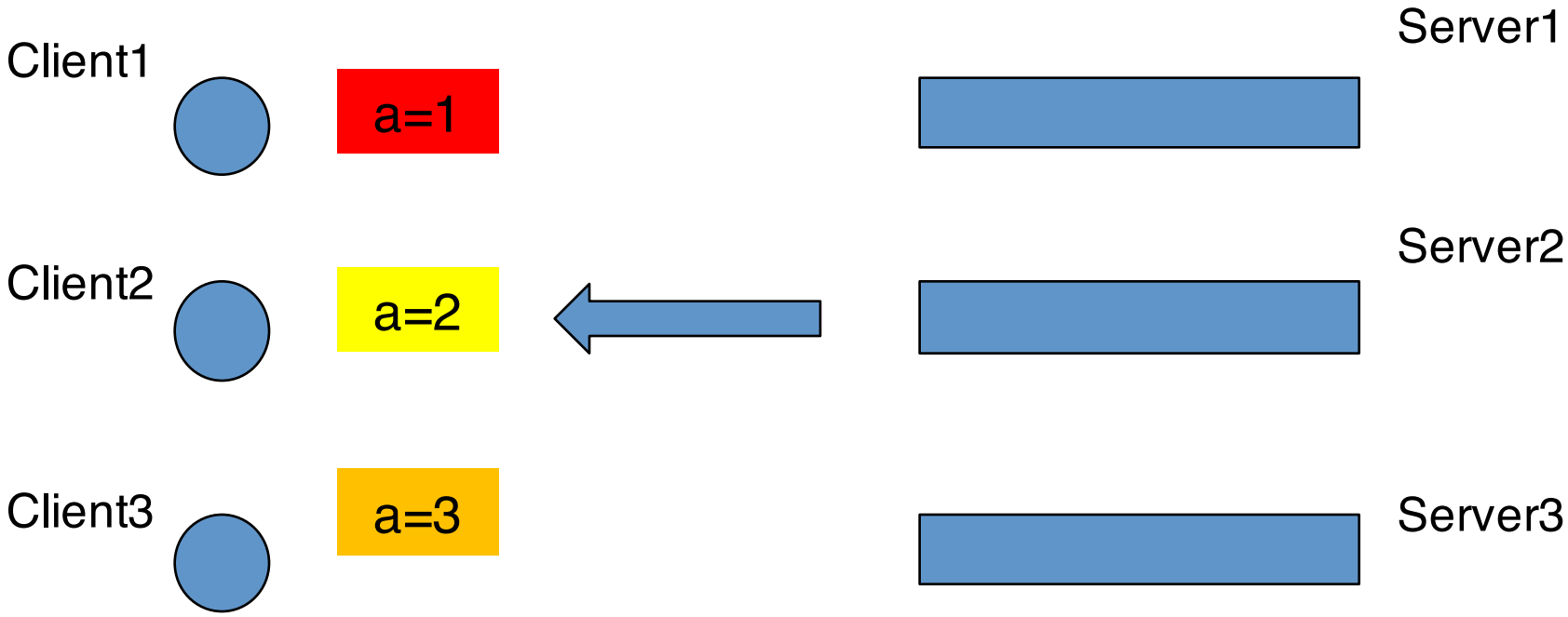
Case 1: f+1 servers respond the same value v. Then obviously the client should not send a new proposal. The client should ask all servers to “execute v”. Why?

Improving the protocol



Case 2: f+1 servers respond “empty”. Is it safe for the client to make a new proposal?

Improving the protocol



Case 2: f+1 servers respond “empty”. Is it safe for the client to make a new proposal? No. Some outstanding proposals may be in flight.

Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority.
- Paxos' second key idea
  - Before a client makes a proposal, it must ask servers to understand what has been agreed
  - But simply asking a server's status does not work, because some proposals may arrive after the server responds.

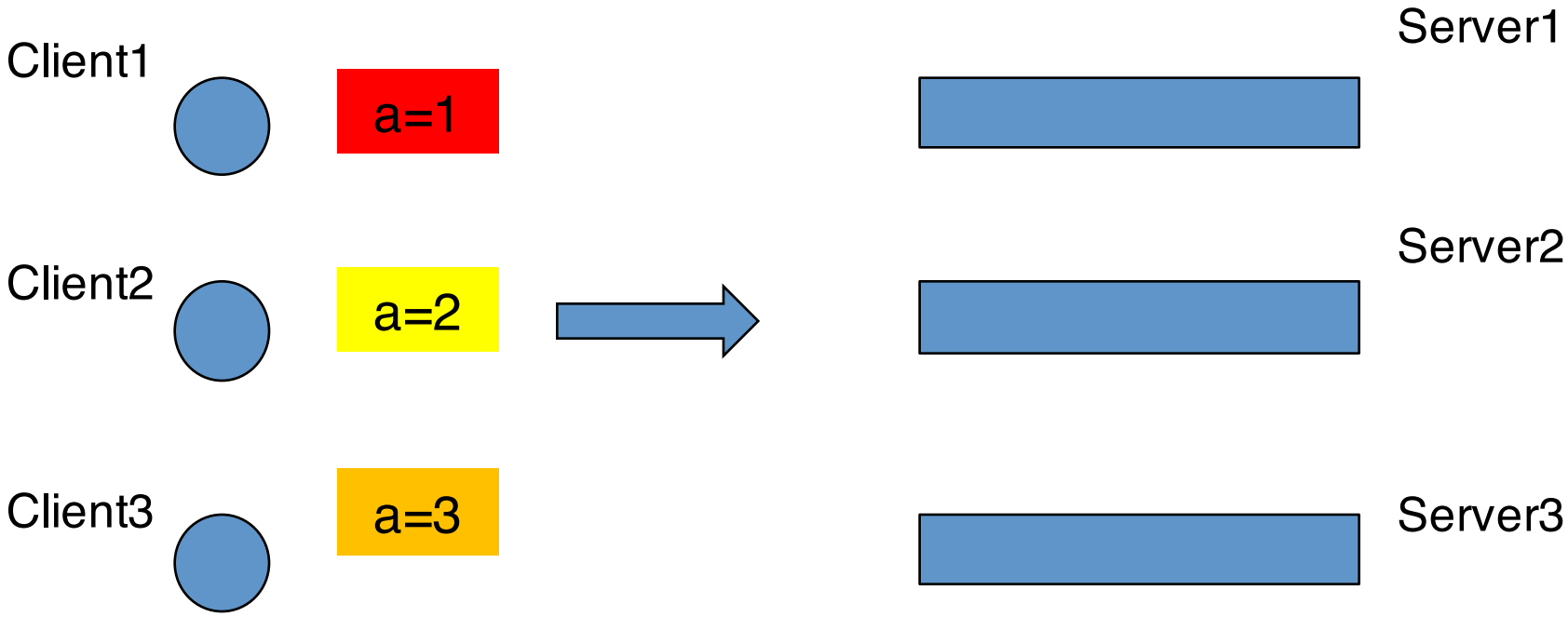
Paxos

- Paxos' first key idea
  - A decision is made if it is agreed by a majority.
- Paxos' second key idea
  - Before a client makes a proposal, it must ask servers to understand what has been agreed
- Paxos' third key idea
  - Order all proposals somehow
  - When asking a server's status, the client also asks the server to reject all "earlier" outstanding proposals.
    - No "earlier" proposal can be agreed by f+1 after a "later" proposal is made.

Paxos

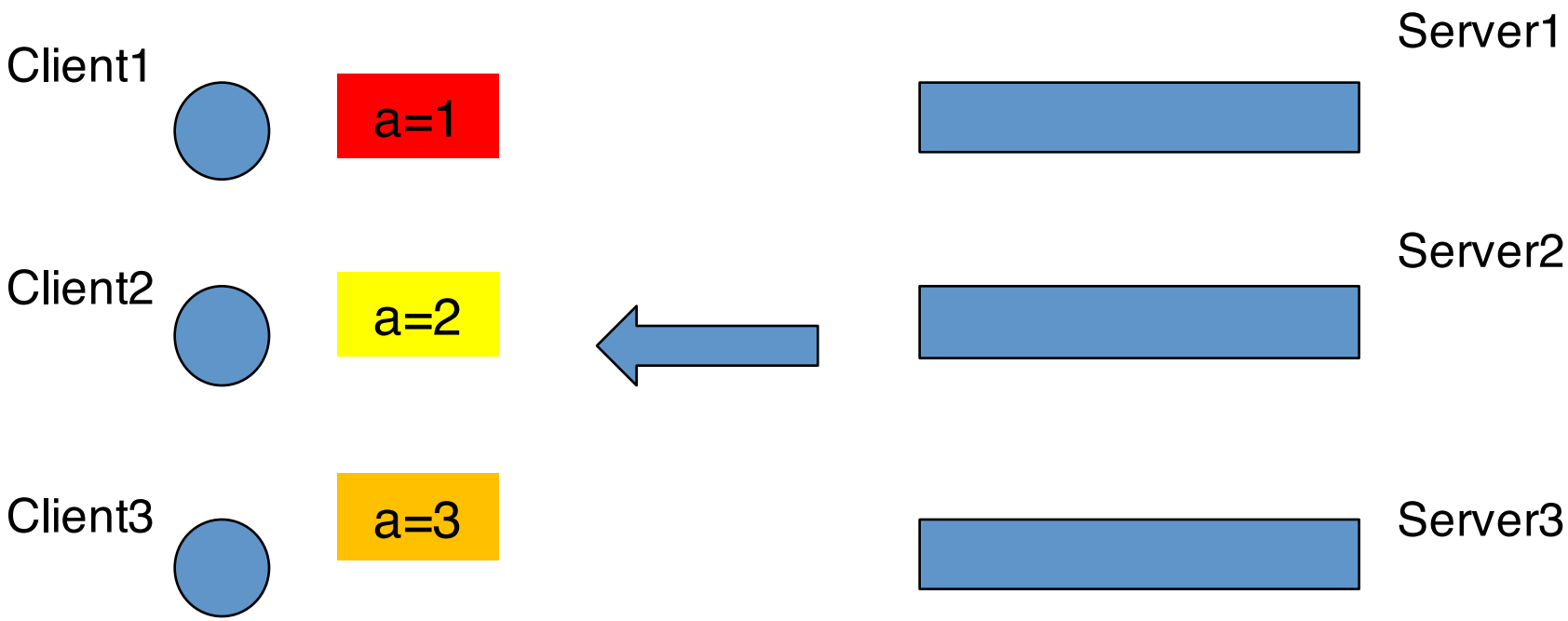
- How to order all proposals?
- One solution: suppose there is a maximum of N clients. Assign a unique number to each proposal
  - Client 0 uses number 0, N, 2N, …
  - Client 1 users number 1, N+1, 2N+1, …
  - ….
- Can we use logic/vector clock?

Paxos – Prepare phase



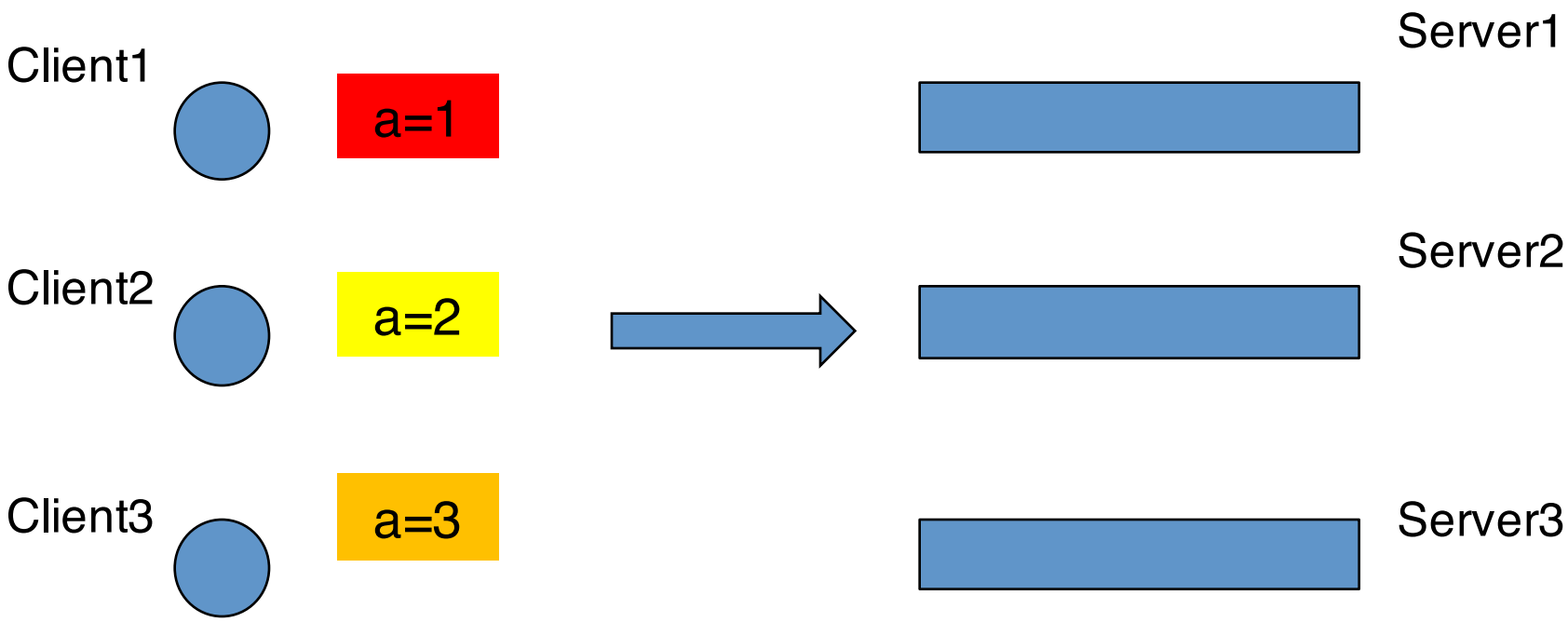
I'd like to make proposal n. What is your status?

Paxos – Prepare phase



Case 1: OK. I promise not to accept earlier proposals. I have not accepted any proposals so far.
Case 2: OK. I promise not to accept earlier proposals. The latest proposal I agree is (v, n').
Case 3: Sorry. I have promised not to accept proposals earlier than m (m>n).

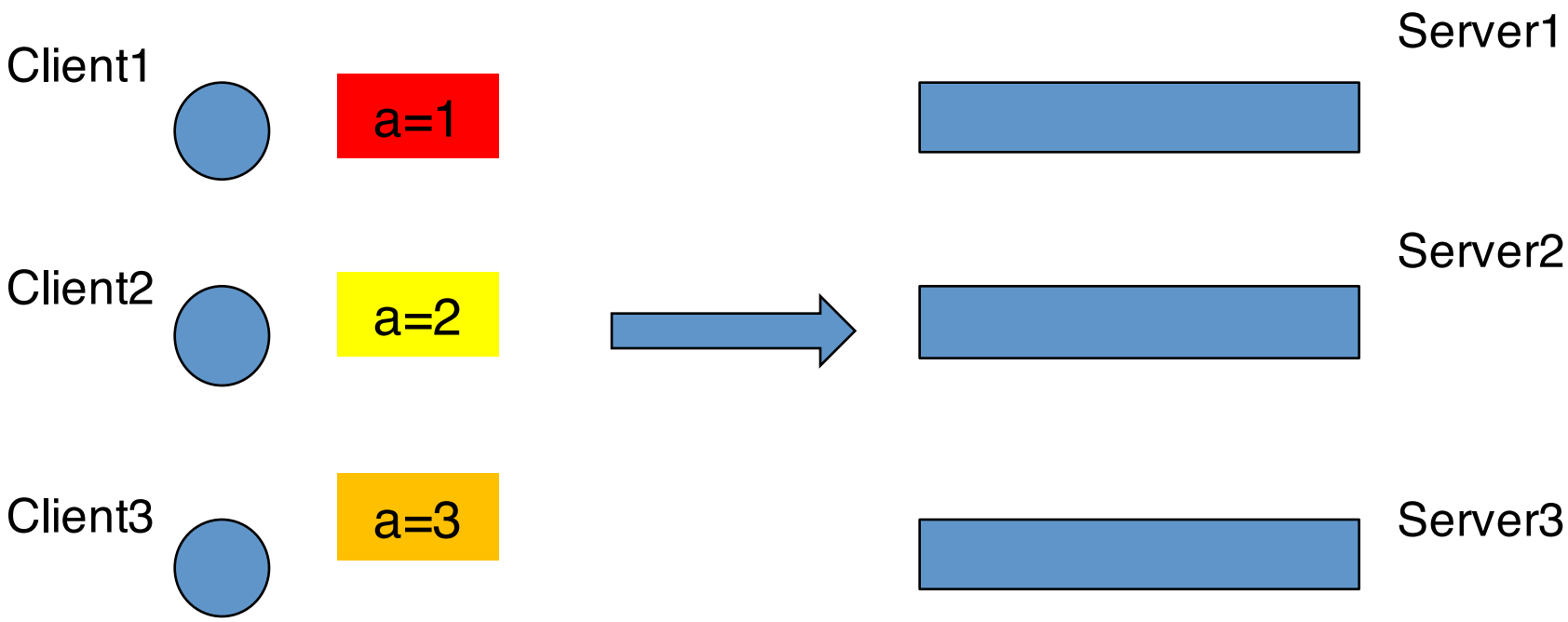
Paxos – Accept phase



A client waits for f+1 responses.

1. If any response is a “sorry”, what to do?

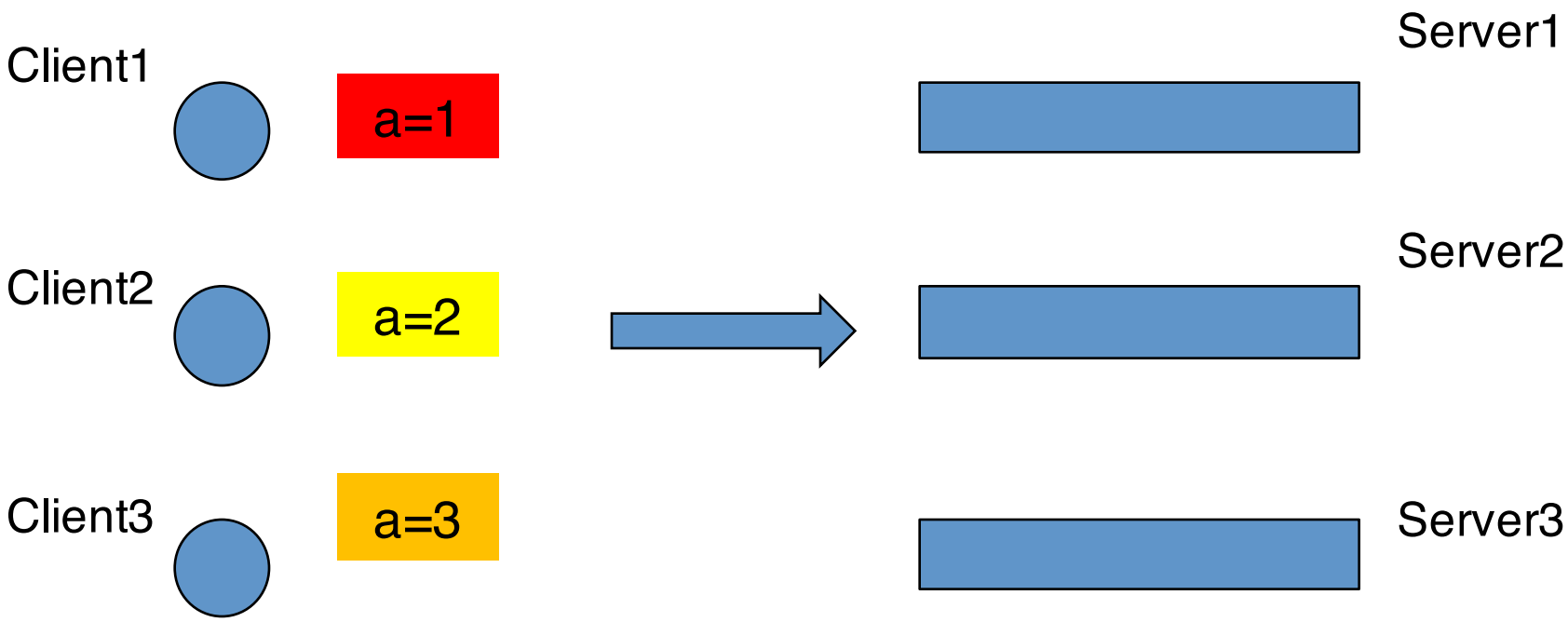
Paxos – Accept phase



A client waits for f+1 responses.

1. If any response is a “sorry”, choose a higher proposal number and restart.
2. If all are OK and all are empty, what to do?

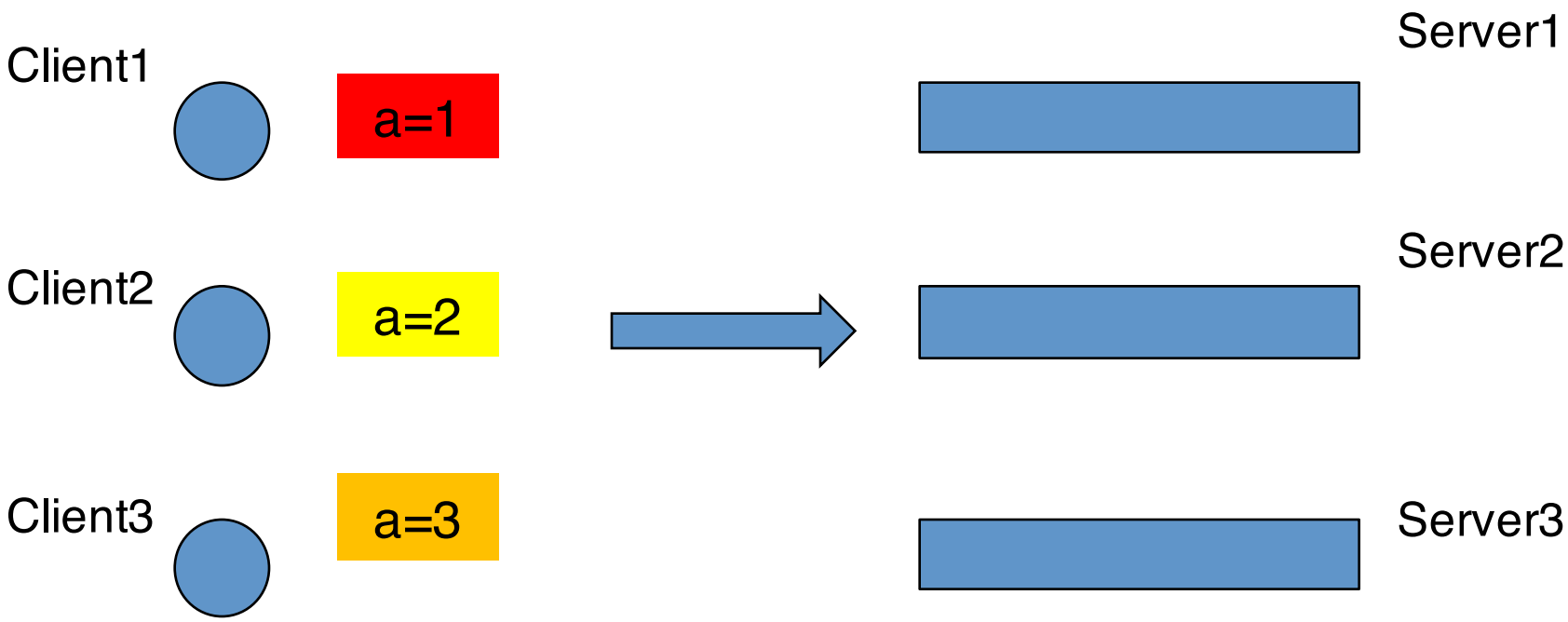
Paxos – Accept phase



A client waits for f+1 responses.

1. If any response is a “sorry”, choose a higher proposal number and restart.
2. If all are OK and all are empty, propose my own request.
3. If all are OK and some are non-empty, what to do?

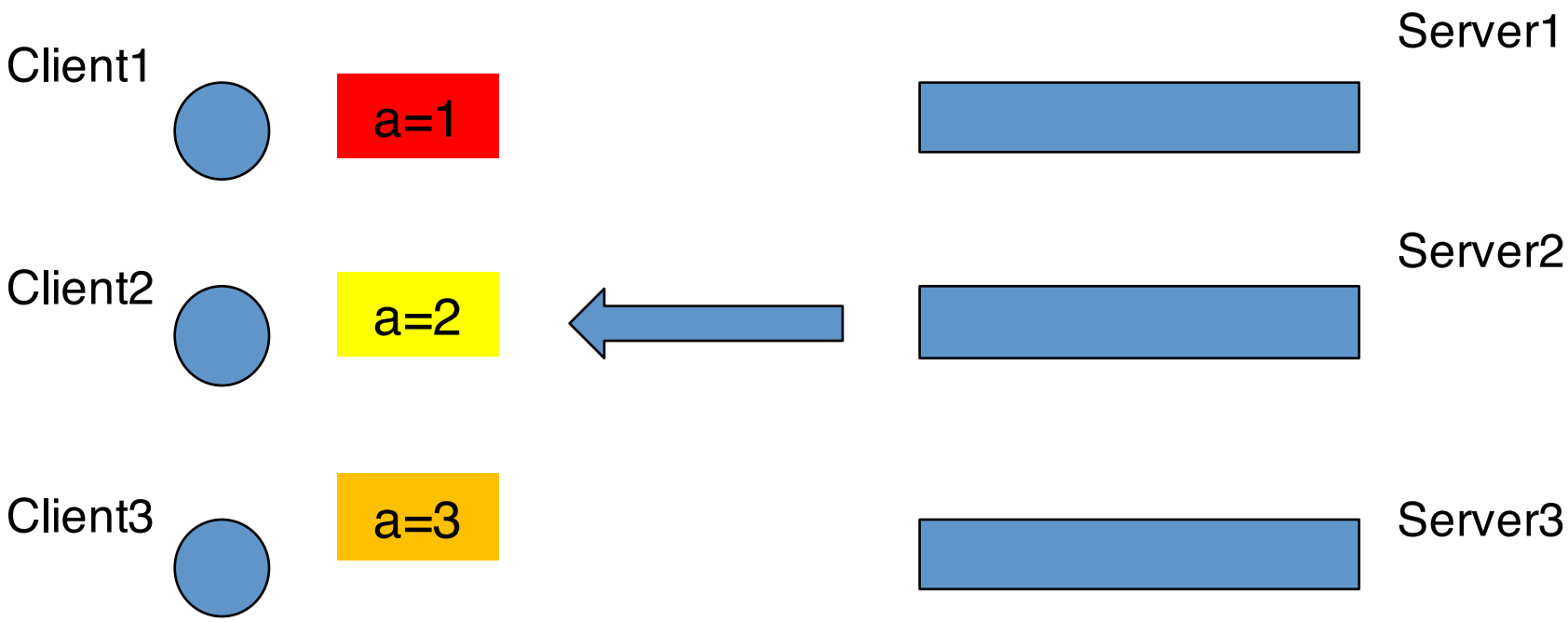
Paxos – Accept phase



A client waits for f+1 responses.

1. If any response is a “sorry”, choose a higher proposal number and restart.
2. If all are OK and all are empty, propose my own request.
3. If all are OK and some are non-empty, pick up latest one and propose it.

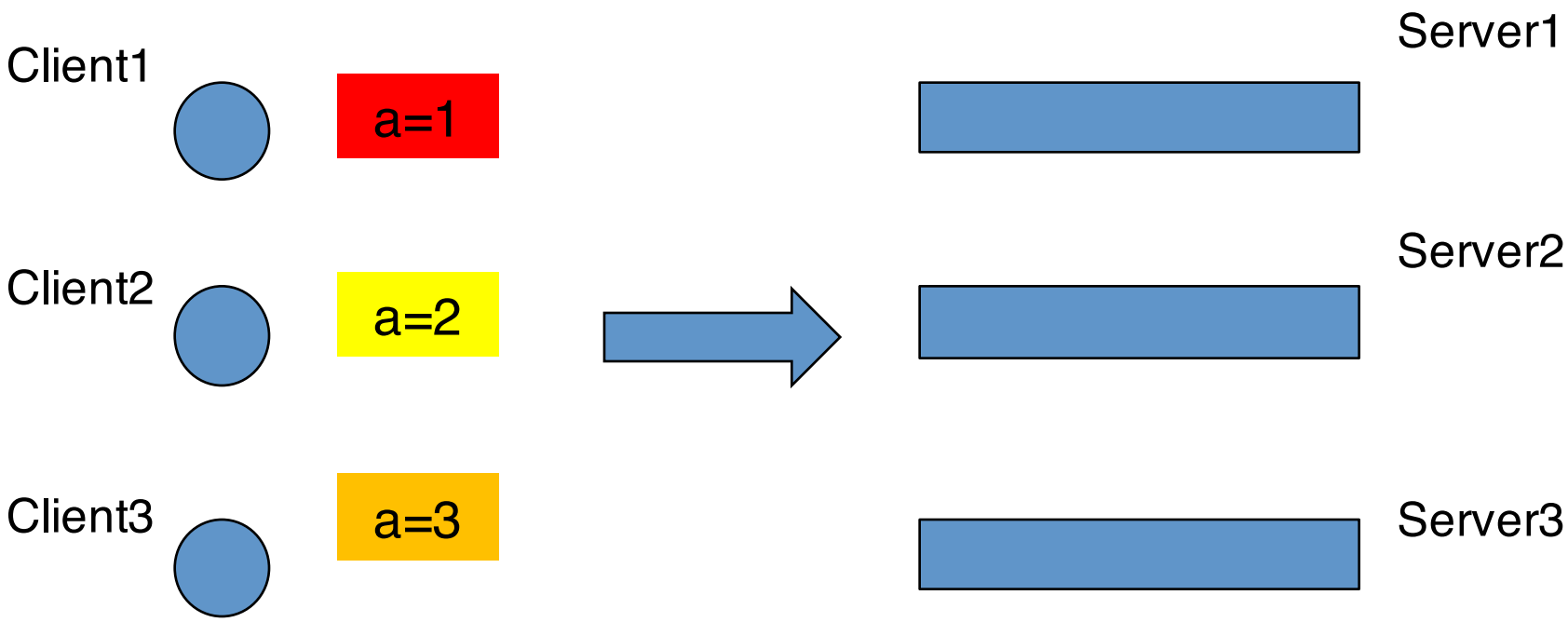
Paxos – Accept phase



When a server receives a proposal (v, n):

1) If it has promised not to accept (v, n), ignore it.
2) Otherwise, it responds “(v, n) agreed”.

Paxos - Learn phase



If a client receives f+1 “(v, n) agreed”, it sends “execute request (v, n)” to all servers.

Prove “agreement”

- Agreement: all correct replicas agree on the same value.
- Lemma 1: if a request (v, n1) is agreed by at least f+1 replicas, it is impossible for a request (v’, n2) (n2>n1 and v’<>v) to be agreed by any replica.

Prove “agreement”

- Lemma 1: if a request (v, n1) is agreed by at least f+1 replicas, it is impossible for a request (v’, n2) (n2>n1 and v’<>v) to be agreed by any replica.
  - Proof by contradiction: suppose there exists such requests and the earliest one is (v’, n2).

Prove “agreement”

- Lemma 1: if a request (v, n1) is agreed by at least f+1 replicas, it is impossible for a request (v’, n2) (n2>n1 and v’<>v) to be agreed by any replica.
  - Proof by contradiction: suppose there exists such requests and the earliest one is (v’, n2).
  - At least f+1 replicas agrees on n1 before they respond OK to the “prepare” of n2. Why?

Prove “agreement”

- Lemma 1: if a request (v, n1) is agreed by at least f+1 replicas, it is impossible for a request (v’, n2) (n2>n1 and v’<>v) to be agreed by any replica.
  - Proof by contradiction: suppose there exists such requests and the earliest one is (v’, n2).
  - At least f+1 replicas agrees on n1 before they respond OK to the “prepare” of n2.
  - v' must be either the same as v or the same as some request between n1 and n2.

Prove “agreement”

- Lemma 1: if a request (v, n1) is agreed by at least f+1 replicas, it is impossible for a request (v’, n2) (n2>n1 and v’<>v) to be agreed by any replica.
  - Proof by contradiction: suppose there exists such requests and the earliest one is (v’, n2).
  - At least f+1 replicas agrees on n1 before they respond OK to the “prepare” of n2.
  - v' must be either the same as v or the same as some request between n1 and n2.
  - This contradicts with our assumption that v’ is the earliest one that is different from v.

Prove “agreement”

- Agreement: all correct replicas agree on the same value.
- Suppose one replica executes request (v, n1) and one executes (v’, n2)

Prove “agreement”

- Agreement: all correct replicas agree on the same value.
- Suppose one replica executes request (v, n1) and one executes (v’, n2)
  - If n1==n2, then obviously they are the same.

Prove “agreement”

- Agreement: all correct replicas agree on the same value.
- Suppose one replica executes request (v, n1) and one executes (v’, n2)
  - If n1==n2, then obviously they are the same.
  - If n1 < n2, it means both n1 and n2 must be agreed by at least f+1 replicas
    - By using Lemma 1, v must be the same as v’.

Termination

- What can prevent termination?

Termination

- What can prevent termination?
  - If a client starts a new round of proposal before an earlier proposal completes.

Termination

- What can prevent termination?
  - If a client starts a new round of proposal before an earlier proposal completes.
- A client should sleep a while (random and exponentially growing) before re-proposing

Termination

- What can prevent termination?
  - If a client starts a new round of proposal before an earlier proposal completes.
- A client should sleep a while (random by exponentially growing) before re-proposing
- Can achieve termination when
  - Clock of the client is reasonably accurate
  - Max network latency < min difference of sleep time

Exercise: true or false

A replica can agree on at most one request.

Exercise: true or false

A replica can agree on at most one request.
False: a later proposal may overwrite an earlier one.

Exercise: true or false

During the “prepare” phase, if f+1 servers respond “empty”, then no proposals have been made so far.

Exercise: true or false

During the “prepare” phase, if f+1 servers respond “empty”, then no proposals have been made so far.
False: a proposal may have been made to the remaining f servers.

Exercise: true or false

During the “prepare” phase, if some servers respond with existing proposals, we pick up the latest one because it must have been agreed by at least f+1 replicas.

Exercise: true or false

During the “prepare” phase, if some servers respond with existing proposals, we pick up the latest one because it must have been agreed by at least f+1 replicas.
False: it may or may not be agreed by f+1, but for safety, we can only pick up the latest one.

Exercise: true or false

During the “prepare” phase, if some servers respond with existing proposals, the ones except the latest one must have not been agreed by f+1 replicas.

Exercise: true or false

During the “prepare” phase, if some servers respond with existing proposals, the ones except the latest one must have not been agreed by f+1 replicas.
True: prove by contradiction. This is why we can give up earlier ones and pick up the latest one.

Exercise: true or false

If a client decides to propose a new value after “prepare”, then no earlier proposals will be agreed by any replica.

Exercise: true or false

If a client decides to propose a new value after “prepare”, then no earlier proposals will be agreed by any replica.
False: an earlier proposal may still be agreed by up to f replicas, but that is OK.

Exercise: true or false

If a client decides to propose a new value after “prepare”, then its proposal will eventually get through as long as the client does not fail.

Exercise: true or false

If a client decides to propose a new value after “prepare”, then its proposal will eventually get through as long as the client does not fail.
False: another client can start a new “prepare” phase and prevents servers from accepting the proposal

Exercise: true or false

If a request is agreed by at least f+1 replicas, then no client will propose new request.

Exercise: true or false

If a request is agreed by at least f+1 replicas, then no client will propose new request.
True. This is called “stable” or “chosen” request.

Exercise

- If we remove the “promise” part in the prepare phase and add a simple rule in the propose phase: if a server has agreed on n1, it will not agree on n2<n1. Does this work?