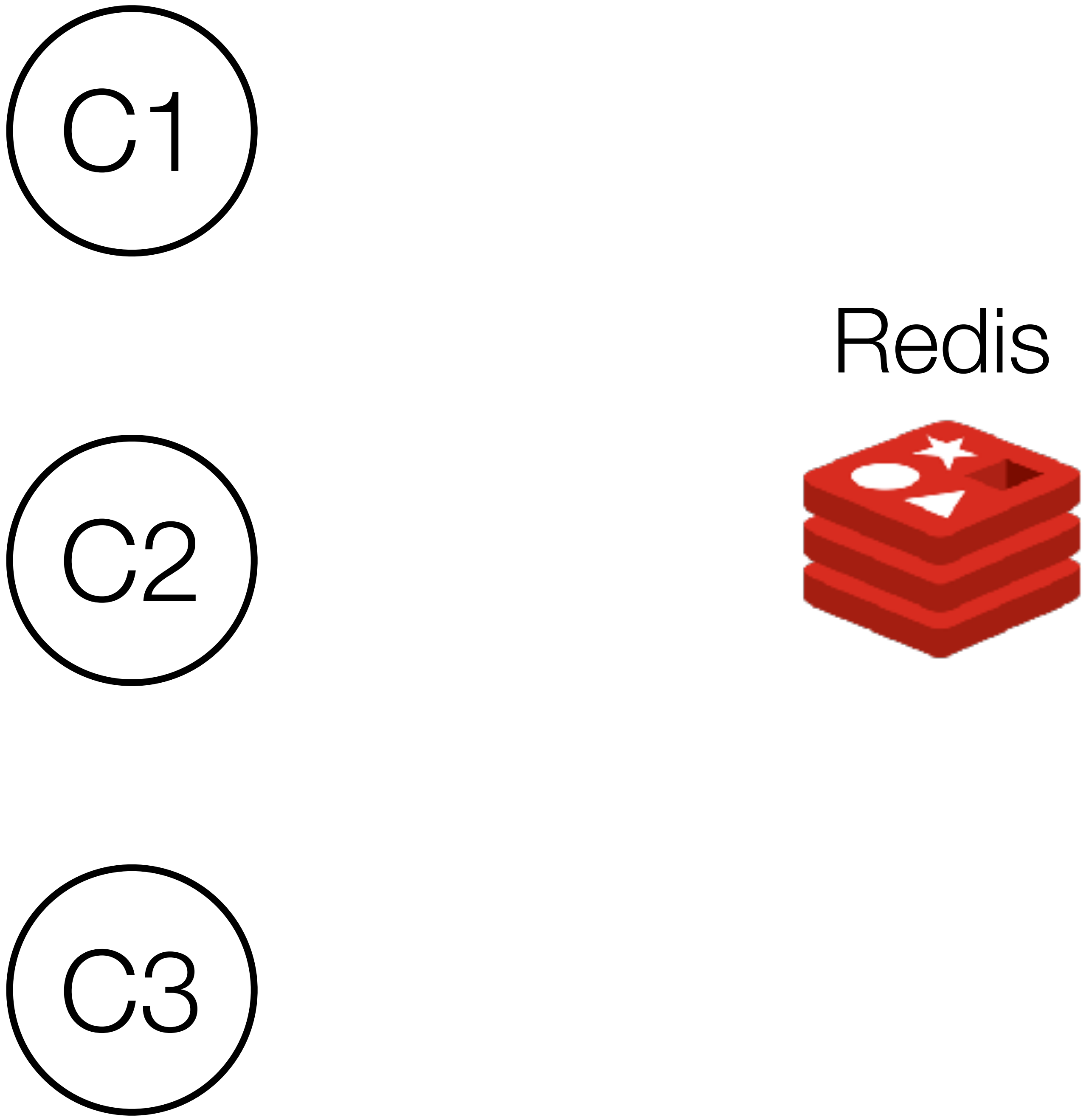
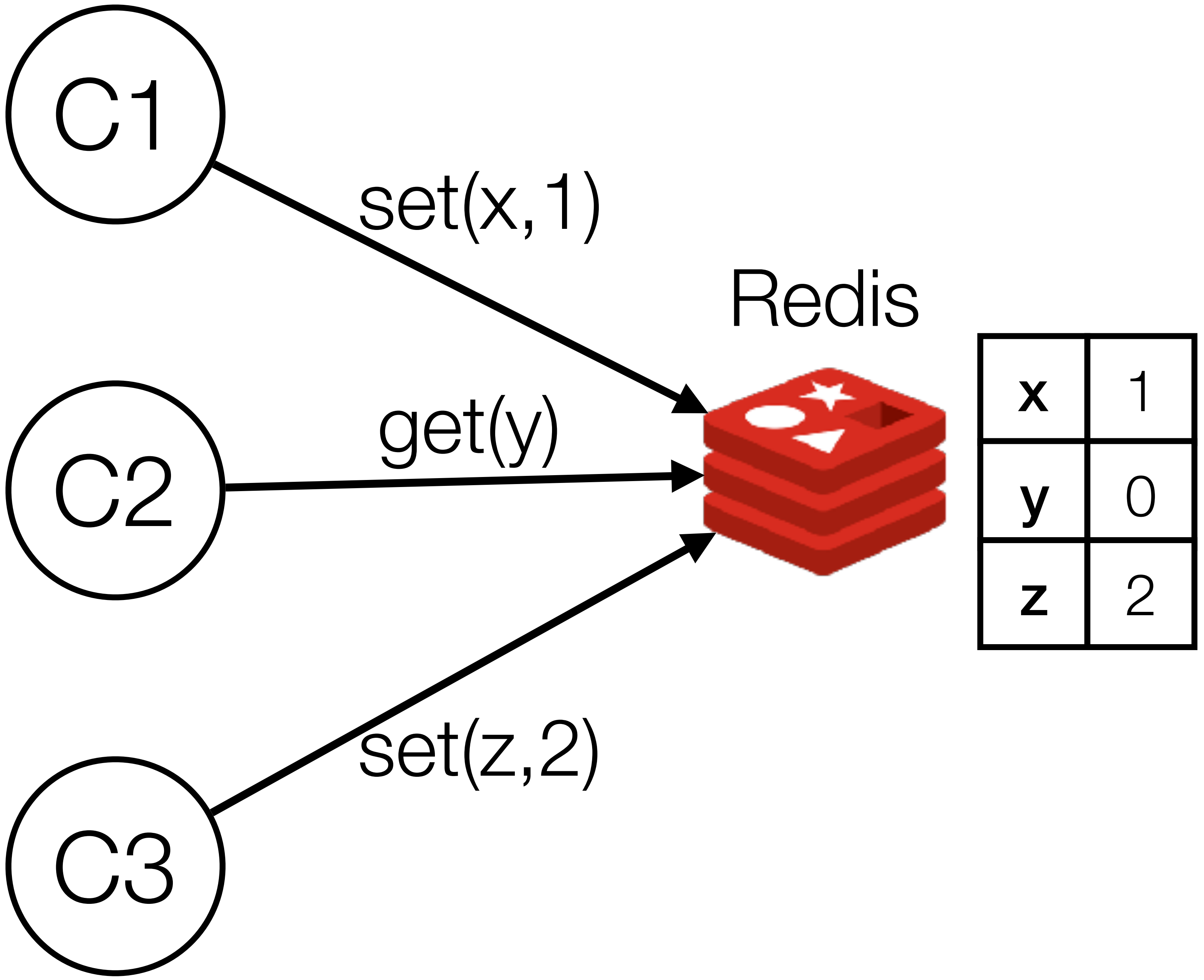
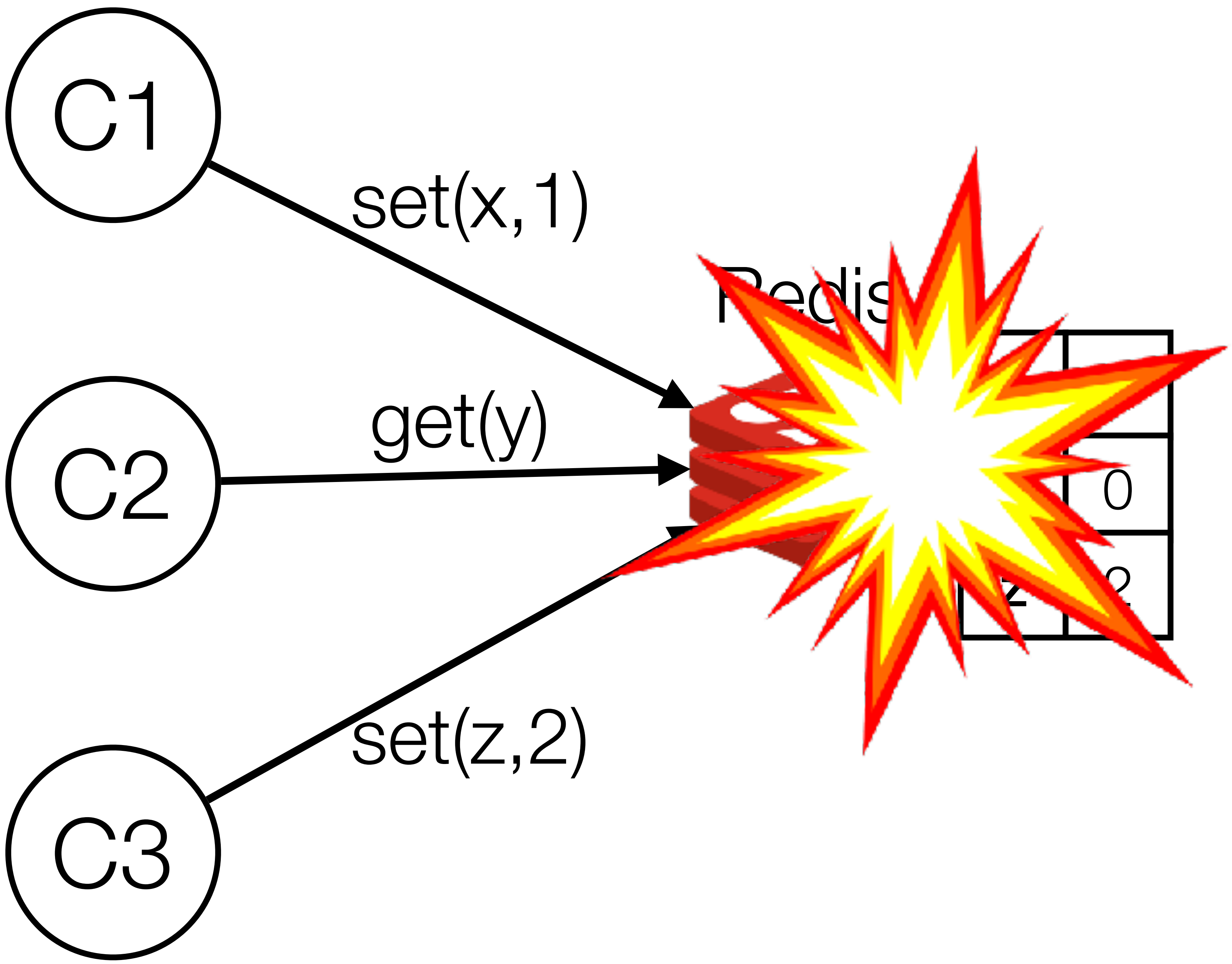
A High Throughput Paxos Variant That Mere Mortals Can Understand

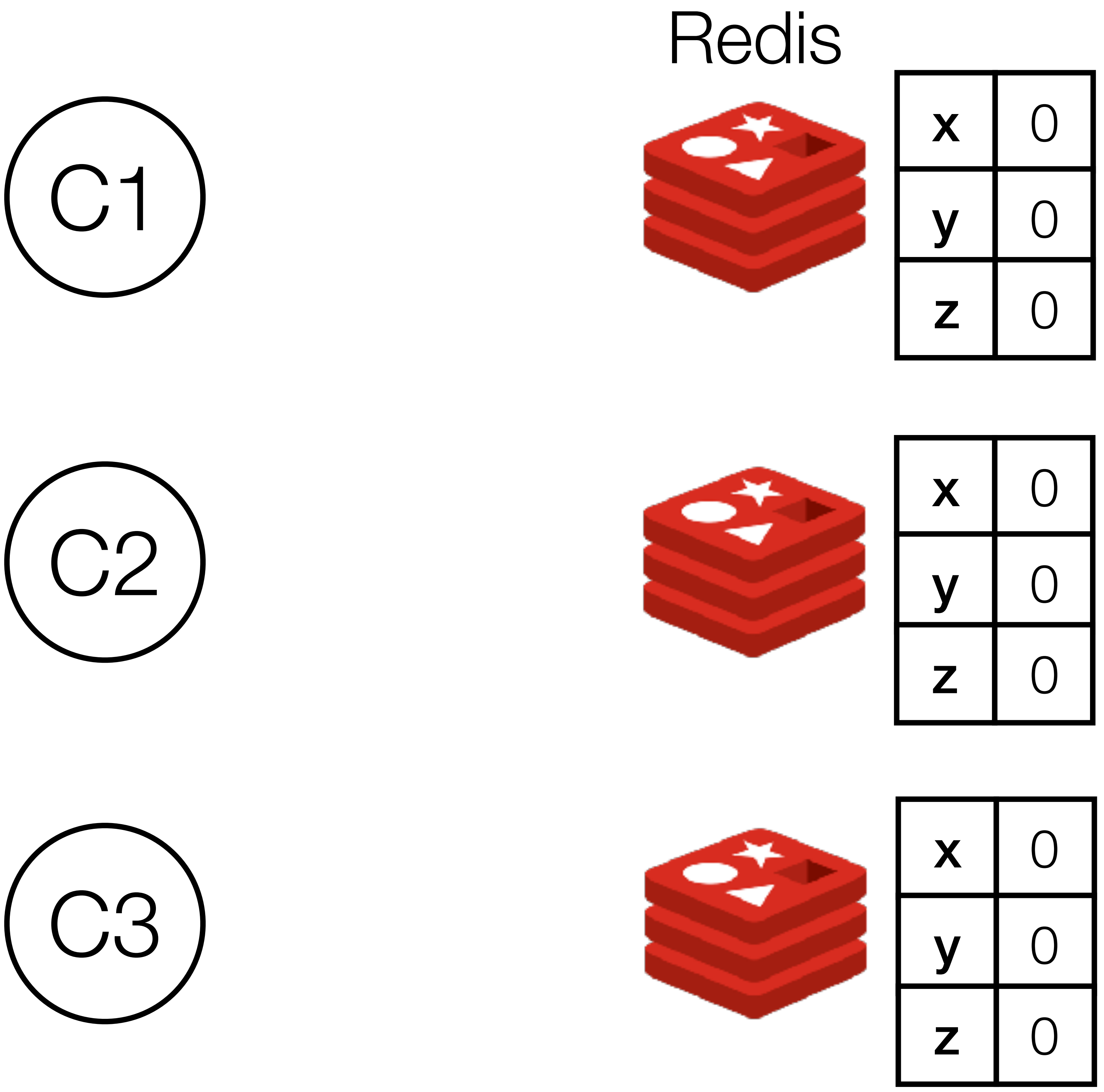
Neil Giridharan, Joseph M. Hellerstein, Ion Stoica,
Adriana Szekeres, **Michael Whittaker**

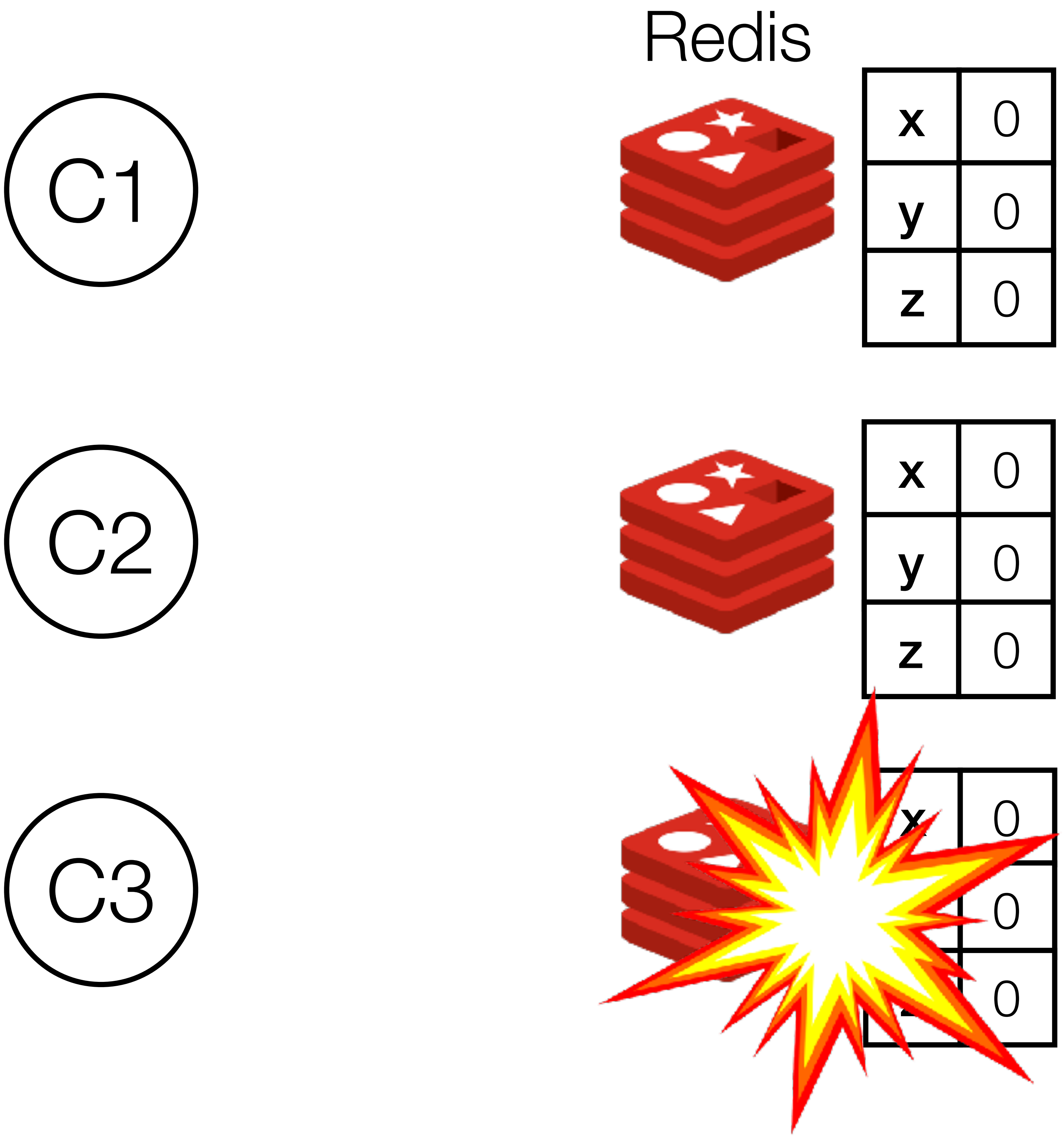

C1
Redis
C2
C3

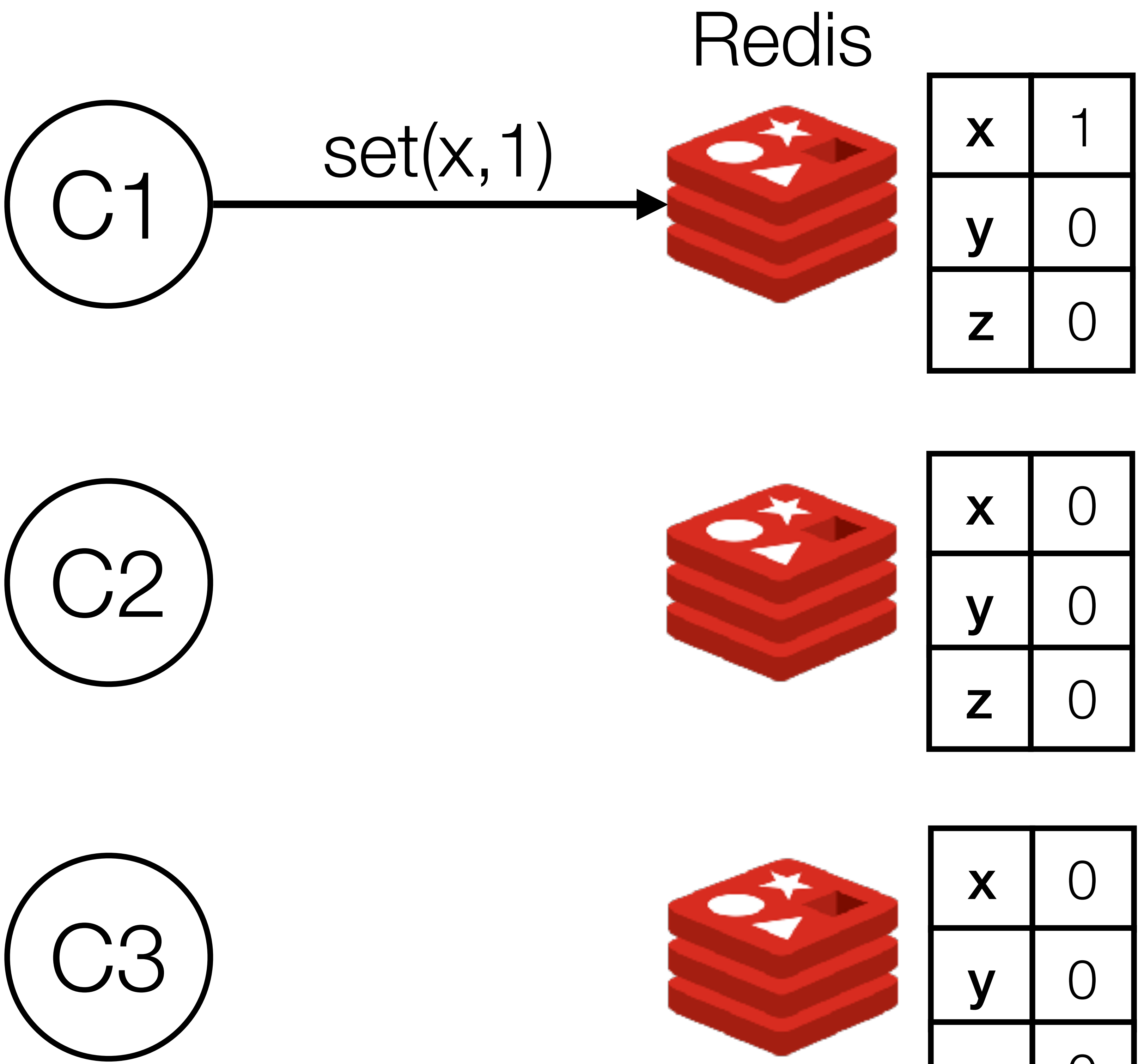


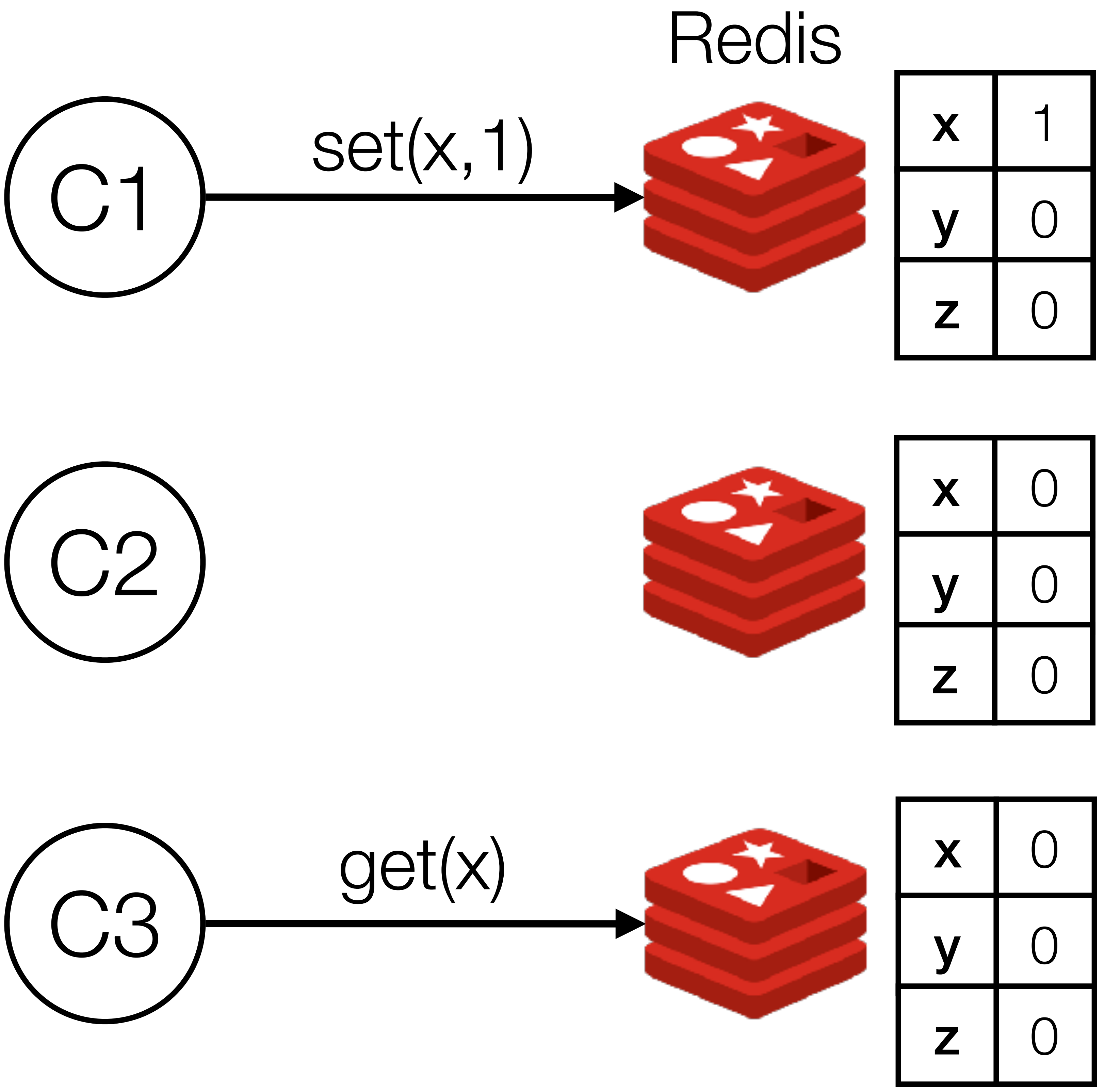
x	1
y	0
z	2

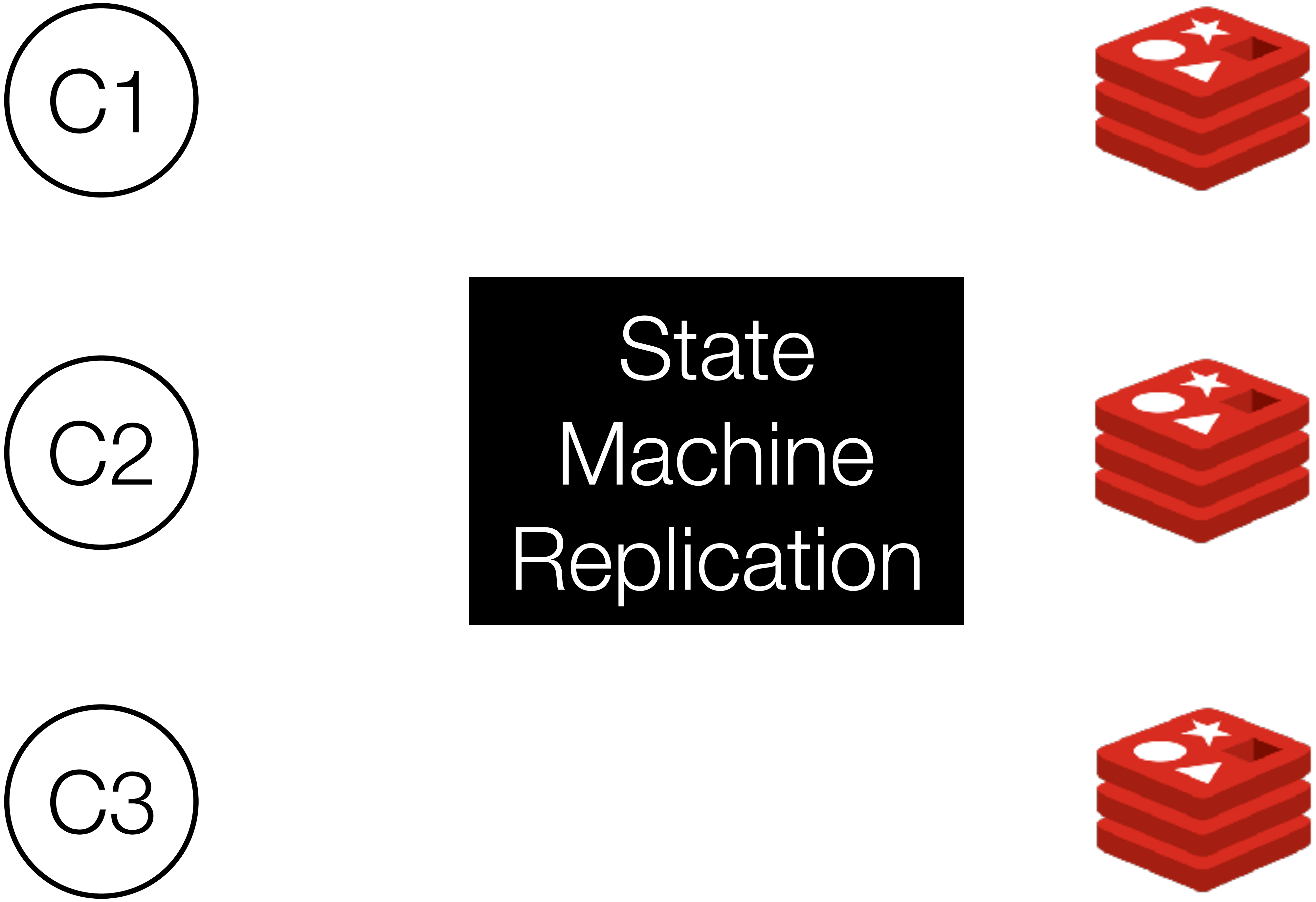

C1
C2
C3
set(x,1)
get(y)
set(z,2)
Redis
0
z
2

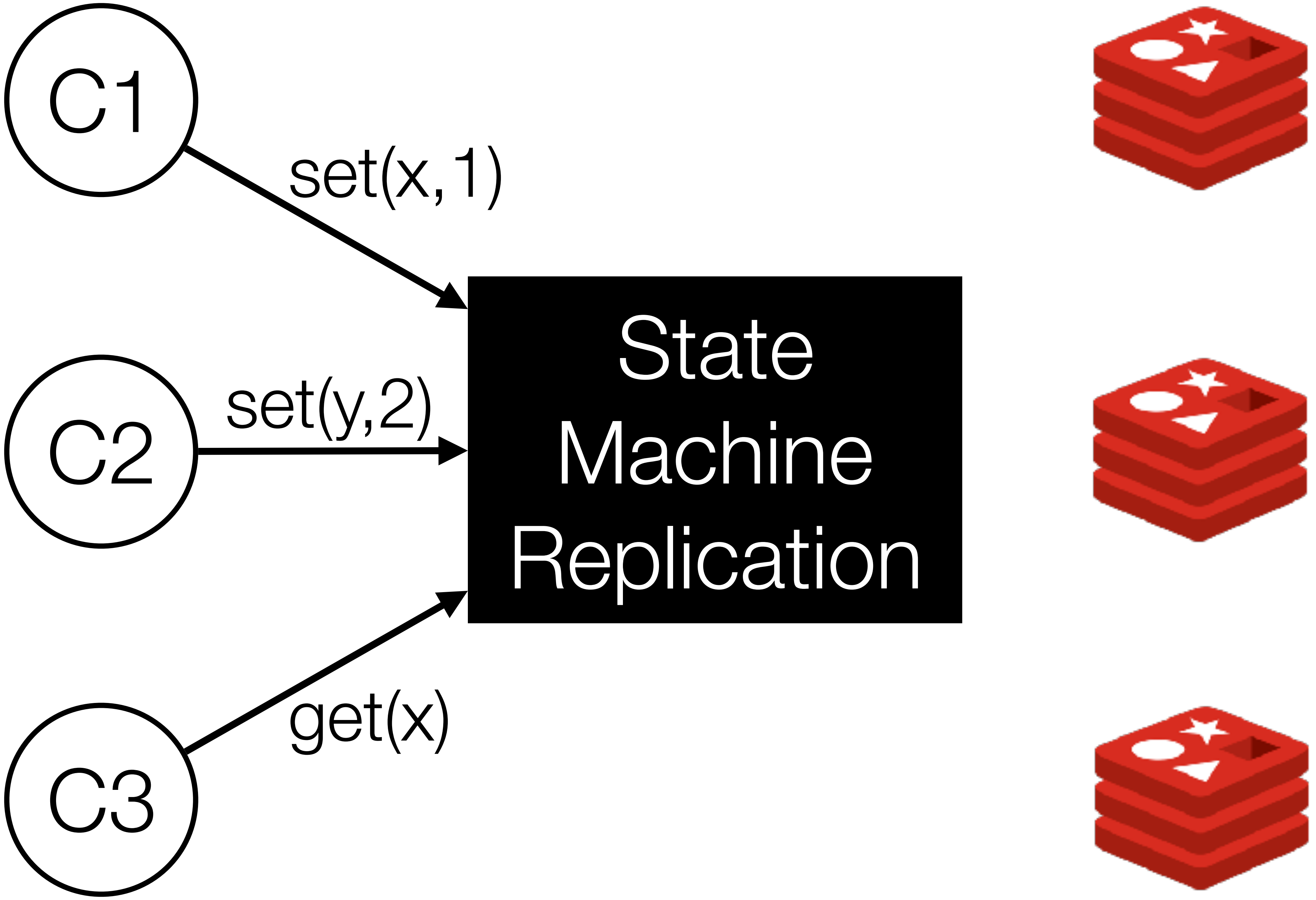

Redis
C1
x 0
y 0
z 0
C2
x 0
y 0
z 0
C3
x 0
y 0
z 0

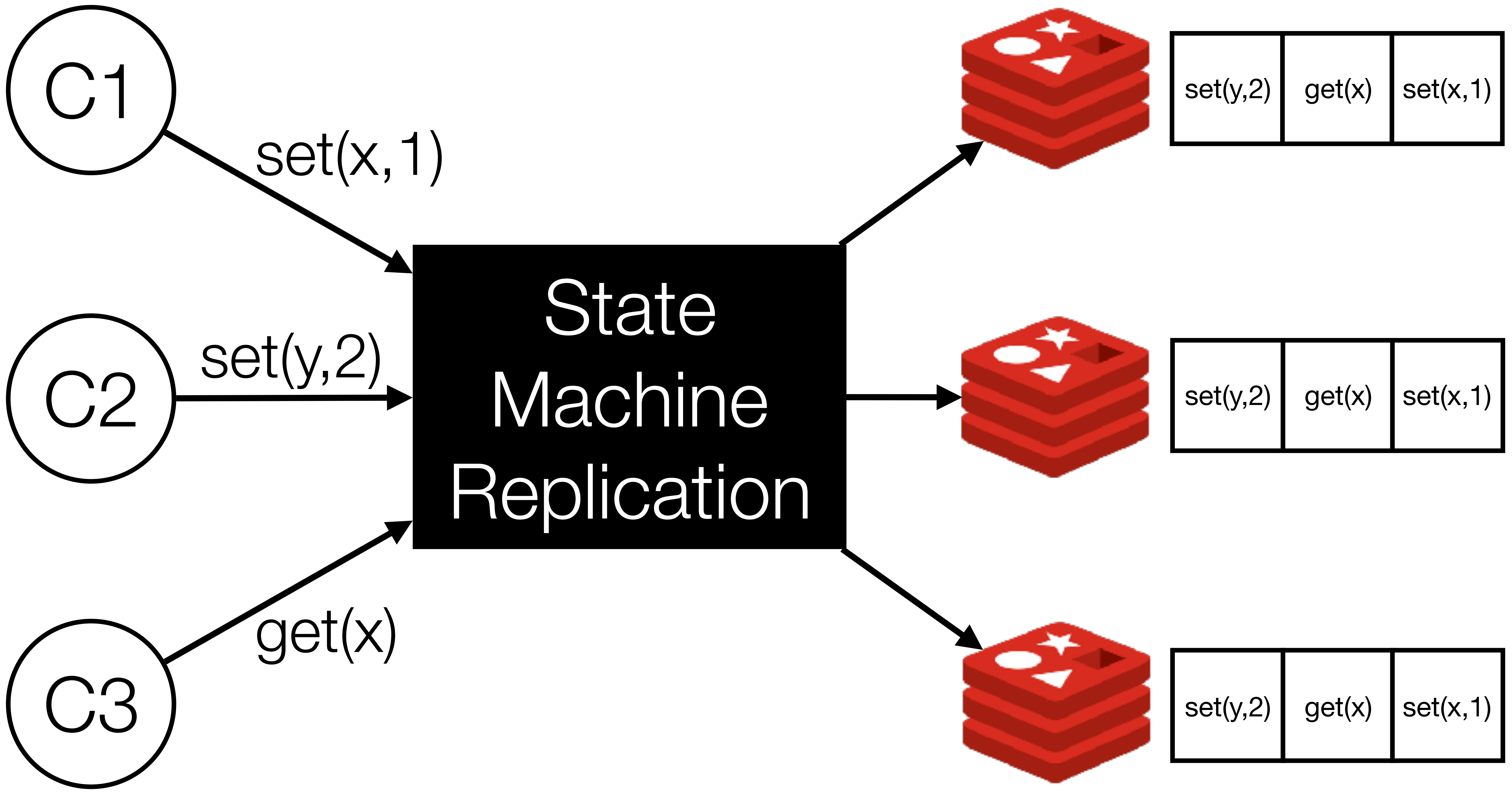

Redis
C1
C2
C3
x 0
y 0
z 0
x 0
y 0
z 0
x 0
0
0


Redis
C1
set(x,1)
C2
C3
x 1
y 0
z 0
x 0
y 0
z 0
x 0
y 0
z 0


Redis
C1
set(x,1)
x 1
y 0
z 0
C2
x 0
y 0
z 0
C3
get(x)
x 0
y 0
z 0

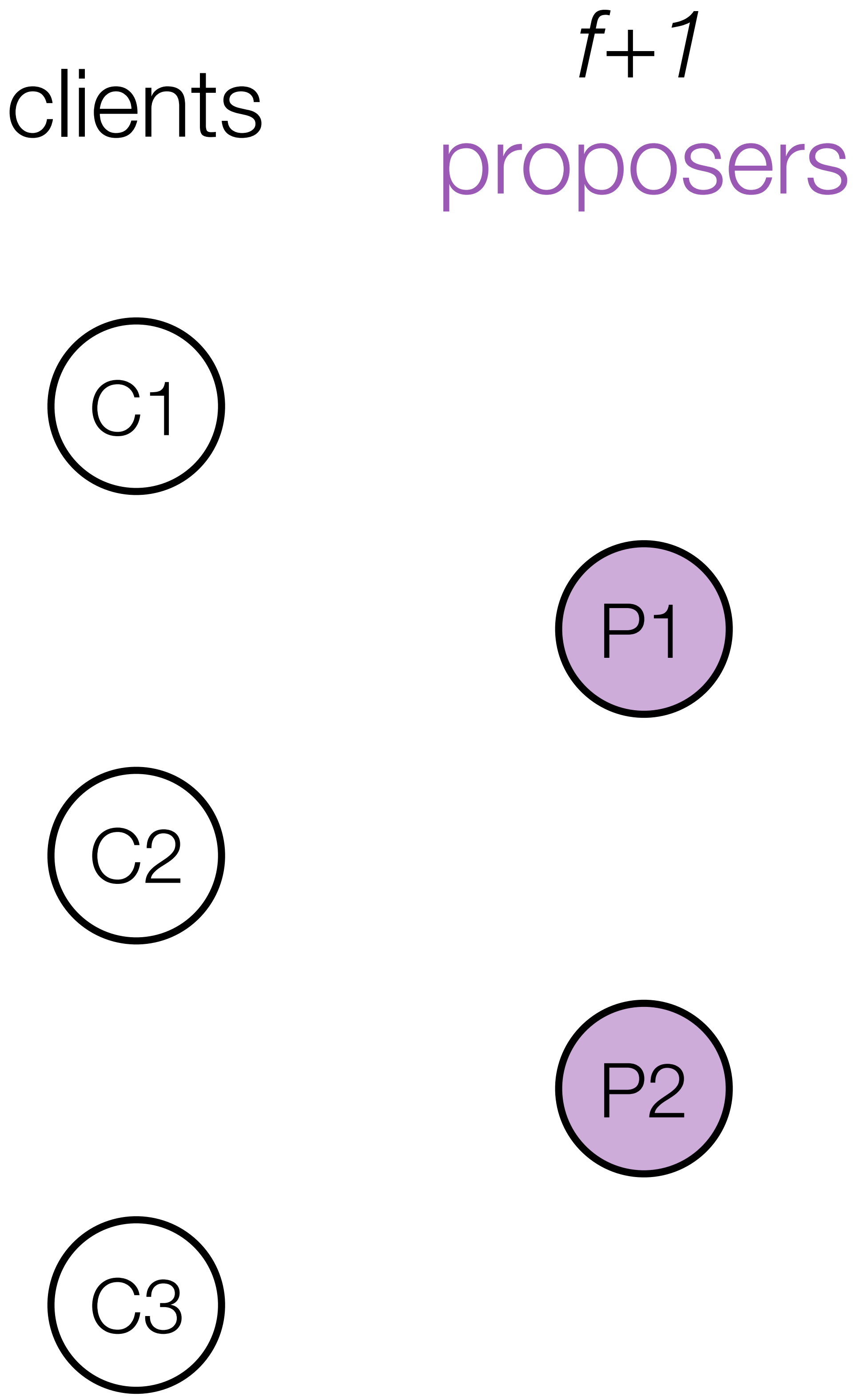

C1
C2
C3
State Machine Replication

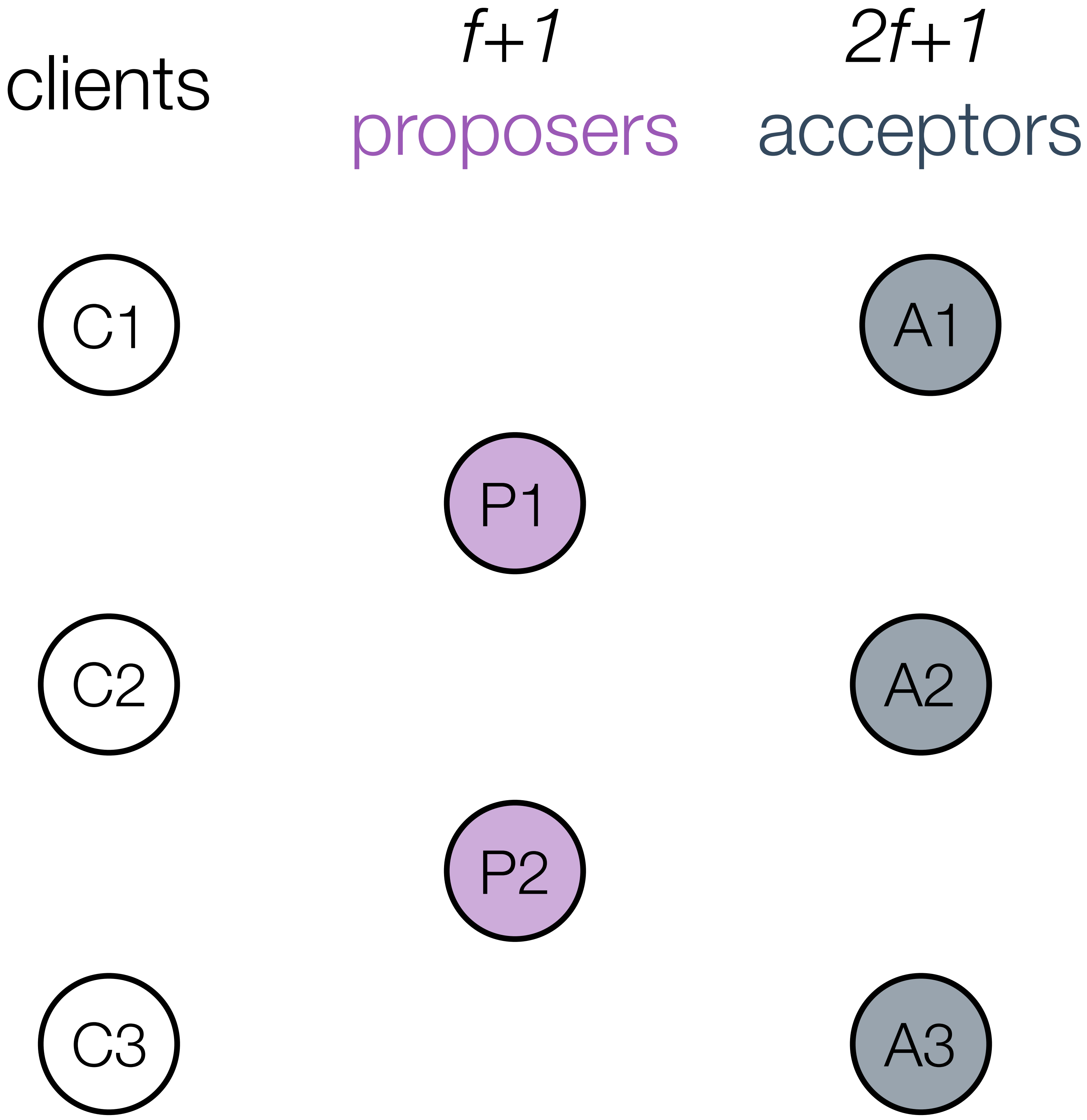

C1
set(x,1)
C2
set(y,2)
C3
get(x)
State Machine Replication

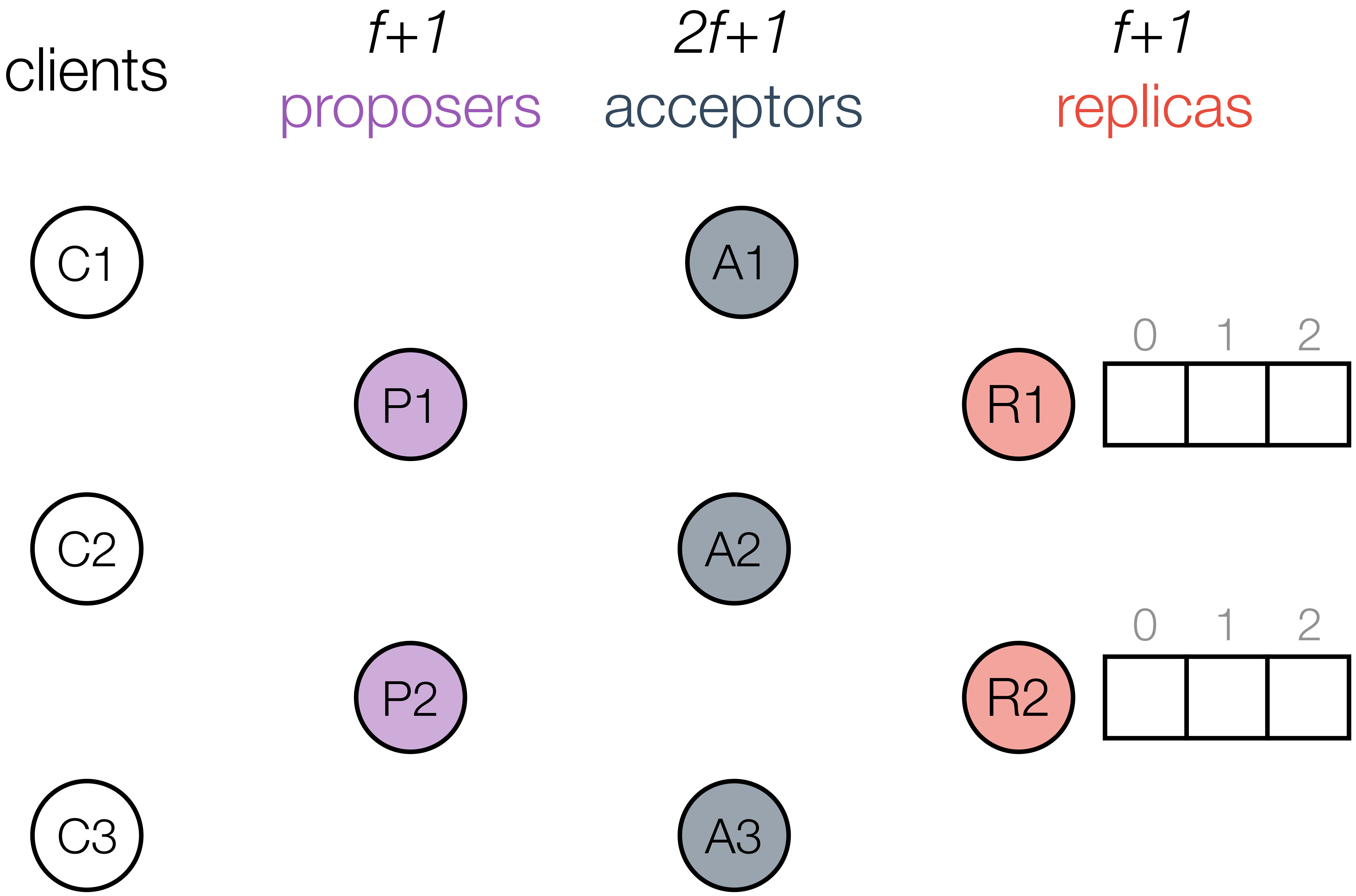

C1
C2
C3
set(x,1)
set(y,2)
get(x)
State Machine Replication
set(y,2)
get(x)
set(x,1)
set(y,2)
get(x)
set(x,1)
set(y,2)
get(x)
set(x,1)

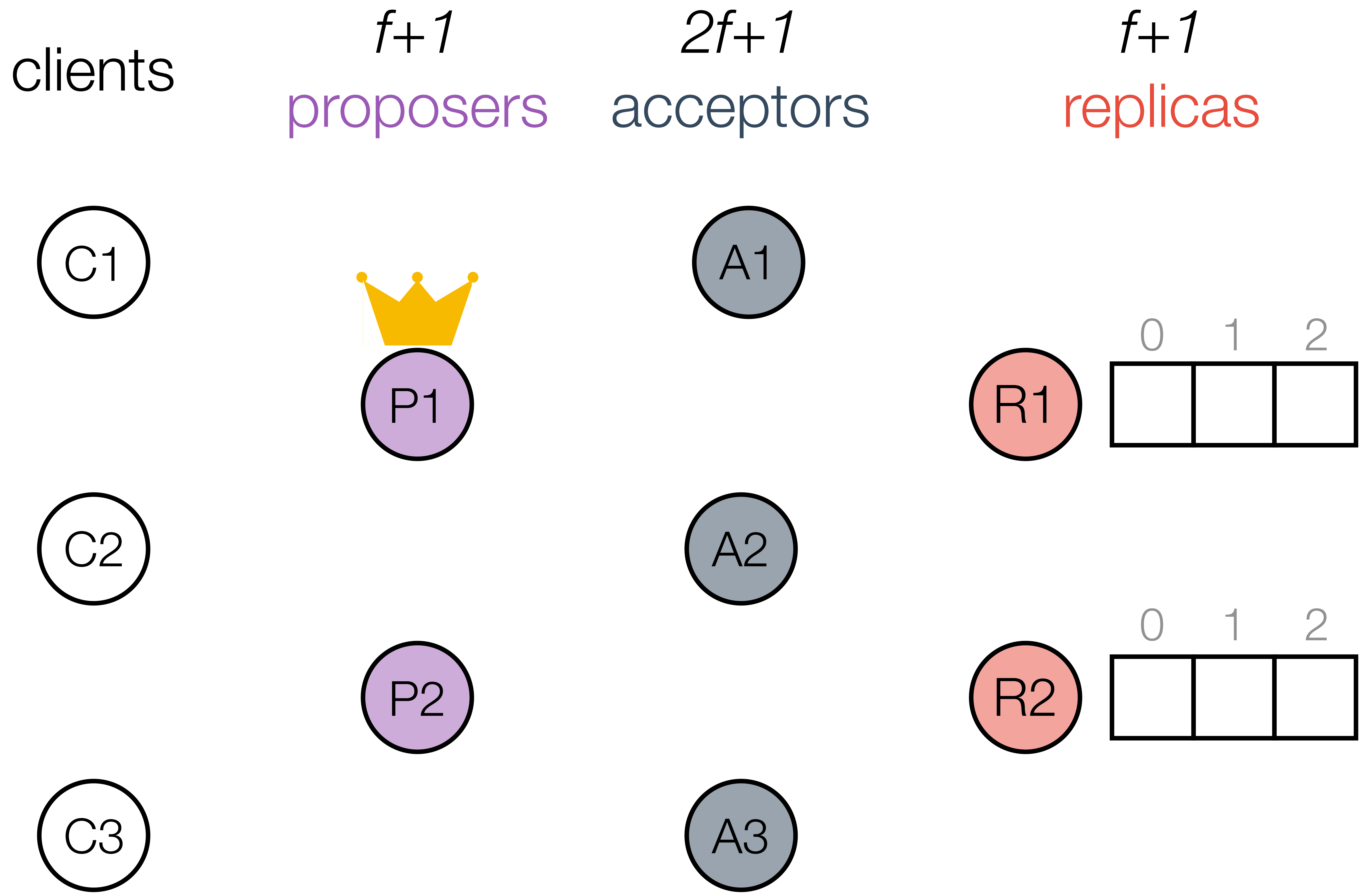
MultiPaxos

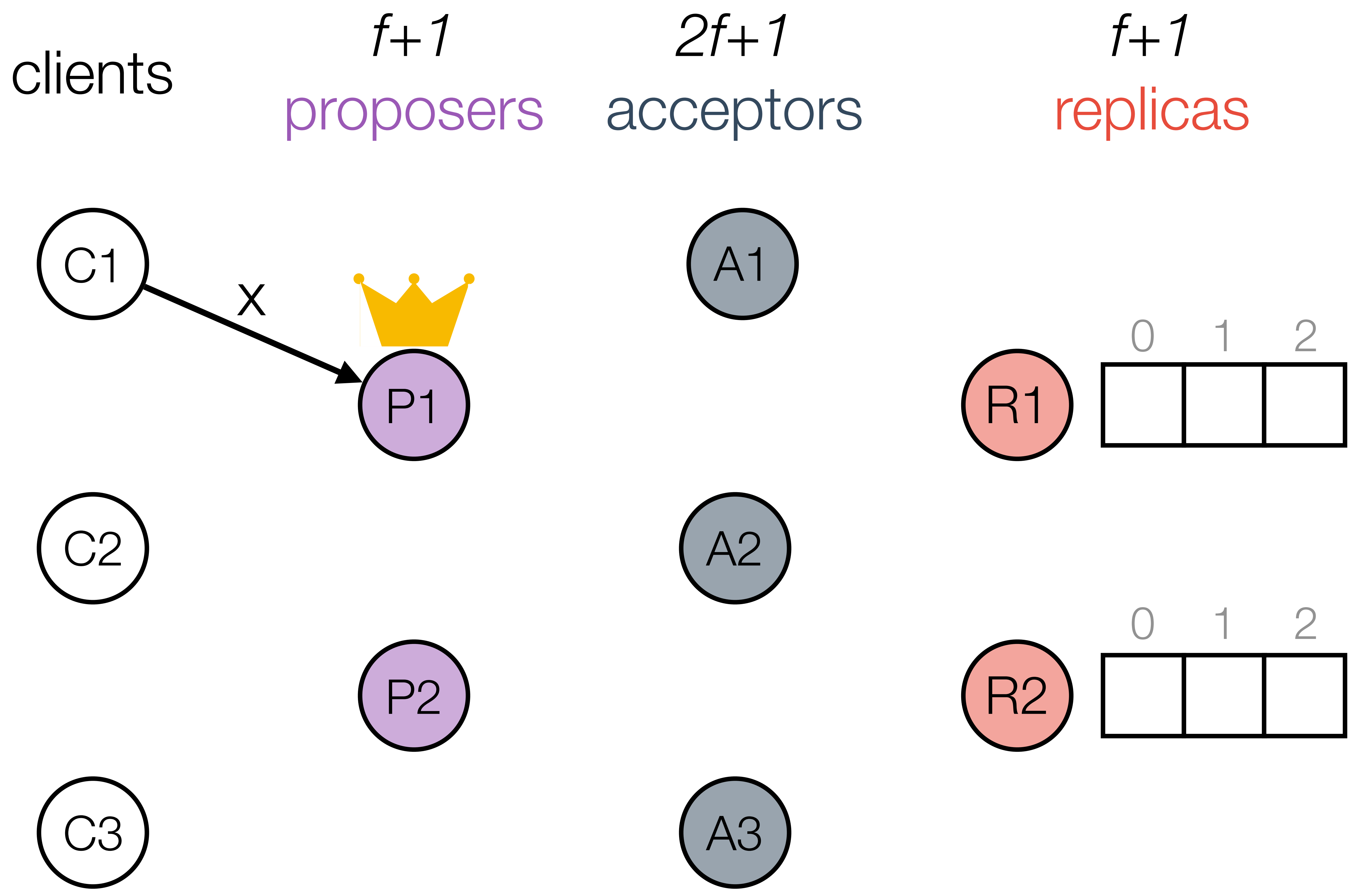
clients

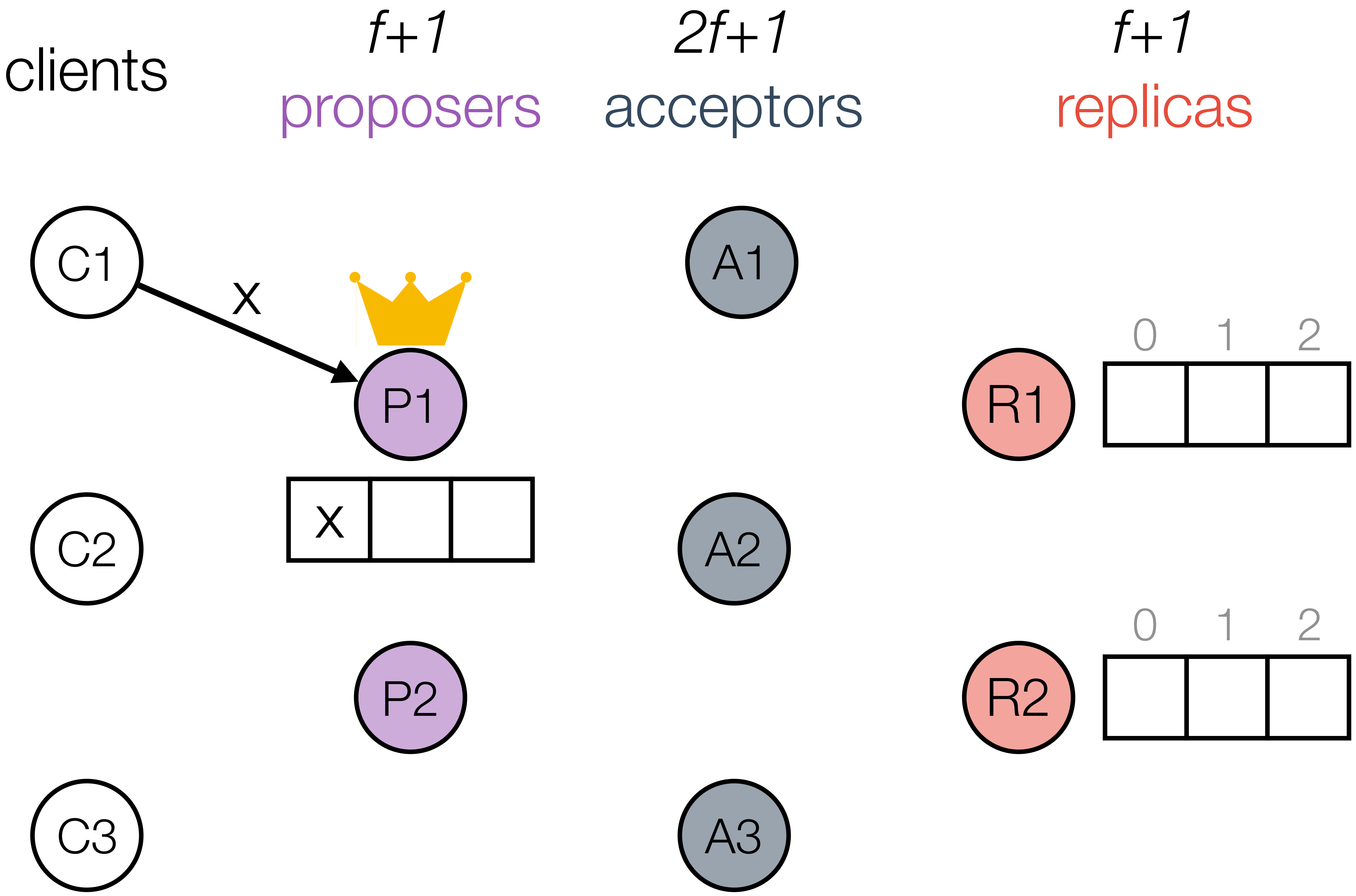

clients
f+1
proposers
C1
C2
C3
P1
P2

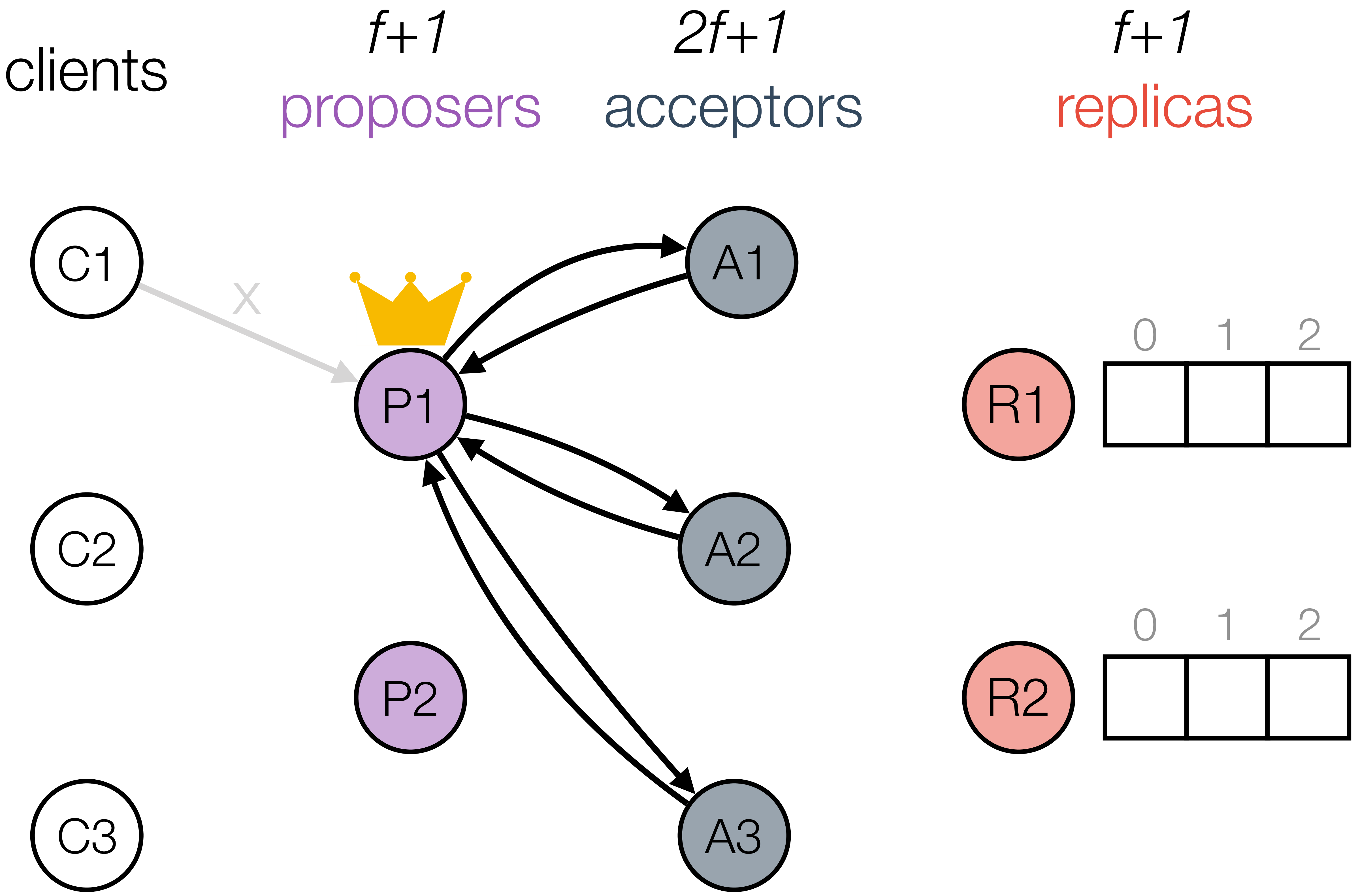

clients
f+1
proposers
2*f*+1
acceptors
C1
C2
C3
P1
P2
A1
A2
A3


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
R2
0
1
2
0
1
2

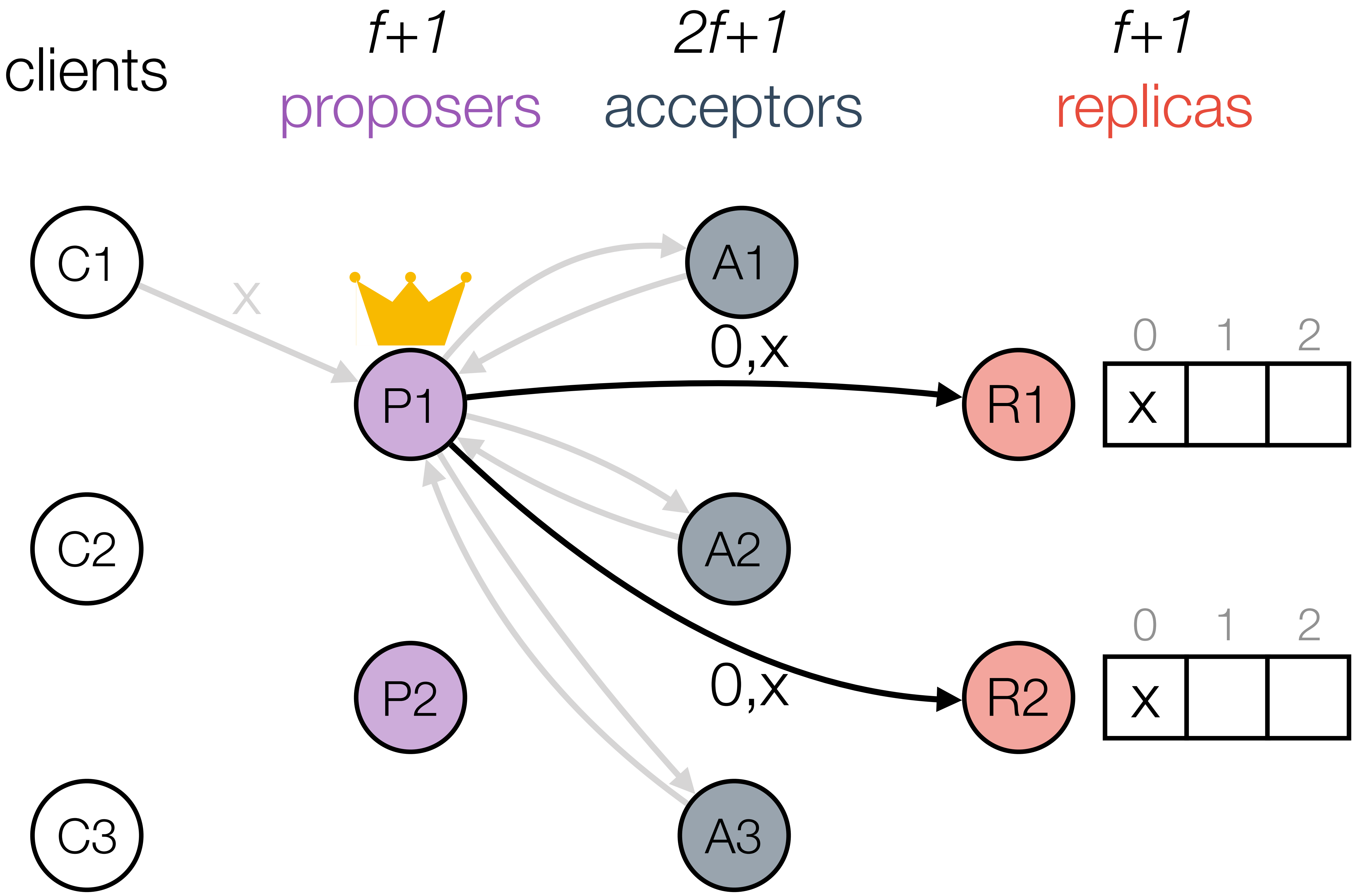

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
R2
0 1 2
0 1 2


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
P1
A1
0 1 2
R1
C2
A2
P2
0 1 2
R2
C3
A3

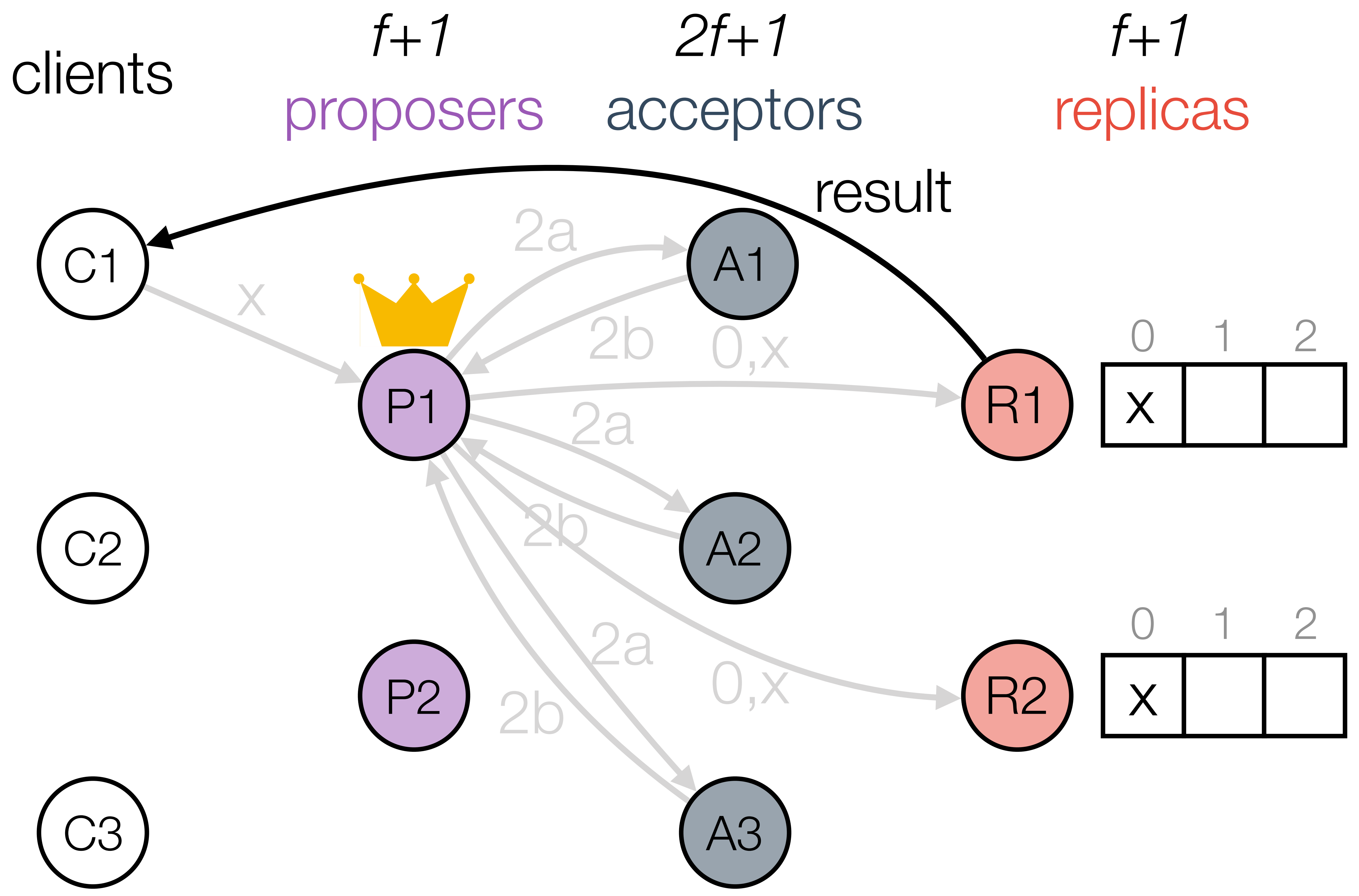

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
P1
x
C2
P2
C3
A1
A2
A3
R1
0 1 2
R2
0 1 2

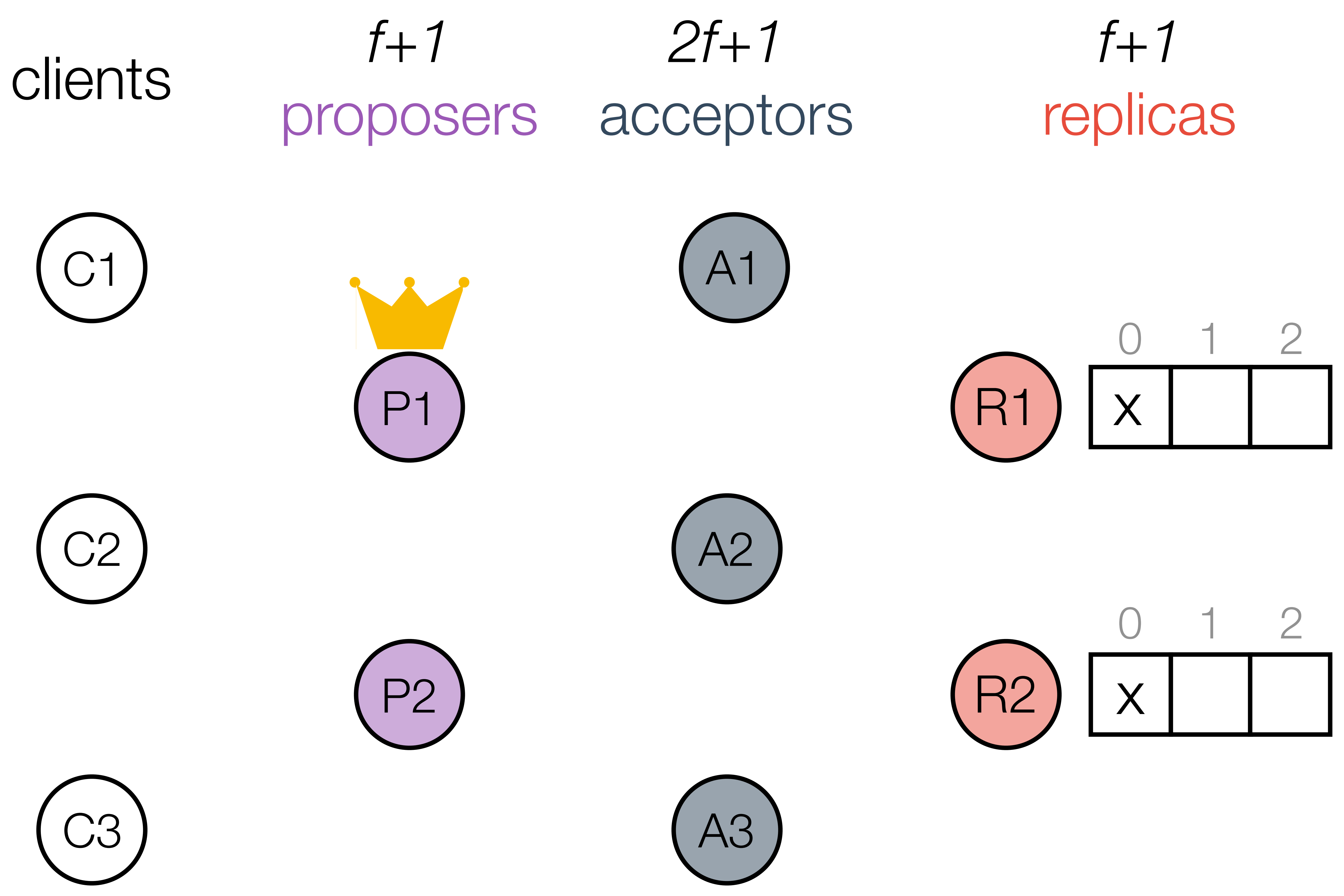


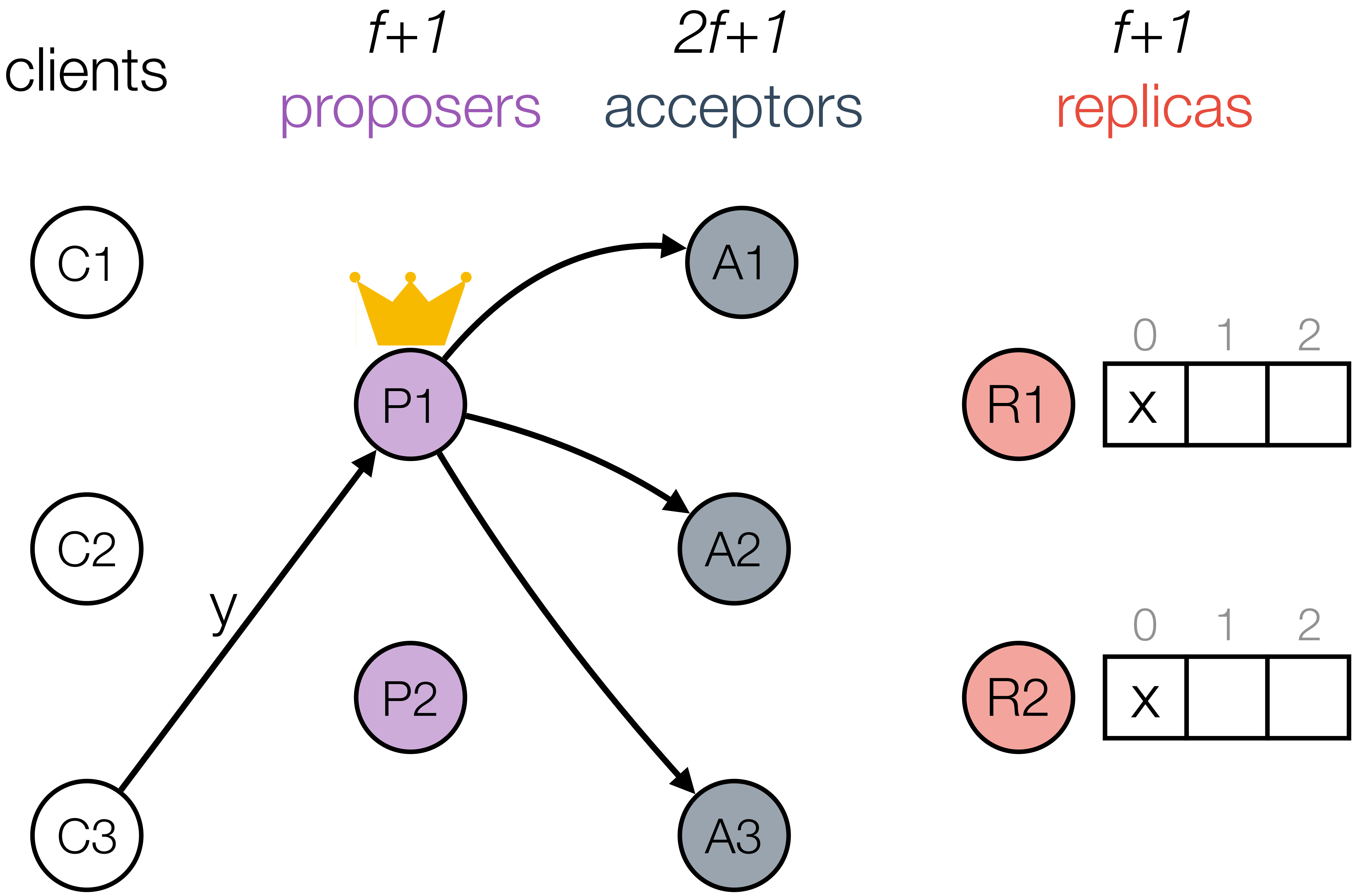
clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
A1
A2
A3
R1
R2
0 1 2
0 1 2

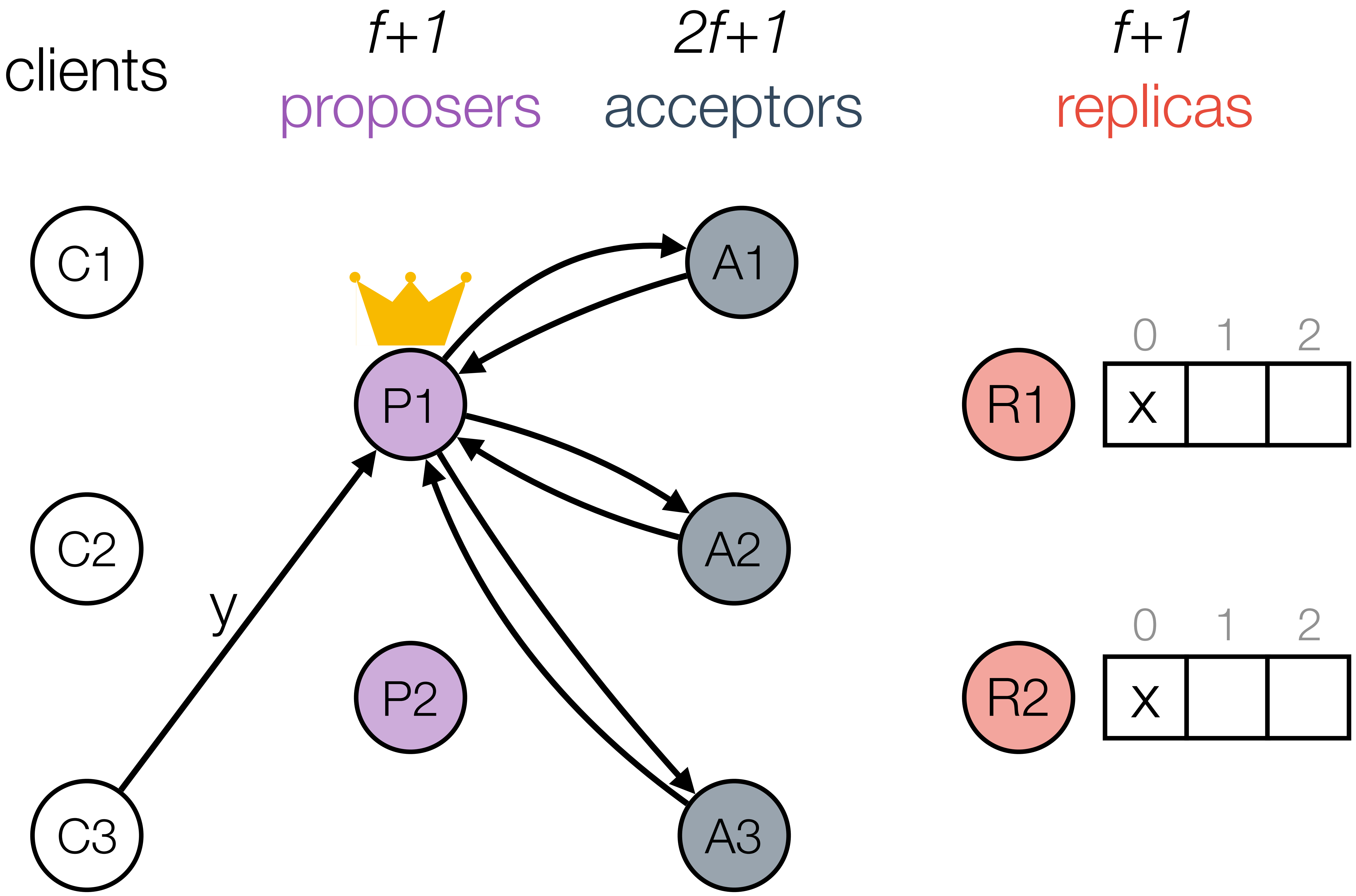


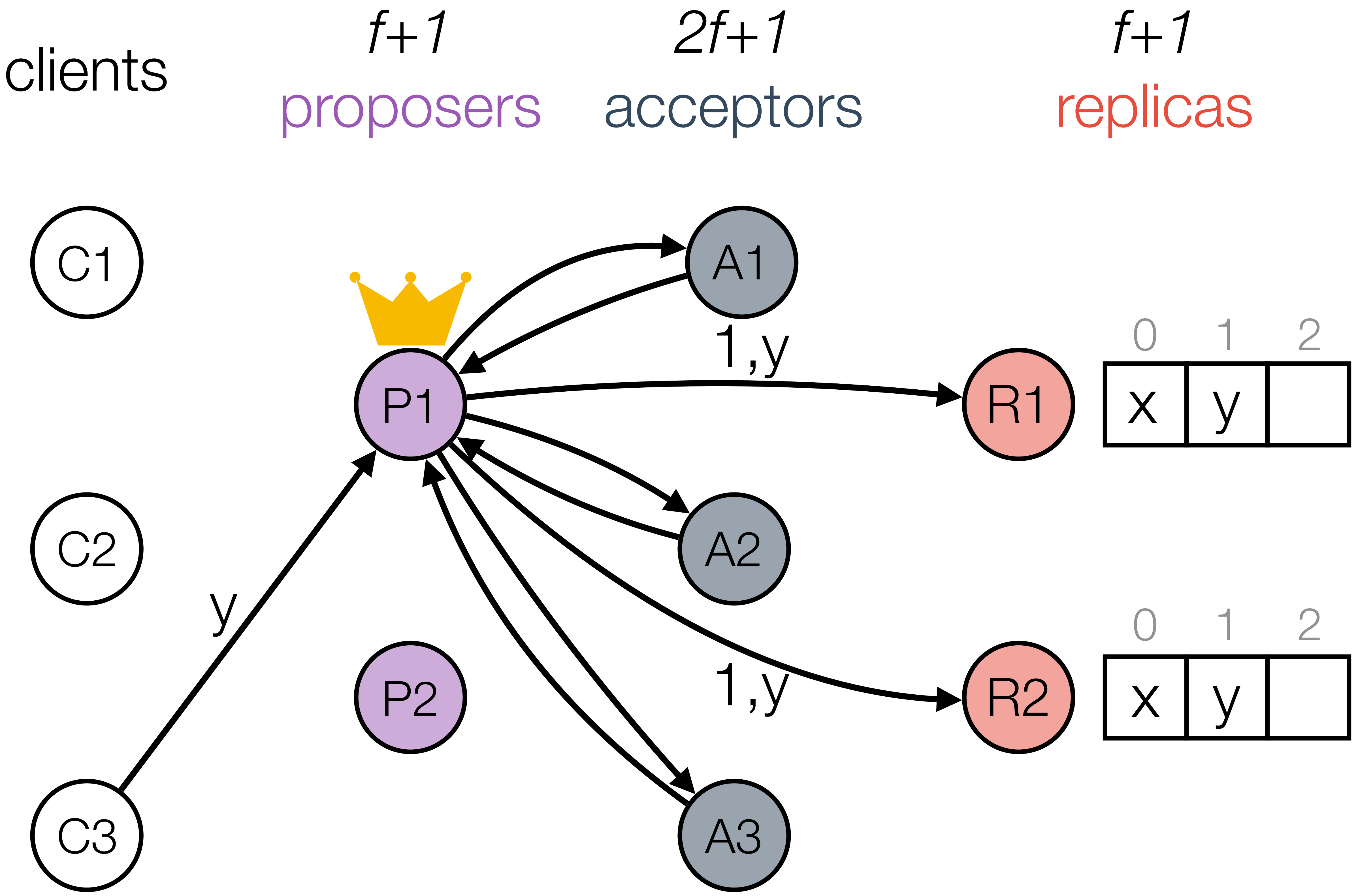
clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
C2
C3
P1
P2
A1
A2
A3
0,x
0,x
R1
R2
0 1 2
x
0 1 2
x

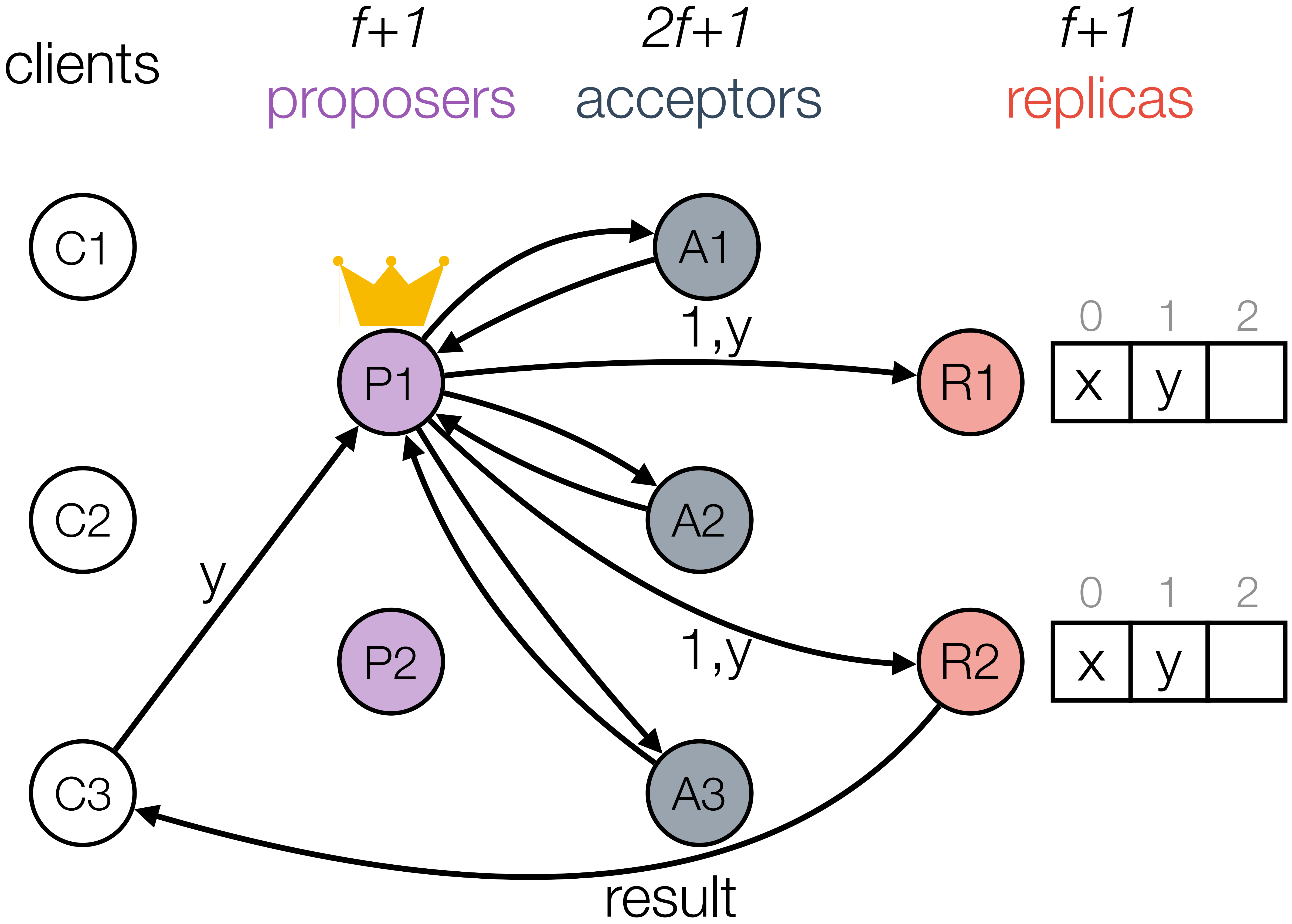

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
result
C1
x
2a
A1
2b
0,x
P1
R1
0 1 2
x
2a
2b
A2
C2
0 1 2
2a
0,x
P2
2b
R2
x
C3
A3


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
0 1 2
x
R2
0 1 2
x


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
y
P1
P2
A1
A2
A3
R1
0 1 2
x
R2
0 1 2
x


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
y
P1
P2
A1
A2
A3
R1
0 1 2
x
R2
0 1 2
x


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
y
P1
P2
A1
A2
A3
1,y
1,y
R1
R2
0 1 2
x y
0 1 2
x y

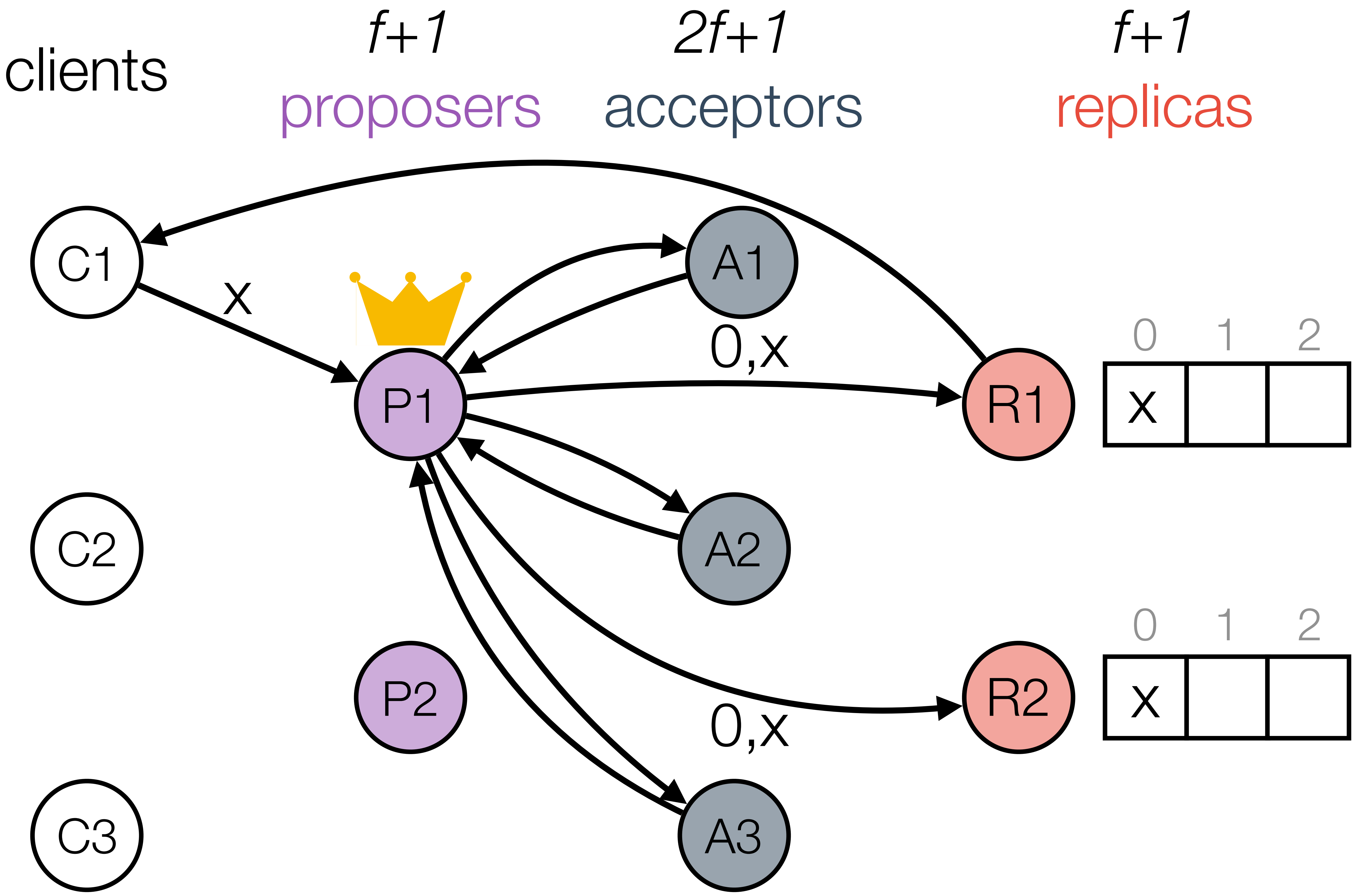

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
R2
y
1,y
1,y
0 1 2
x y
0 1 2
x y
result

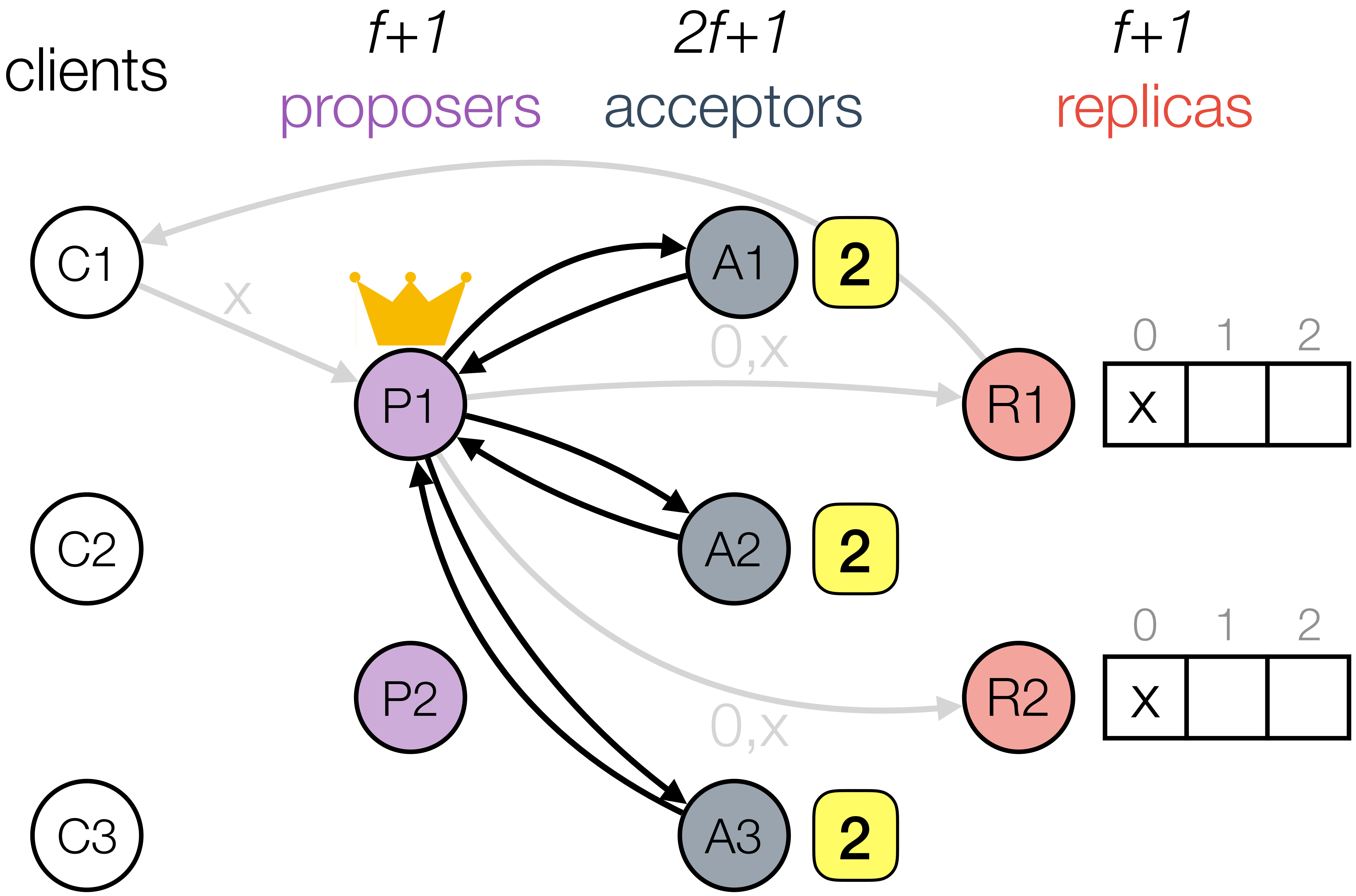
The MultiPaxos leader is a throughput bottleneck

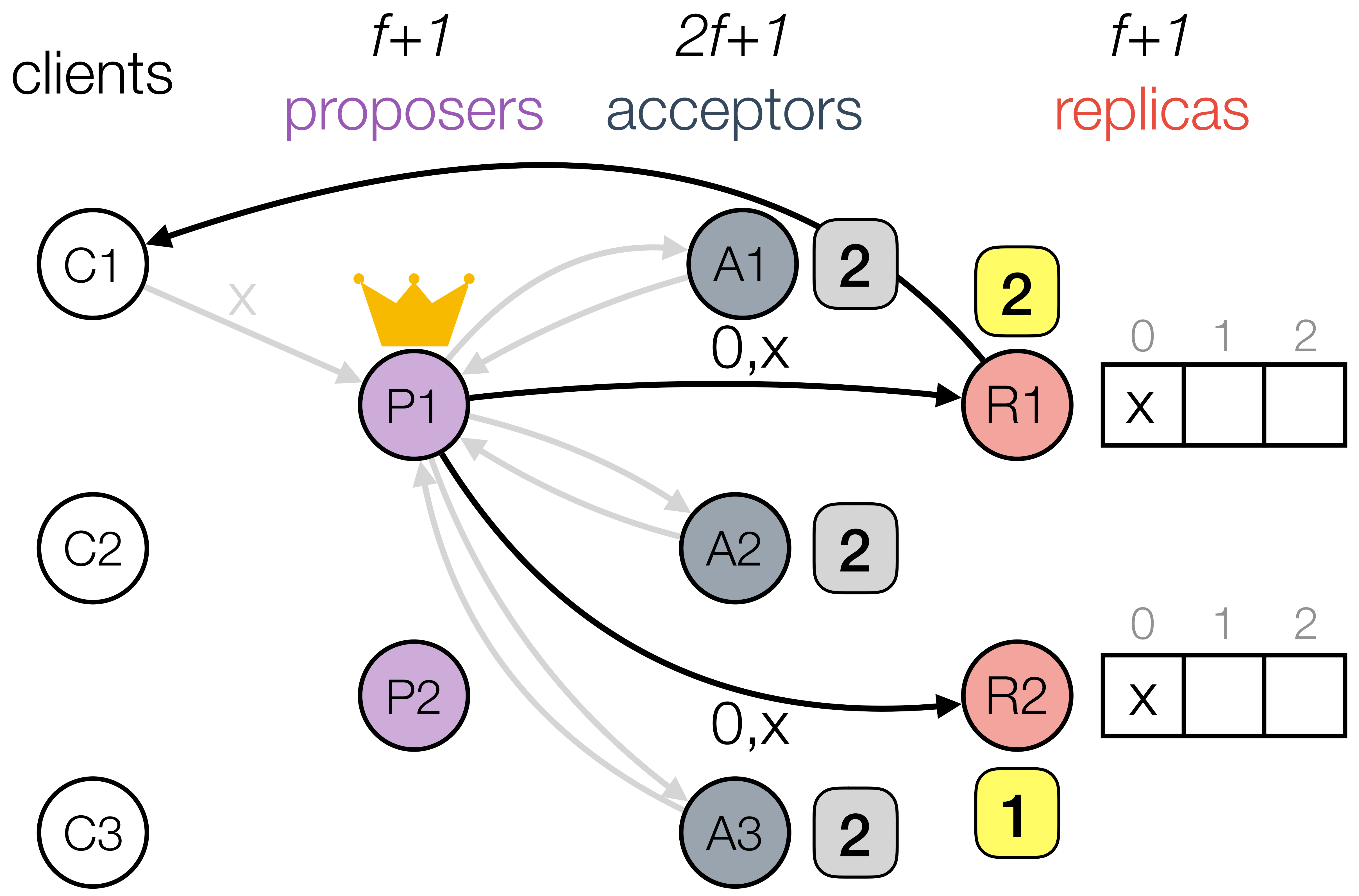
"The leader in Paxos is a bottleneck that limits throughput" -Mencius

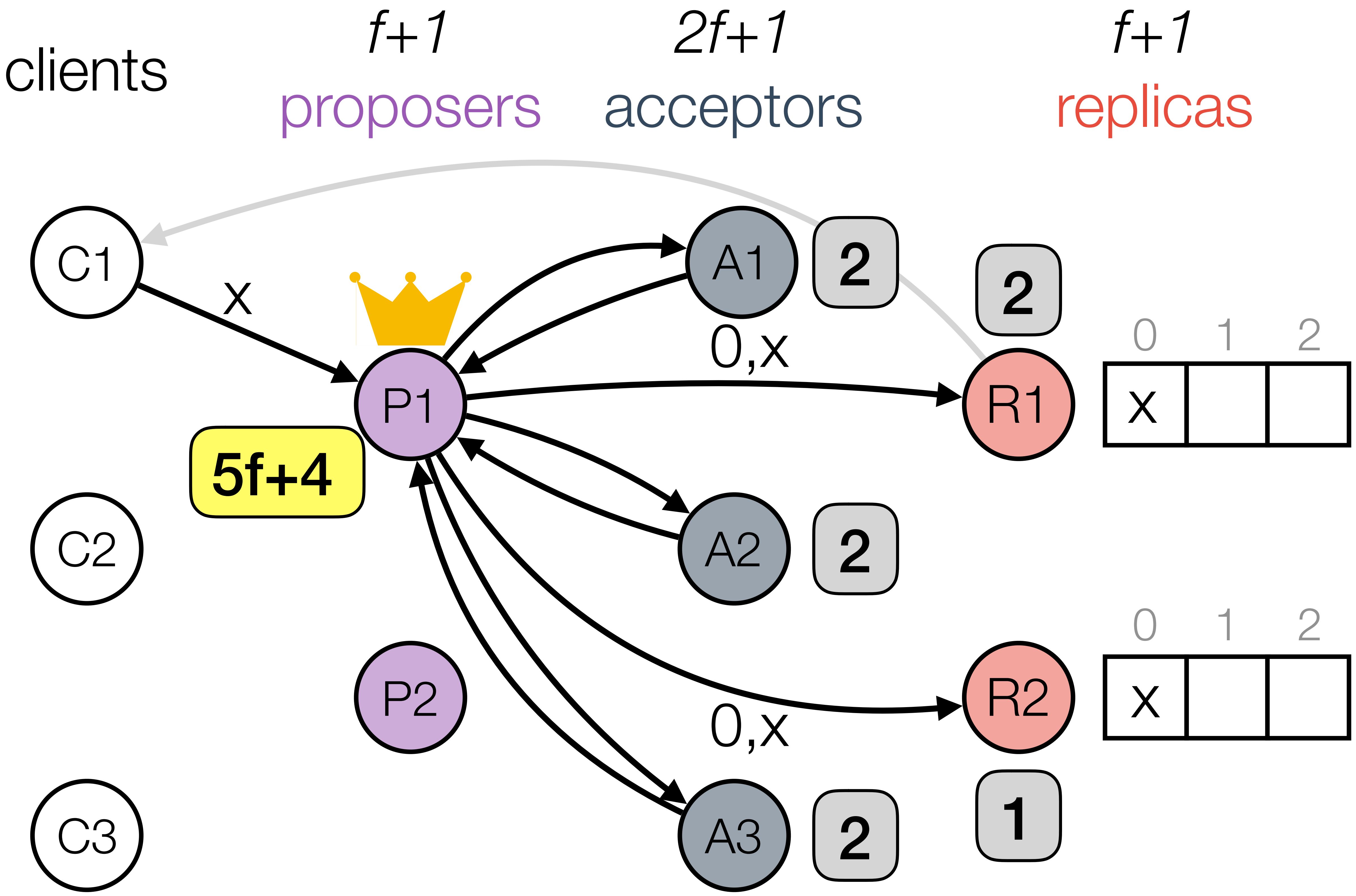
"It impairs scalability by placing a disproportionately high load on the master, which must process more messages than the other replicas" -EPaxos

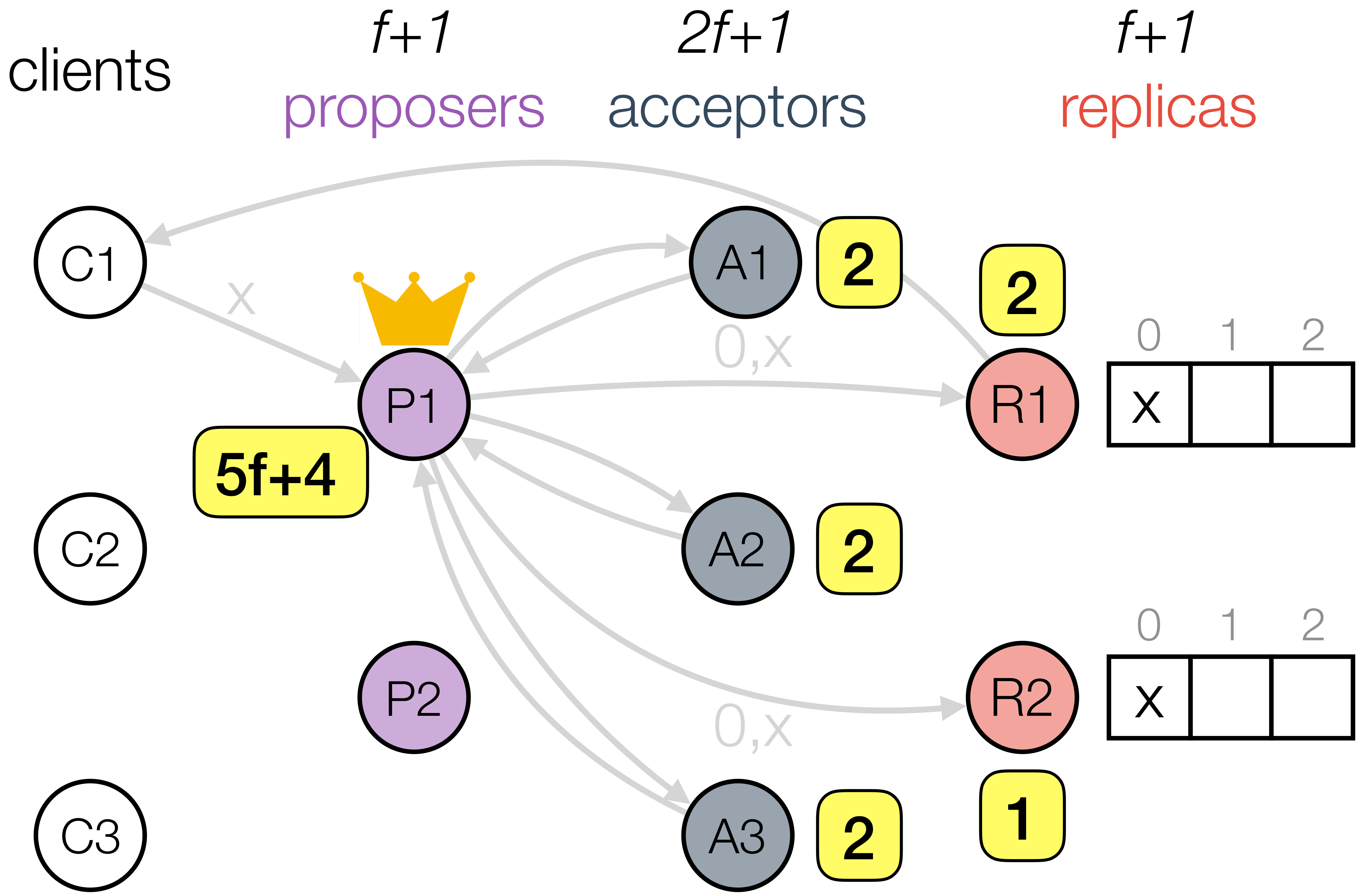
"In practice, [MultiPaxos'] performance is tied to the performance of the leader" -Caesar


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
P1
A1
0,x
R1
0 1 2
x
A2
C2
P2
R2
0 1 2
x
0,x
C3
A3


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
C2
C3
P1
P2
A1
2
0,x
A2
2
A3
2
0,x
R1
0 1 2
x
R2
0 1 2
x


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
A1
A2
A3
2
2
2
2
0,x
0,x
R1
R2
2
1
0 1 2
x
0 1 2
x

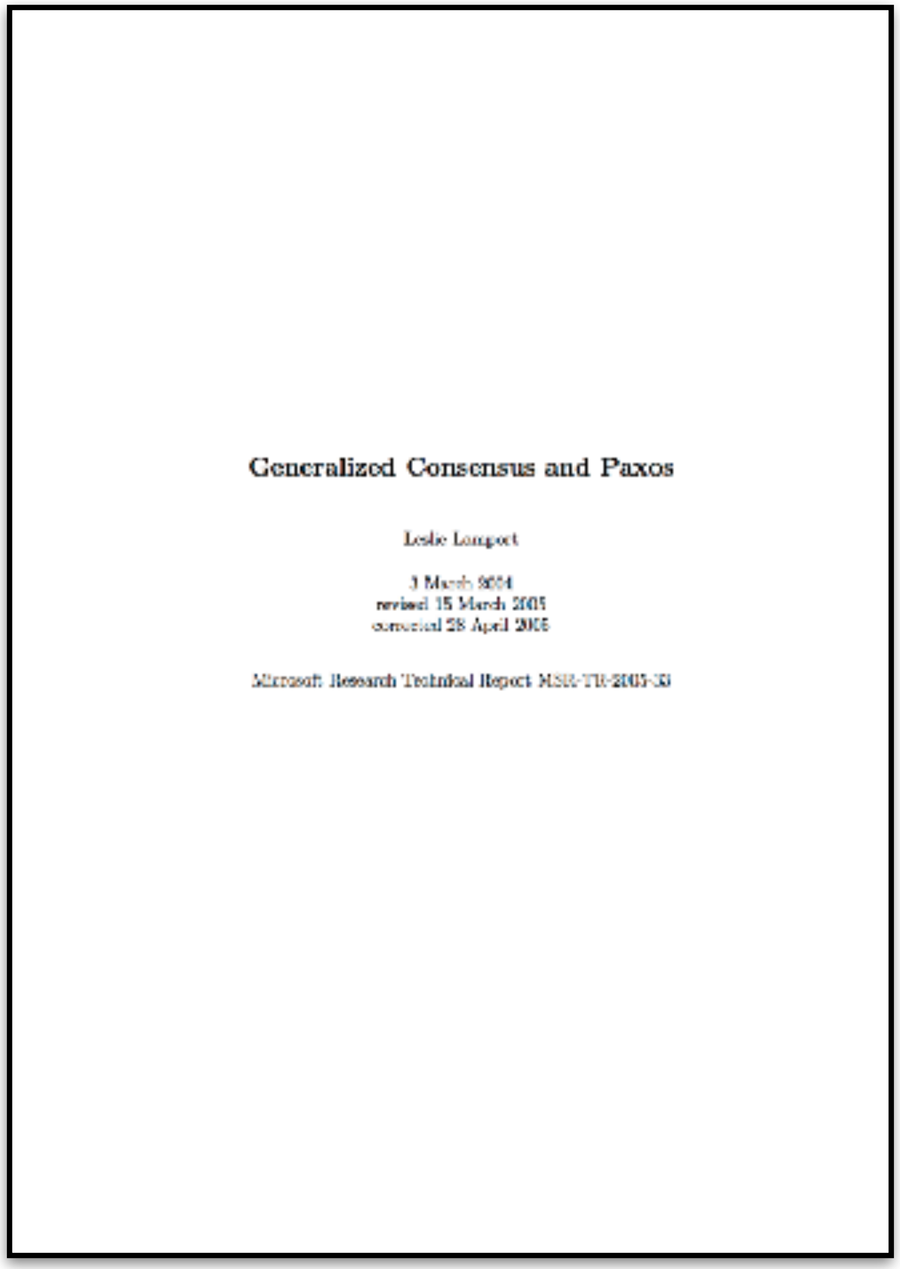

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
x
C2
C3
P1
5f+4
P2
A1
2
A2
2
A3
2
0,x
0,x
R1
2
0 1 2
x
R2
1
0 1 2
x



clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
5f+4
A1
A2
A3
2
2
2
0,x
0,x
R1
R2
2
1
0 1 2
x
0 1 2
x

Goal: to avoid the leader bottleneck and increase MultiPaxos' throughput.

Generalized Paxos



Generalized Consensus and Paxos

Leslie Lamport

3 March 2004
revised 15 March 2005
corrected 28 April 2005

Microsoft Research Technical Report MSR-TR-2005-33

EPaxos

There Is More Consensus in Egalitarian Parliaments

Iulian Moraru, David G. Andersen, Michael Kaminsky
Carnegie Mellon University and Intel Labs

Abstract

This paper describes the design and implementation of Egalitarian Paxos (EPaxos), a new distributed consensus algorithm based on Paxos. EPaxos achieves three goals: (1) optimal commit latency in the wide-area when tolerating one and two failures, under realistic conditions; (2) uniform load balancing across all replicas (thus achieving high throughput); and (3) graceful performance degradation when replicas are slow or crash.

Egalitarian Paxos is to our knowledge the first protocol to achieve the previously stated goals efficiently—that is, requiring only a simple majority of replicas to be non-faulty, using a number of messages linear in the number of replicas to choose a command, and committing commands after just one communication round (one round trip) in the common case or after at most two rounds in any case. We prove Egalitarian Paxos's properties theoretically and demonstrate its advantages empirically through an implementation running on Amazon EC2.

1 Introduction

Distributed computing places two main demands on replication protocols: (1) high throughput for replication inside a computing cluster; and (2) low latency for replication across data centers. Today's clusters use fault-tolerant coordination engines such as Chubby [4], Boxwood [22], or ZooKeeper [12] for activities including operation sequencing, coordination, leader election, and resource discovery. Many databases are accessed simultaneously from different continents, requiring geo-replication [2, 8].

An important limitation on these systems is that during efficient, failure-free operation, all clients communicate with a single master (or leader) server at all times.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

Copyright is held by the Owner/Author(s).
SOSP'13, Nov. 3–6, 2013, Farmington, Pennsylvania, USA.
ACM 978-1-4503-2388-8/13/11.
http://dx.doi.org/10.1145/2517349.2517350

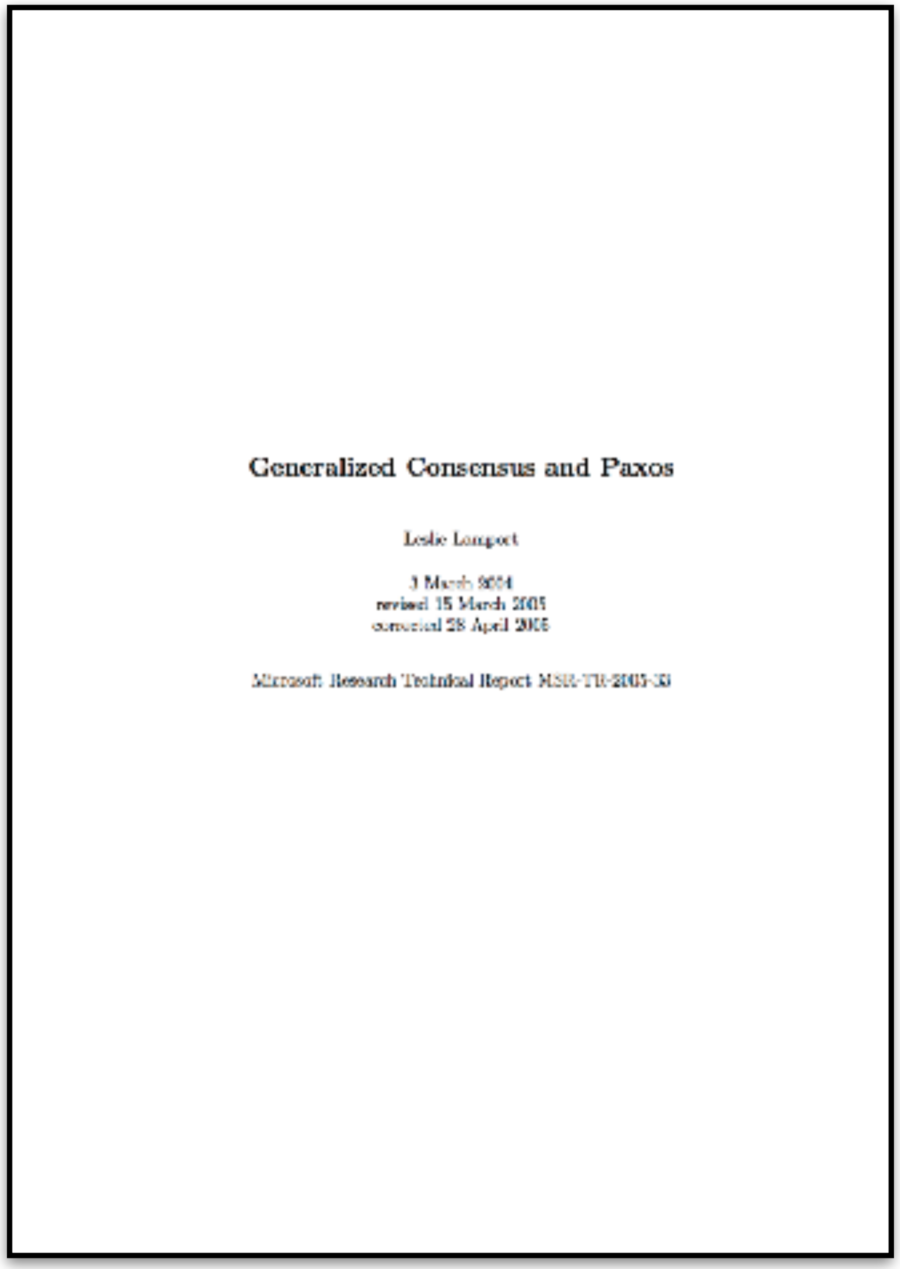
This optimization—termed "Multi-Paxos" for systems based on the Paxos protocol [16]—is important to achieving high throughput in practical systems [7]. Changing the leader requires invoking additional consensus mechanisms that substantially reduce performance.

This algorithmic limitation has several important consequences. First, it impairs scalability by placing a disproportionately high load on the master, which must process more messages than the other replicas [23]. Second, when performing geo-replication, clients incur additional latency for communicating with a remote master. Third, as we show in this paper, traditional Paxos variants are sensitive to both long-term and transient load spikes and network delays that increase latency at the master. Finally, this single-master optimization can harm availability: if the master fails, the system cannot service requests until a new master is elected. Previously proposed solutions such as partitioning or using proxy servers are undesirable because they restrict the type of operations the cluster can perform. For example, a partitioned cluster cannot perform atomic operations across partitions without using additional techniques.

Egalitarian Paxos (EPaxos) has no designated leader process. Instead, clients can choose at every step which replica to submit a command to, and in most cases the command is committed without interfering with other concurrent commands. This allows the system to evenly distribute the load to all replicas, eliminating the first bottleneck identified above (i.e., having one server that must be on the critical path for all communication). EPaxos's flexible load distribution better handles permanently or transiently slow nodes, as well as the latency heterogeneity caused by geographical distribution of replicas; this substantially reduces both the median and tail commit latency. Finally, the system can provide higher availability and higher performance under failures because there is no transient interruption caused by leader election: there is no leader, and hence, no need for leader election, as long as more than half of the replicas are available.

We begin by reviewing the core Paxos algorithm and the intuition behind Egalitarian Paxos in Section 2. We then describe several Paxos variants that reduce overhead or commit latency in Section 3. Throughout the paper we compare extensively against Multi-Paxos and two recent

Generalized Paxos



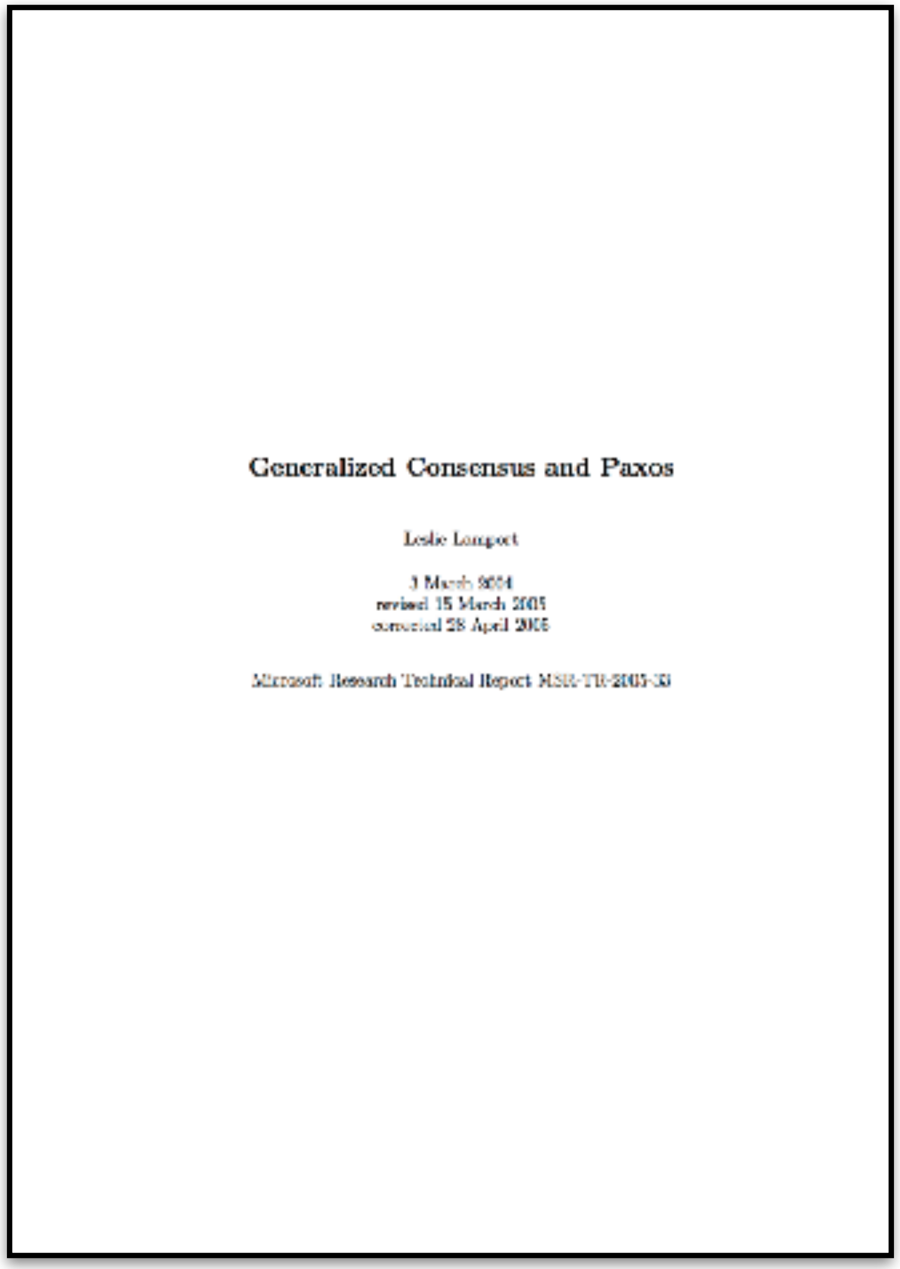
Generalized Consensus and Paxos

Leslie Lamport

3 March 2004
revised 15 March 2005
corrected 28 April 2005

Microsoft Research Technical Report MSR-TR-2005-33

Generalized Paxos

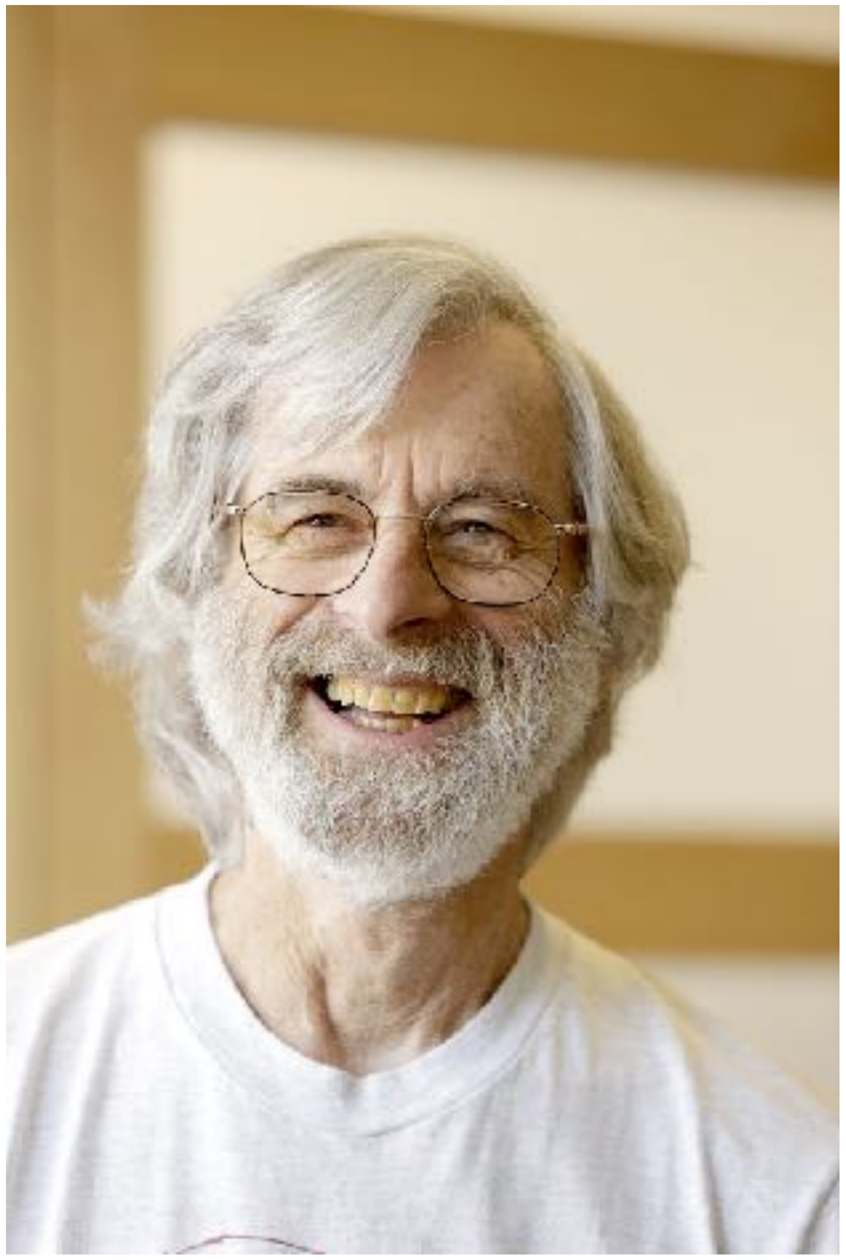


Generalized Consensus and Paxos

Leslie Lamport

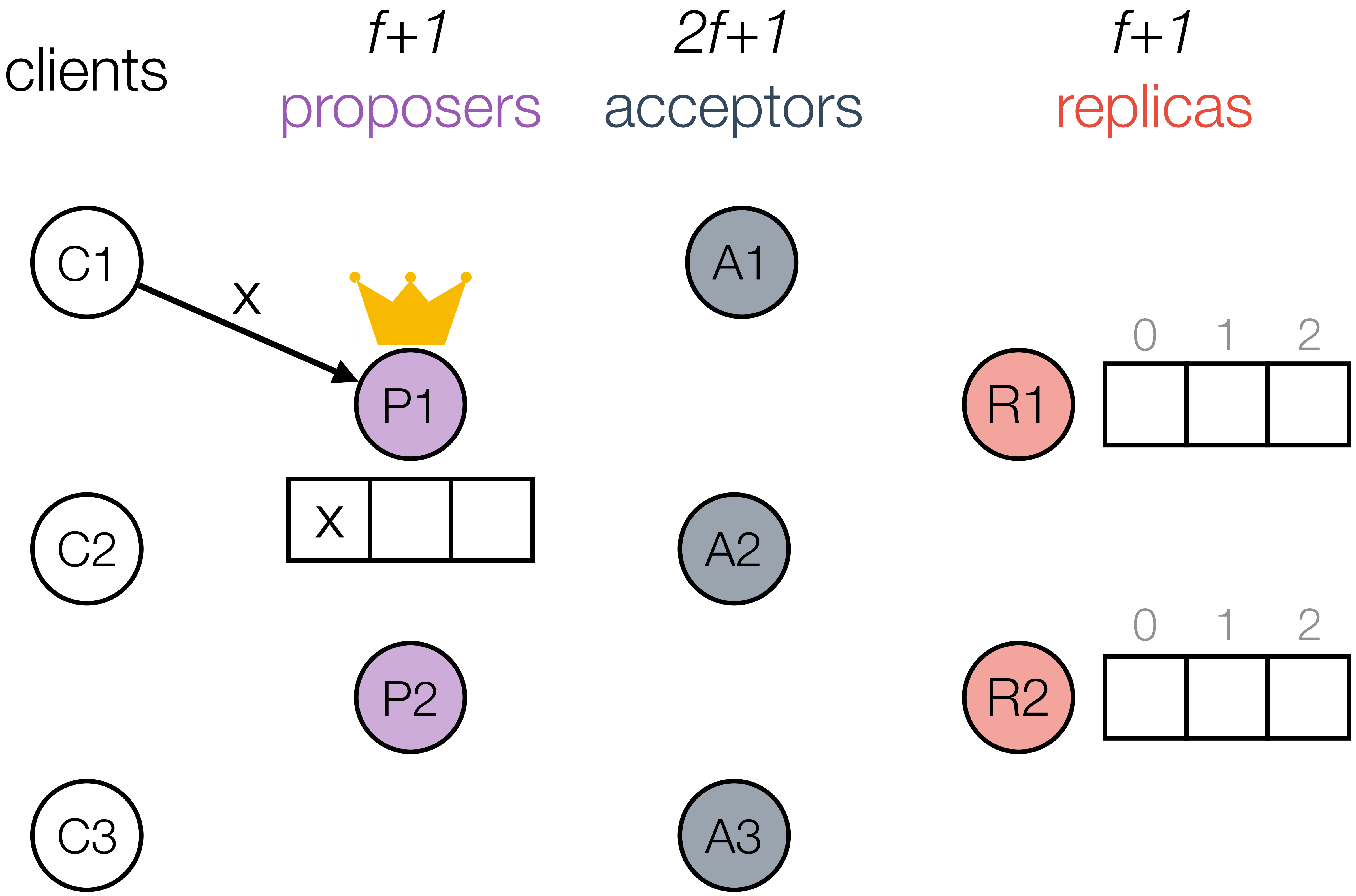
3 March 2004
revised 15 March 2005
corrected 28 April 2005

Microsoft Research Technical Report MSR-TR-2005-33

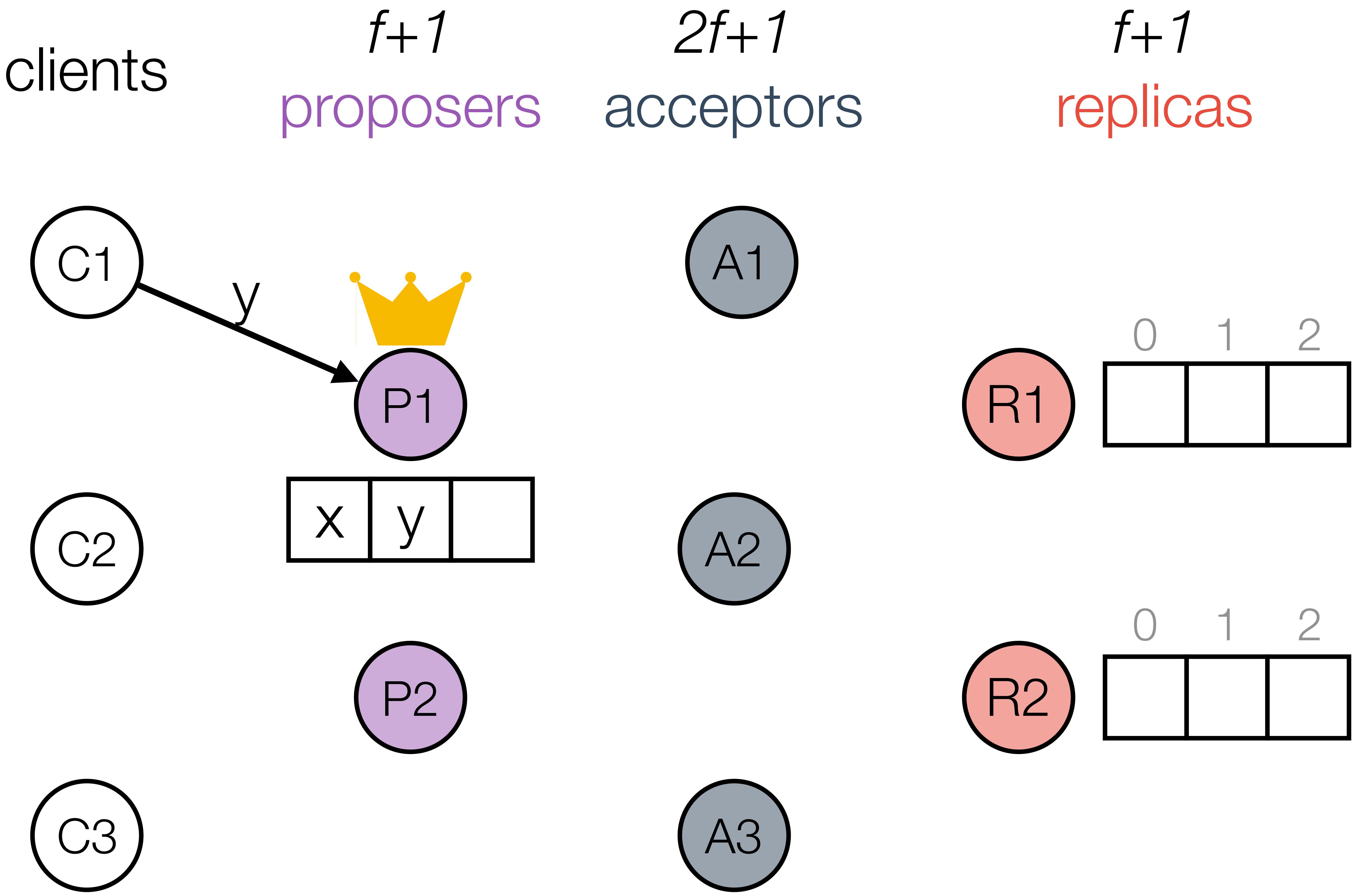


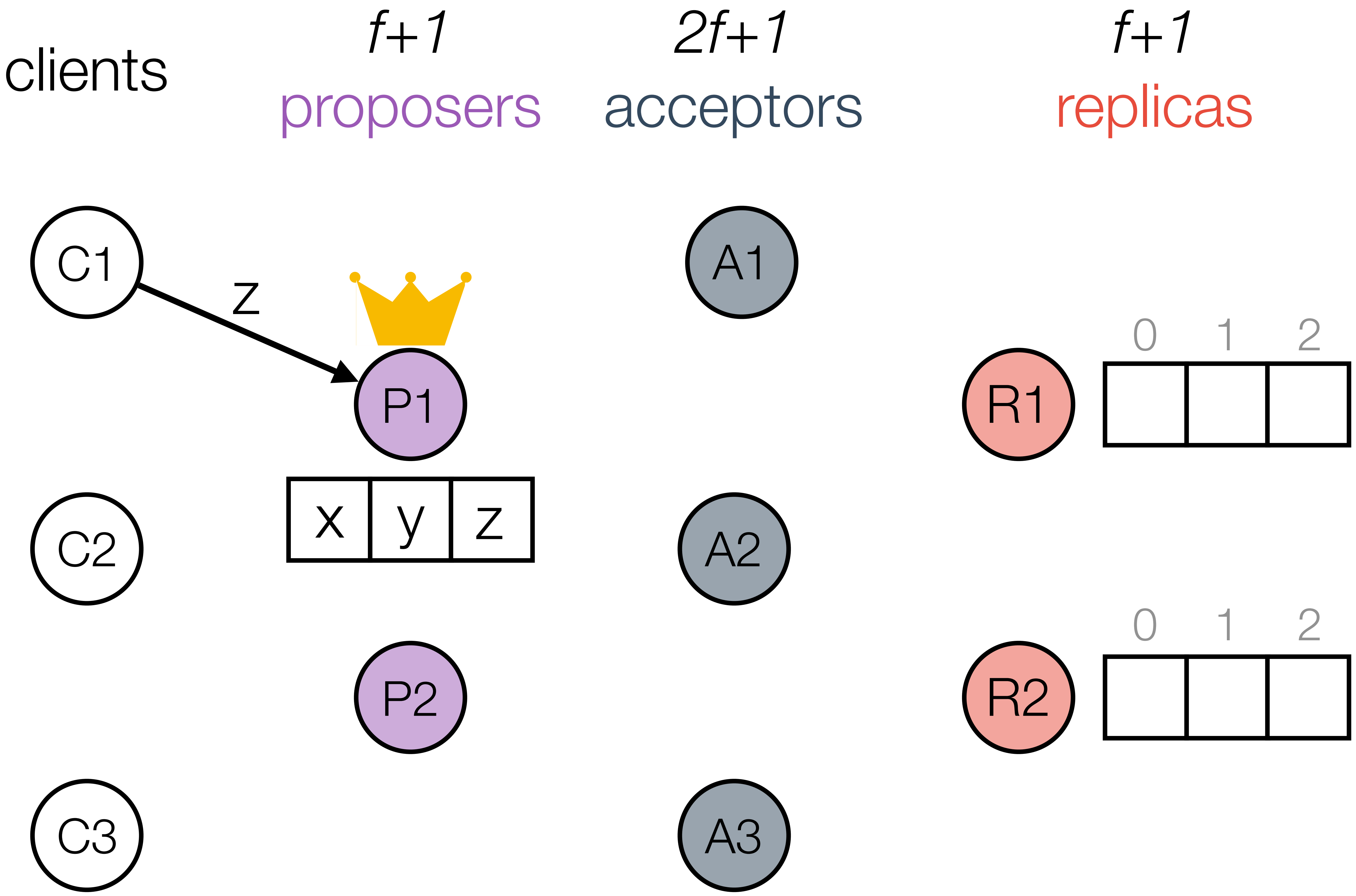
To increase MultiPaxos' throughput, we must trade off complexity?

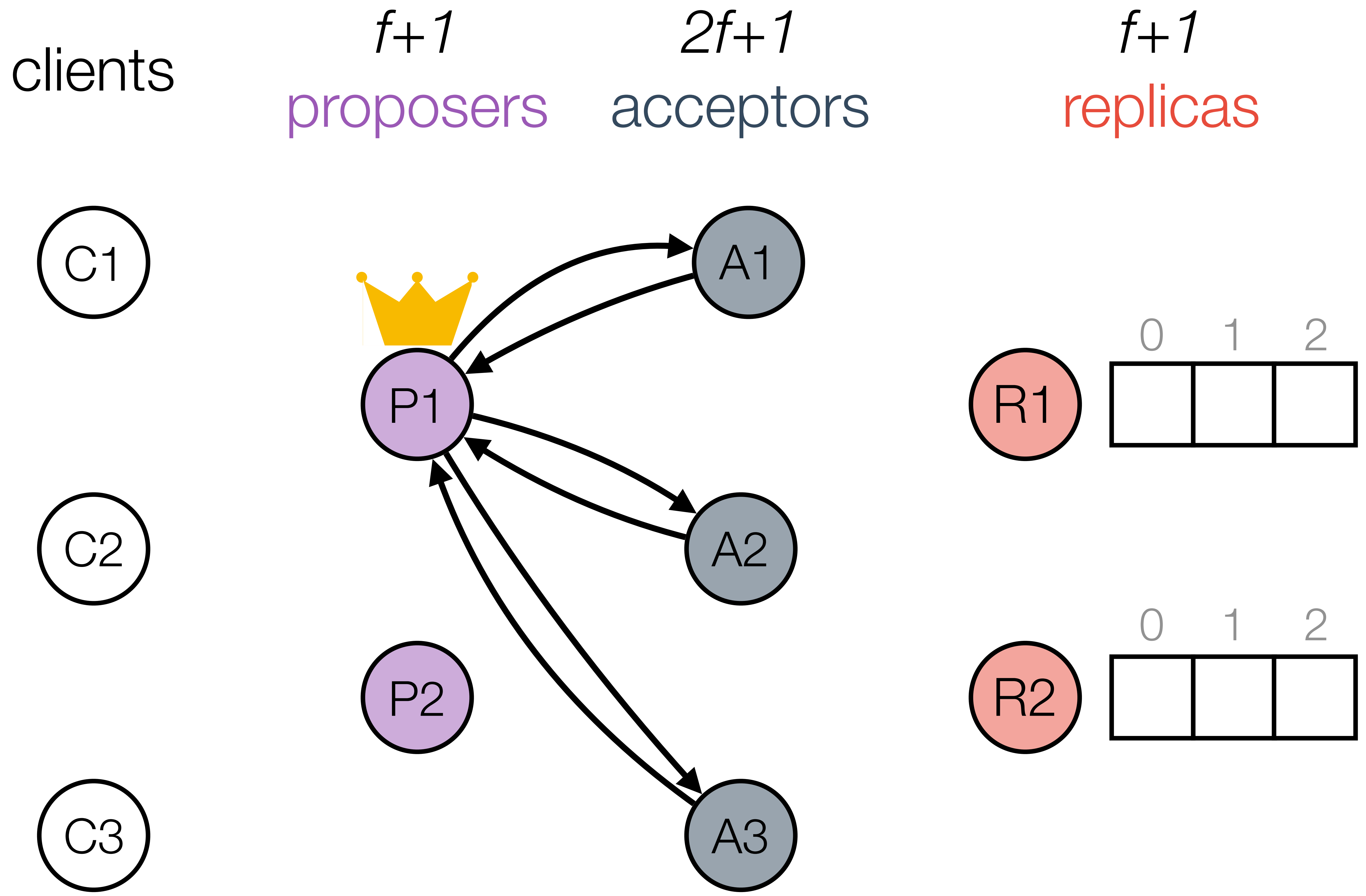
Key Insight: The MultiPaxos leader has two independent responsibilities.



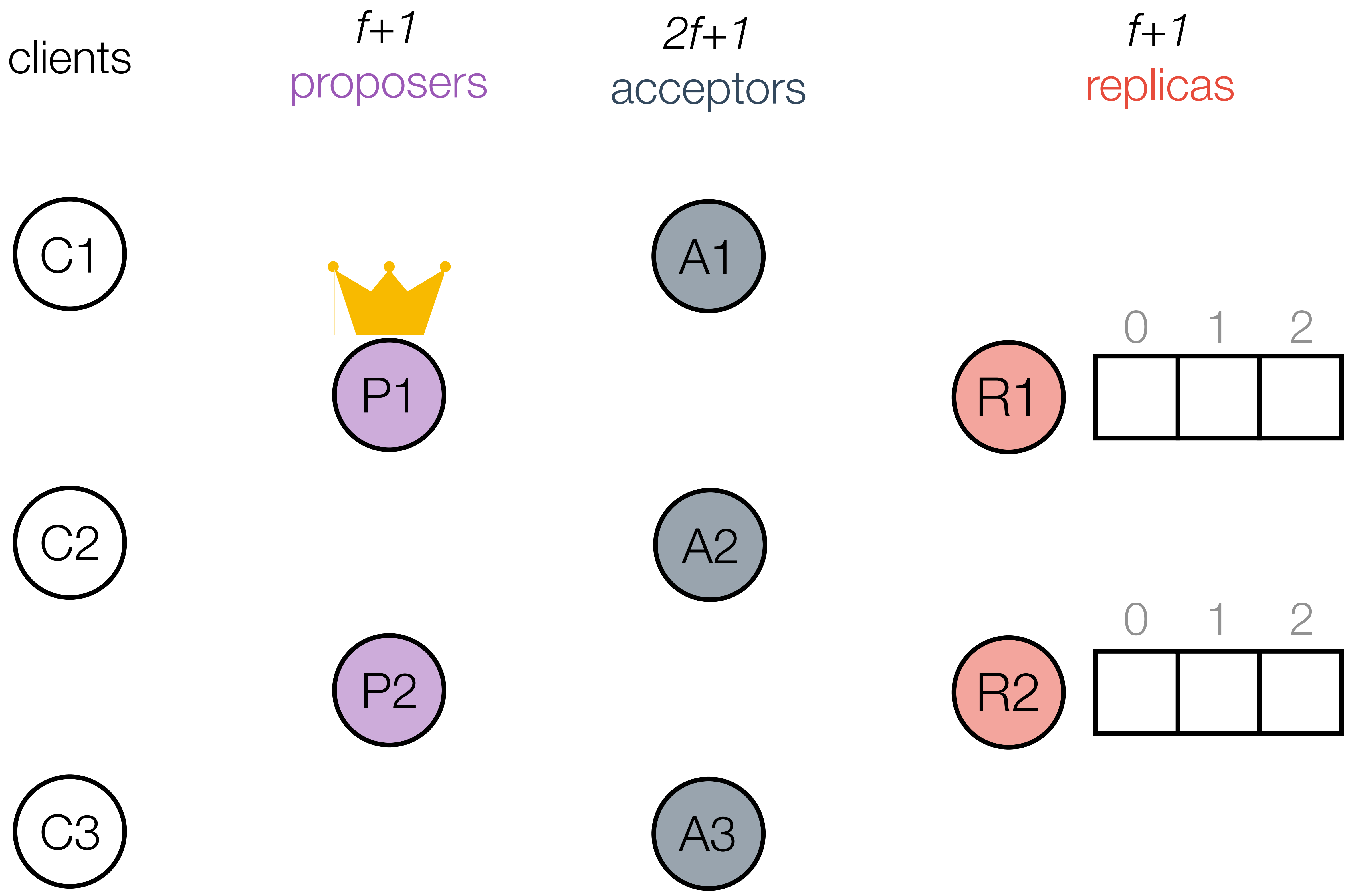
clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
x
P2
A1
A2
A3
0 1 2
R1
0 1 2
R2

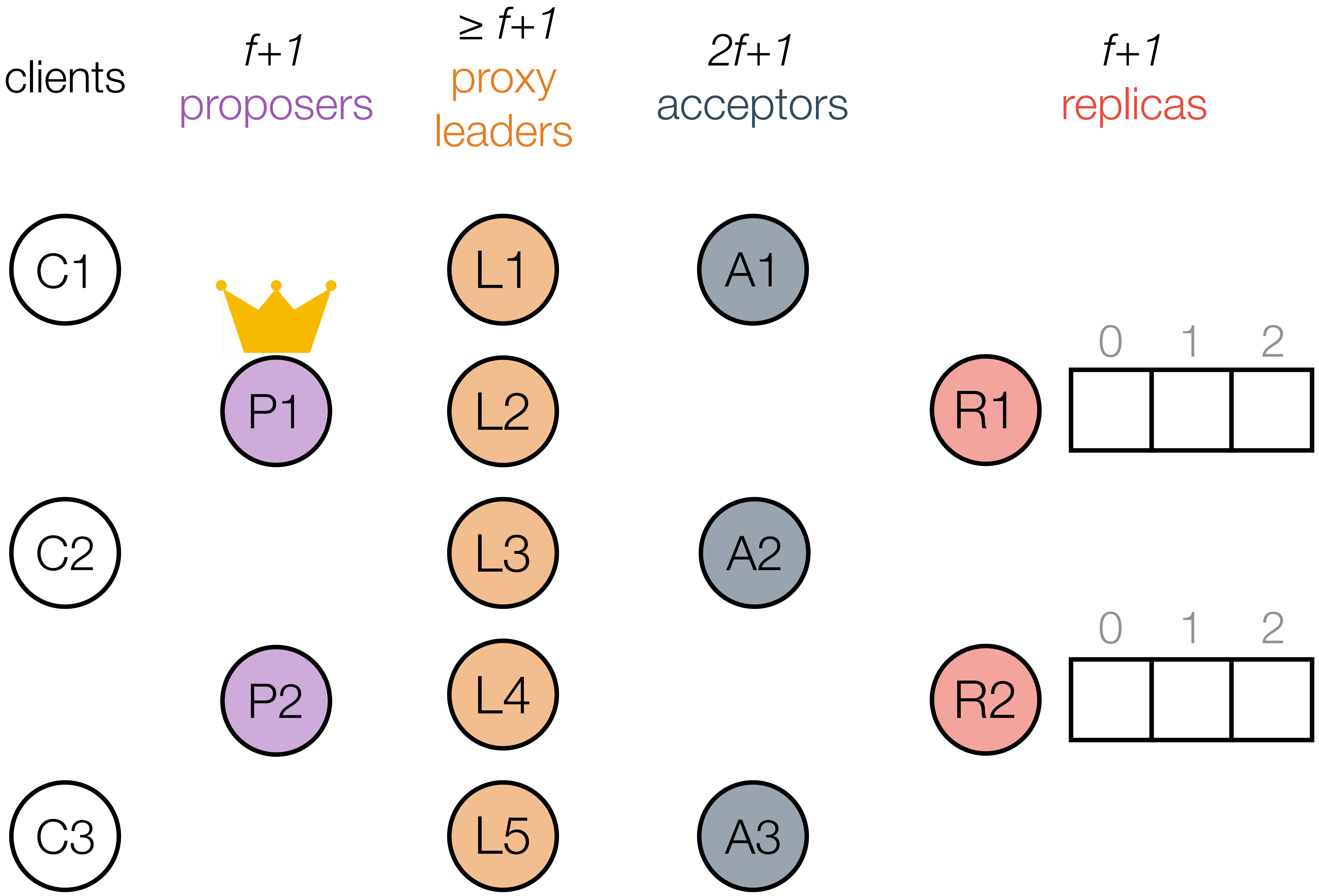

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
y
P1
x
y
A1
0
1
2
R1
C2
A2
P2
0
1
2
R2
C3
A3

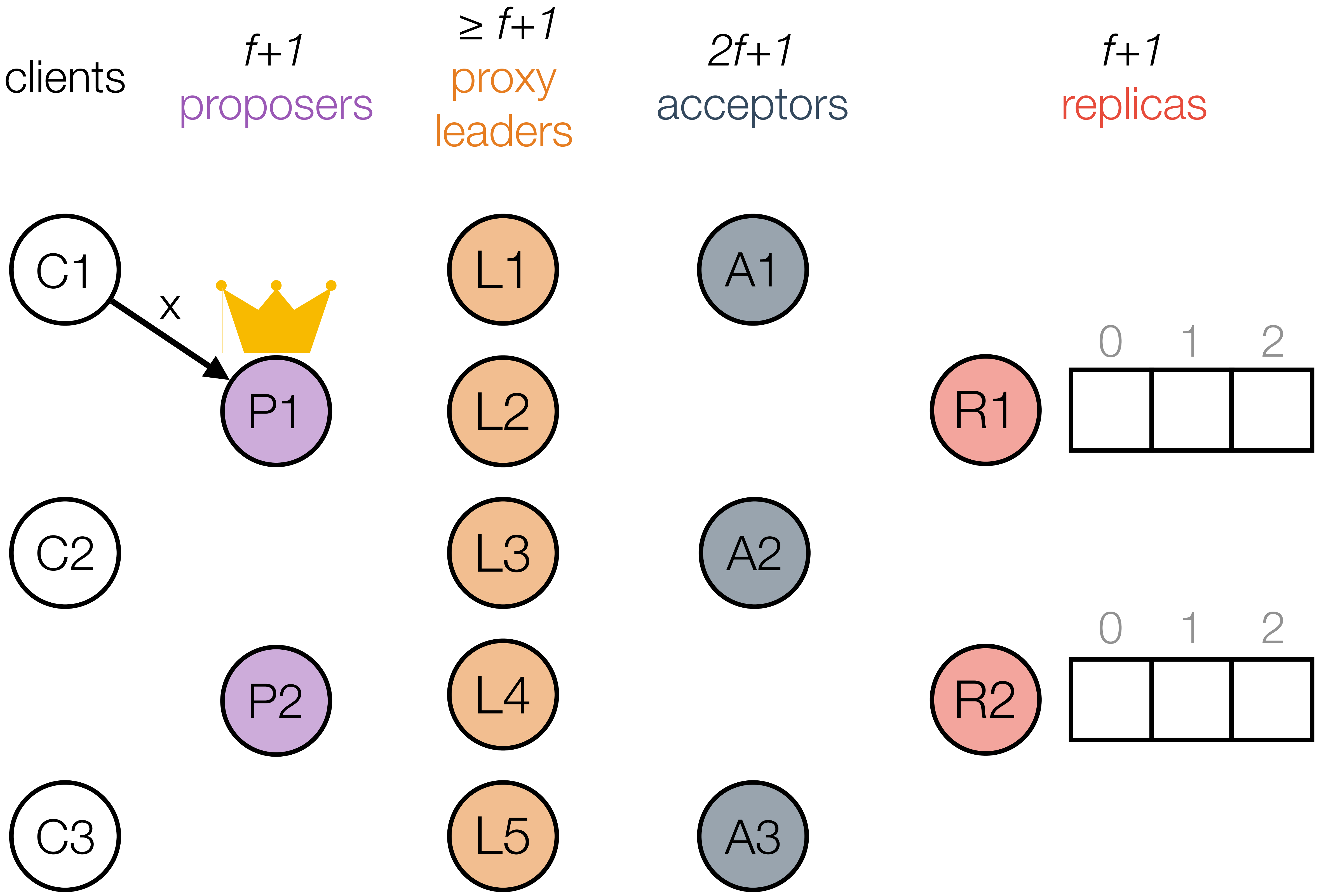

clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
z
P1
x
y
z
C2
P2
C3
A1
A2
A3
R1
0
1
2
R2
0
1
2


clients
f+1
proposers
2*f*+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
0 1 2
R2
0 1 2

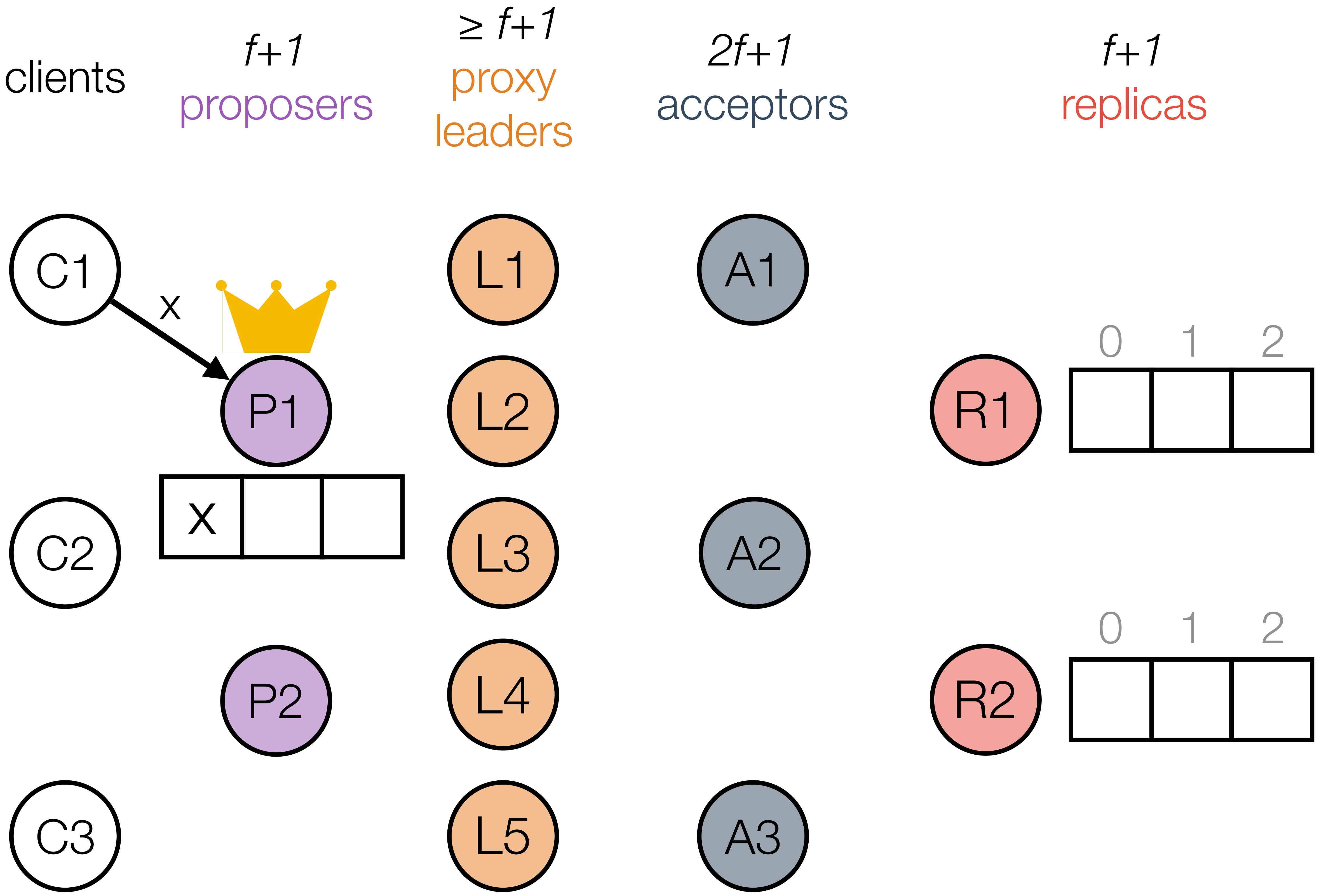
Key Insight: Decouple the MultiPaxos leader’s two responsibilities

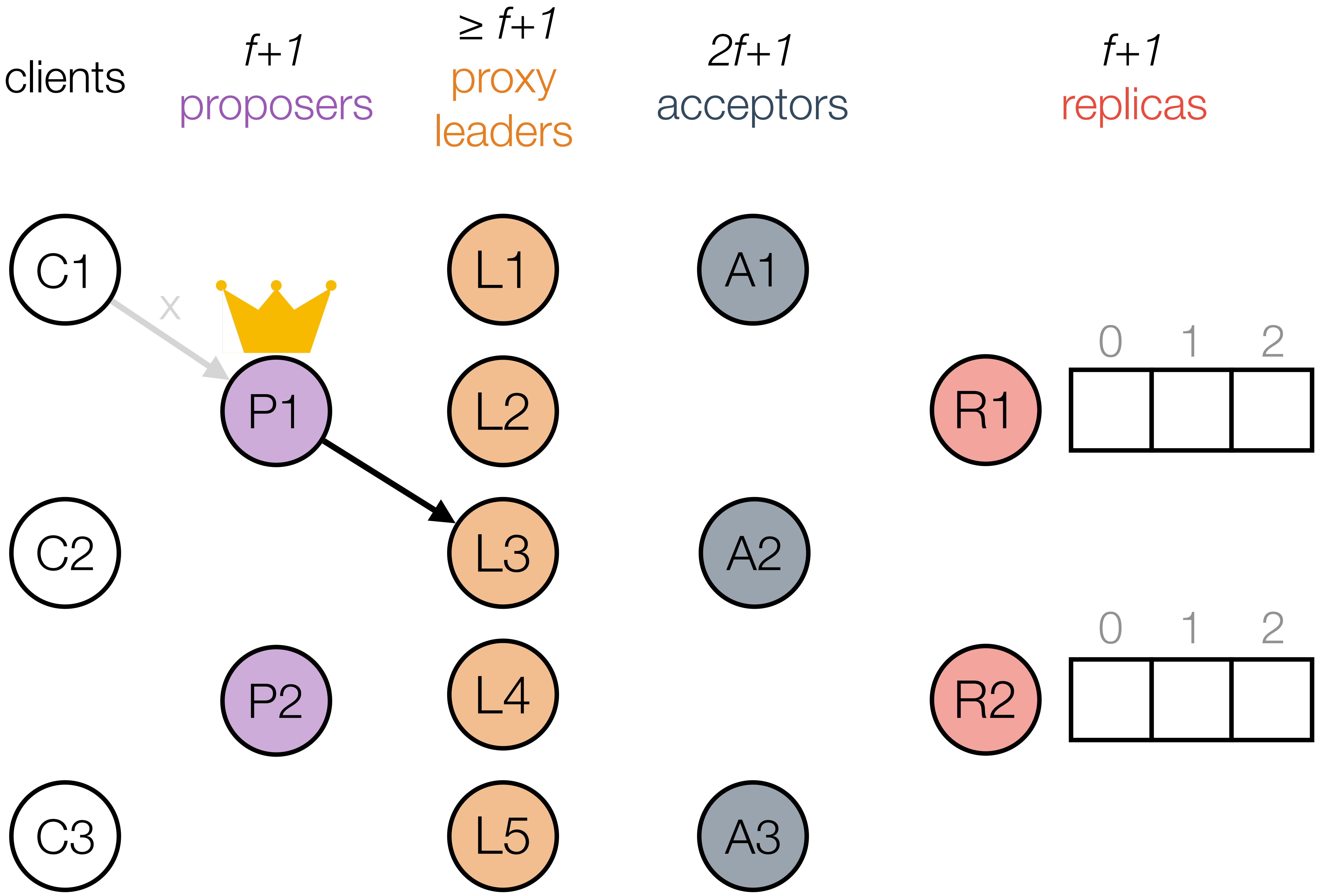

clients
f+*1*
proposers
2f+*1*
acceptors
f+*1*
replicas
C1
C2
C3
P1
P2
A1
A2
A3
R1
0 1 2
R2
0 1 2

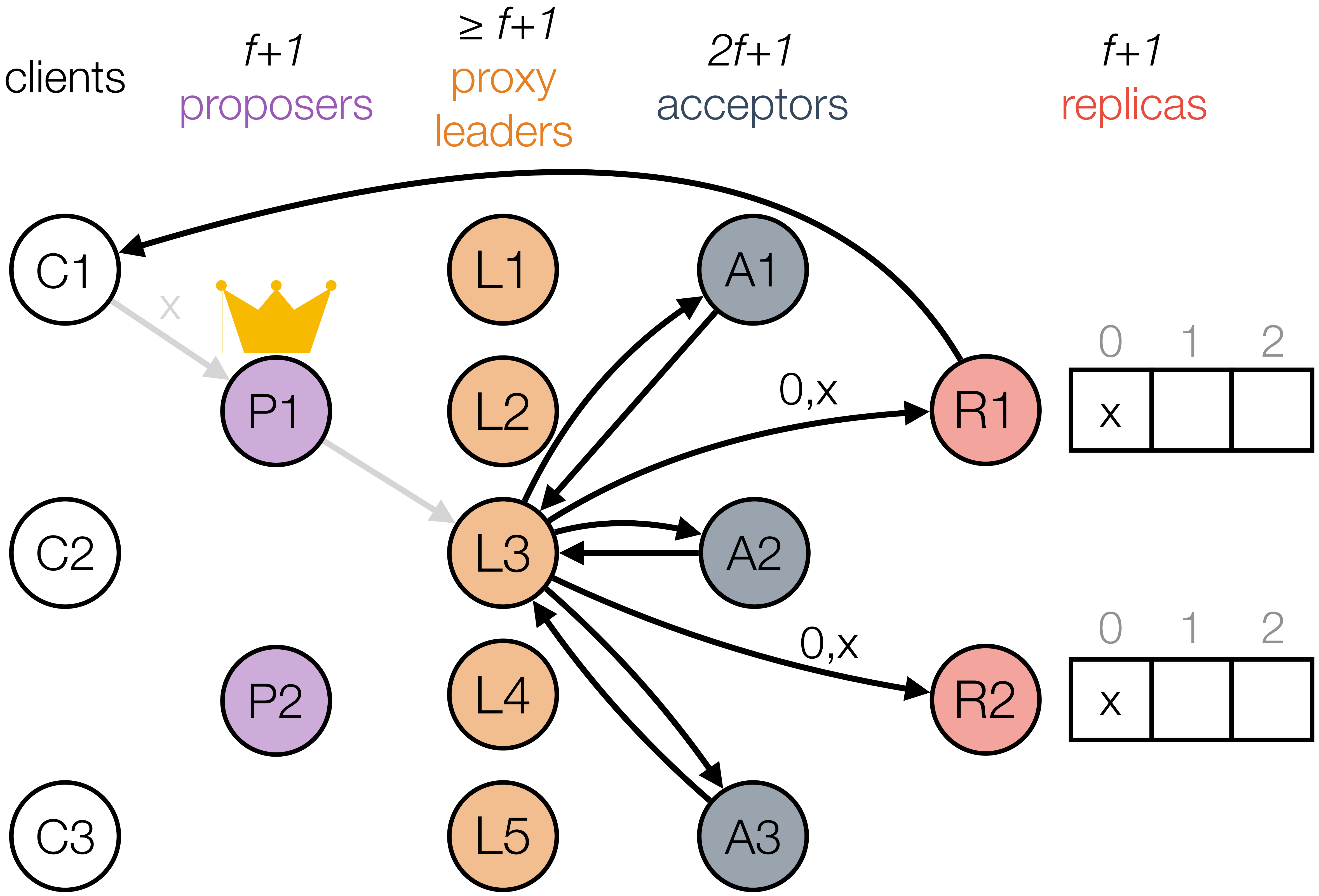

clients
$f+1$
proposers
$\geq f+1$
proxy leaders
$2f+1$
acceptors
$f+1$
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
0 1 2
R2
0 1 2



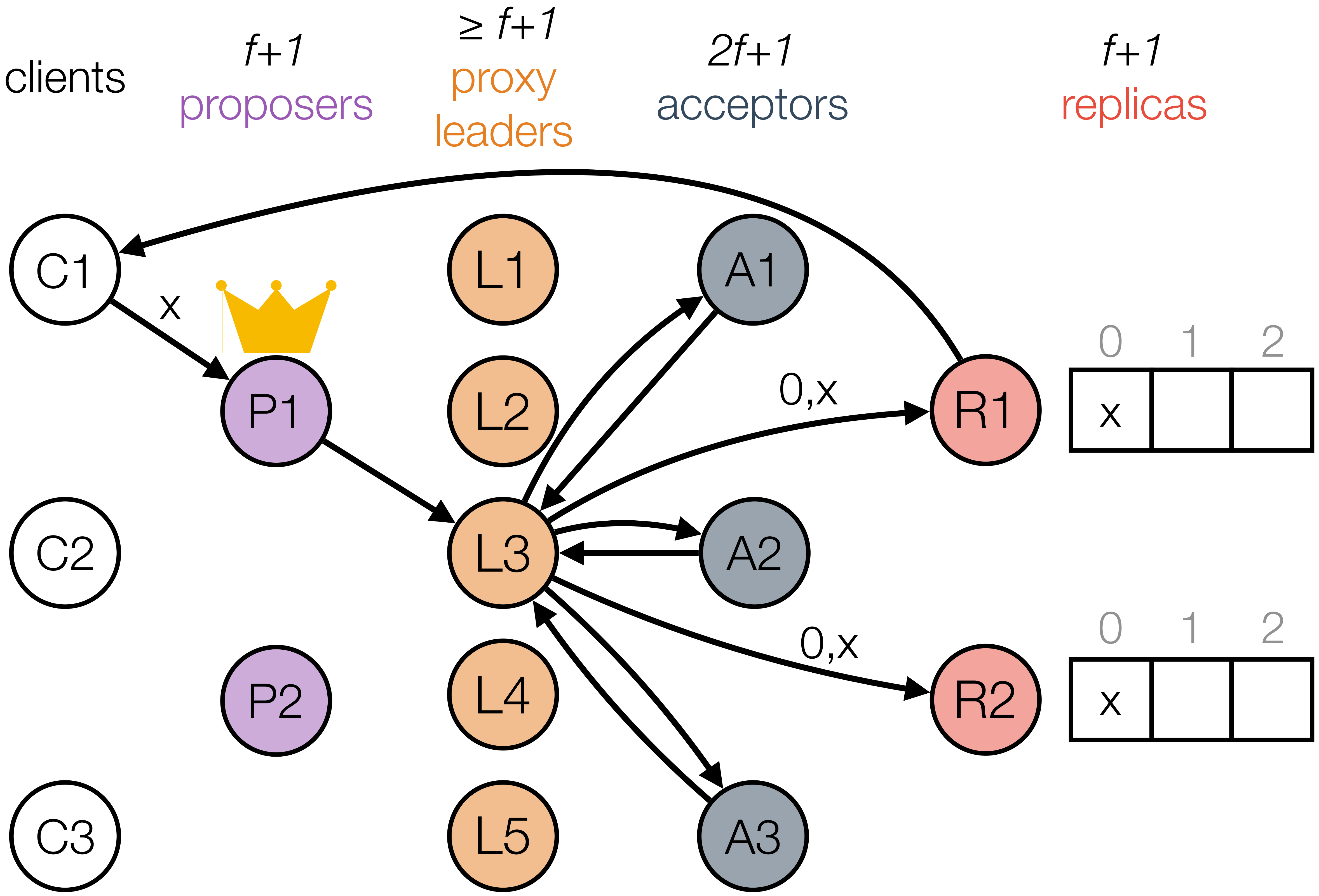
clients
f+1
proposers
≥ f+1
proxy leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
R2
0
1
2
0
1
2

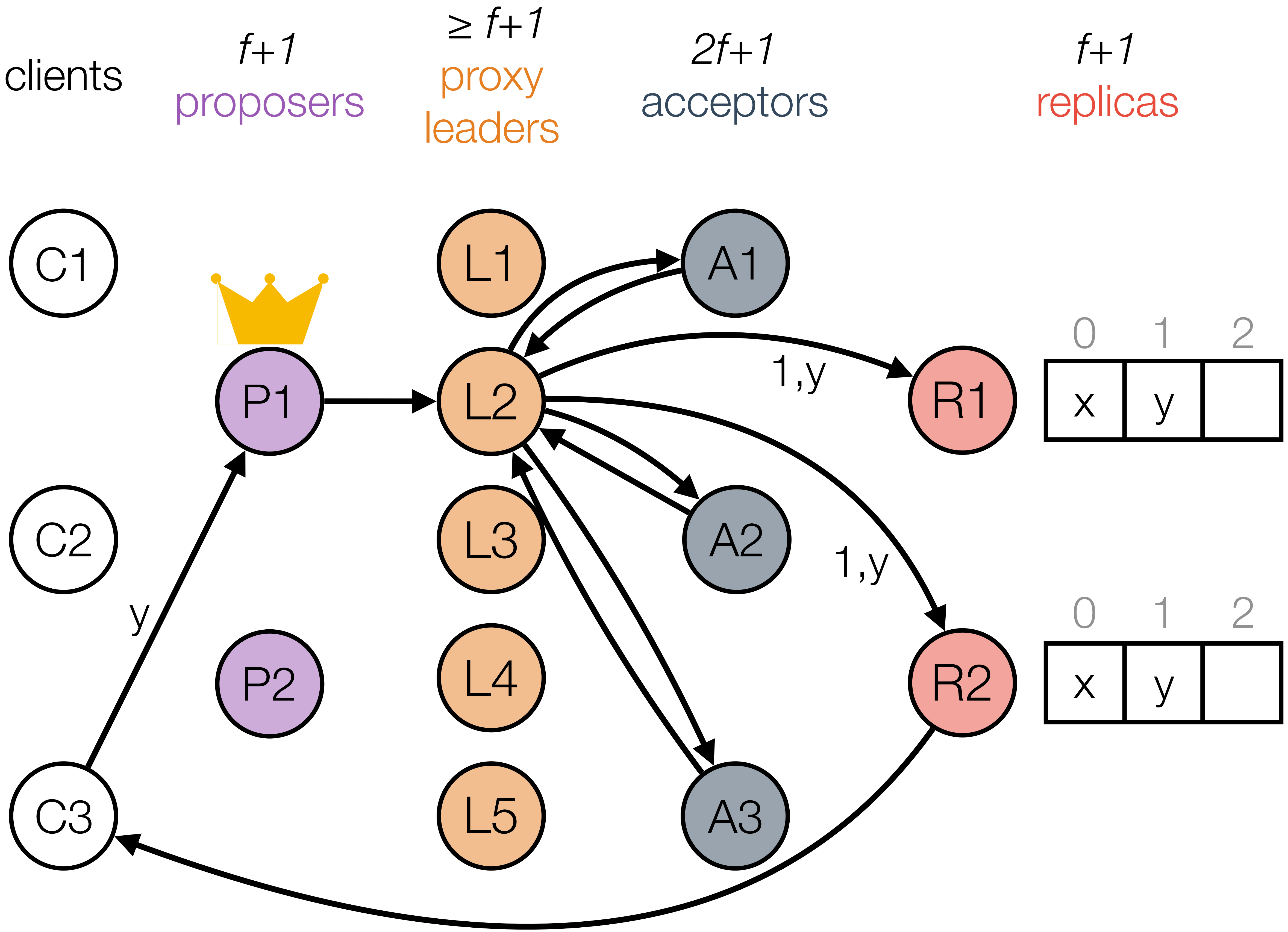

clients
f+1
proposers
≥ f+1
proxy leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
x
L1
L2
L3
L4
L5
A1
A2
A3
R1
R2
0
1
2
0
1
2


clients
$f+1$
proposers
$\geq f+1$
proxy leaders
$2f+1$
acceptors
$f+1$
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
R2
0
1
2
0
1
2

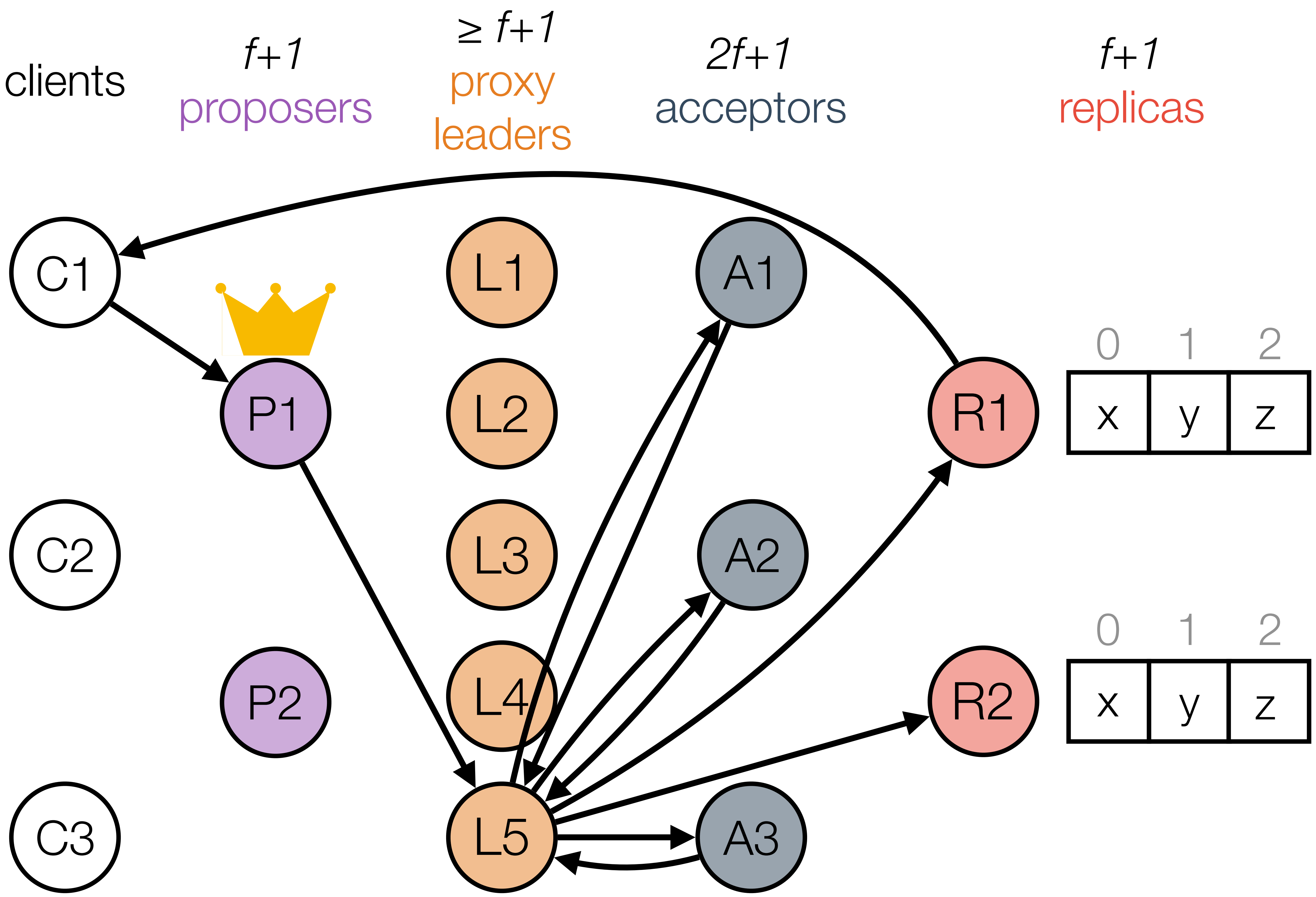

clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
0,x
0,x
R1
R2
0
1
2
x
0
1
2
x

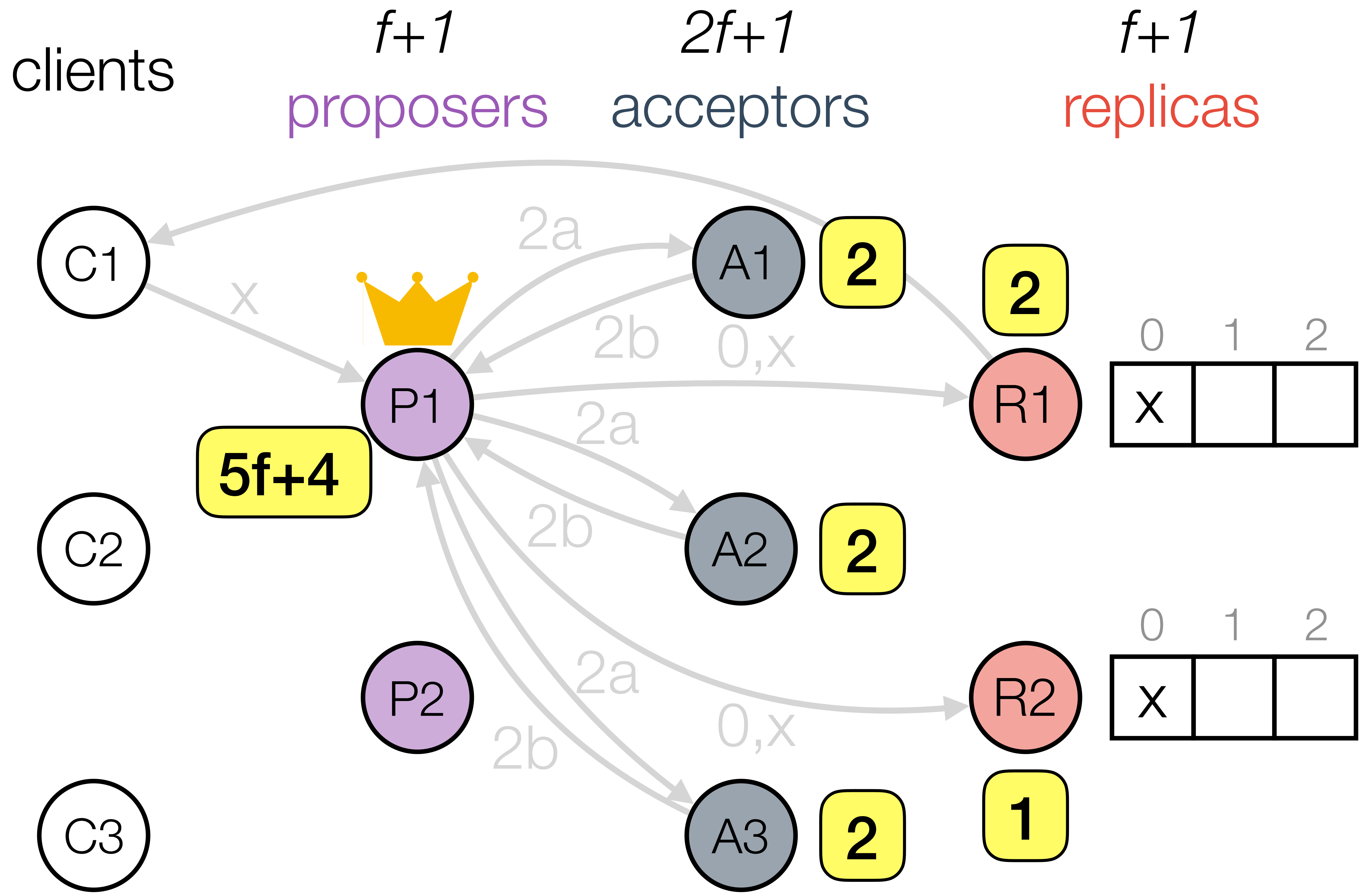
Key Insight: Scale up the proxy leaders


clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
x
P1
L1
L2
L3
A1
A2
A3
0,x
R1
0 1 2
x
C2
P2
L4
0,x
R2
0 1 2
x
C3
L5

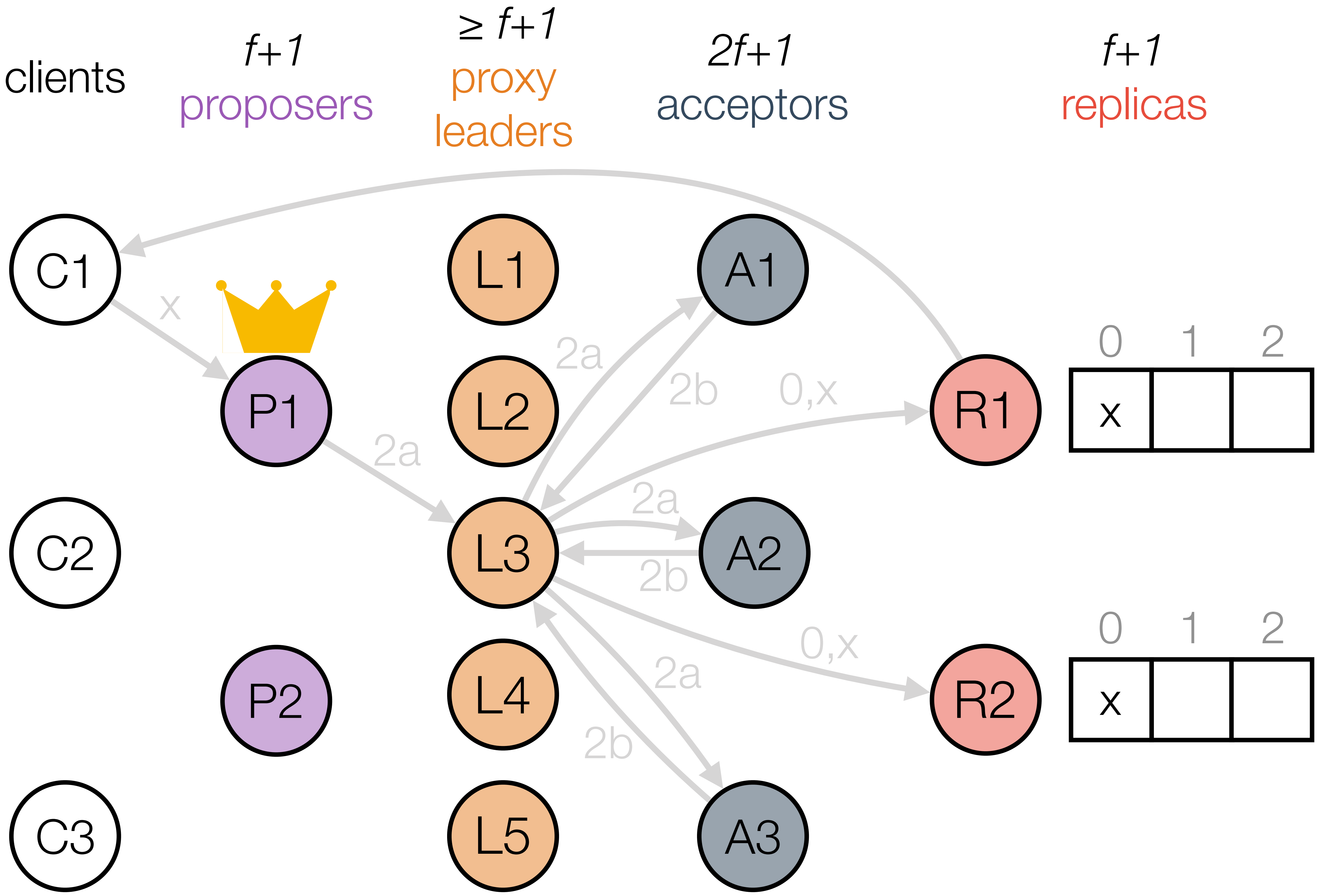


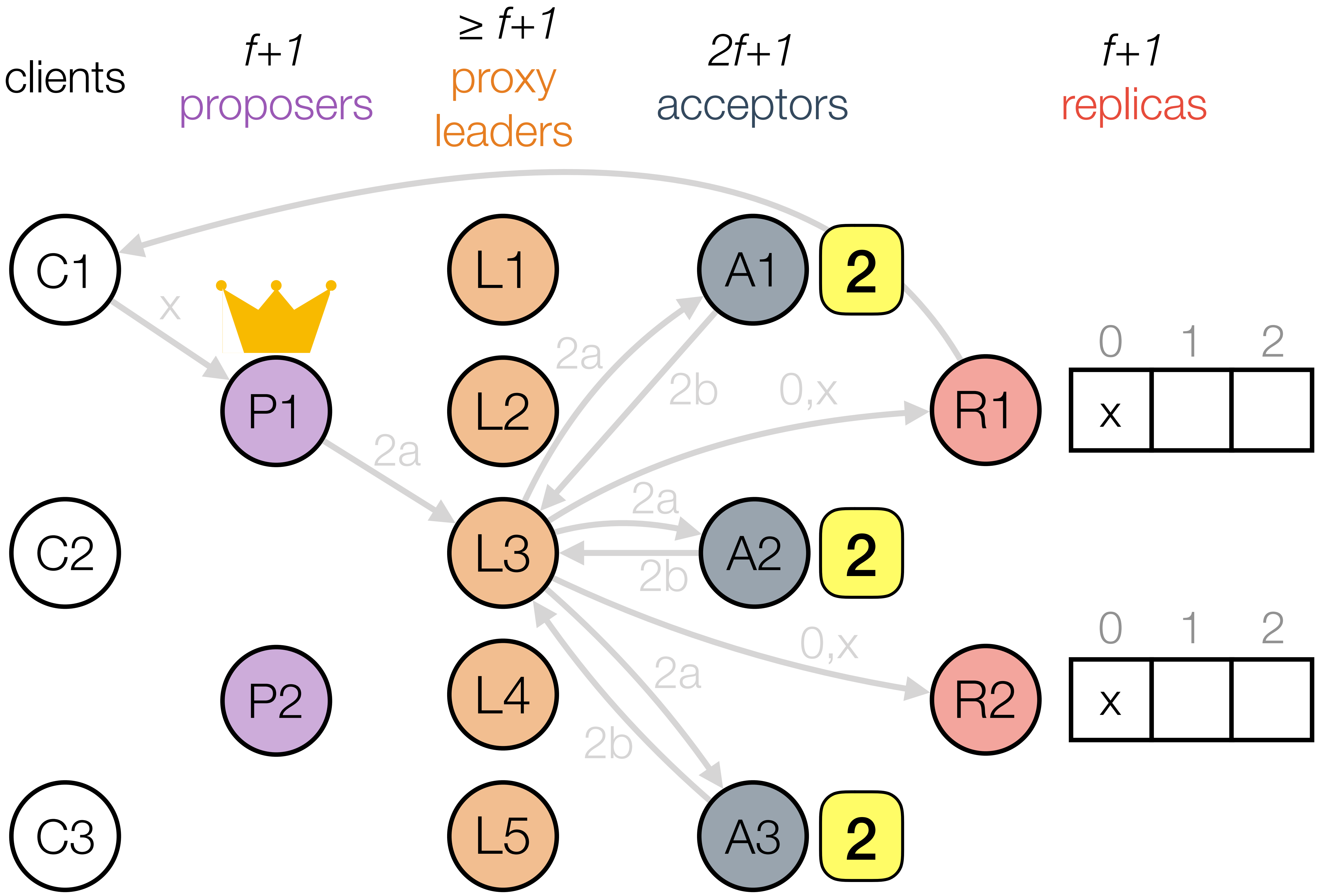
clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
R2
y
1,y
1,y
0
1
2
x
y
0
1
2
x
y


clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
R2
0
1
2
x
y
z
0
1
2
x
y
z

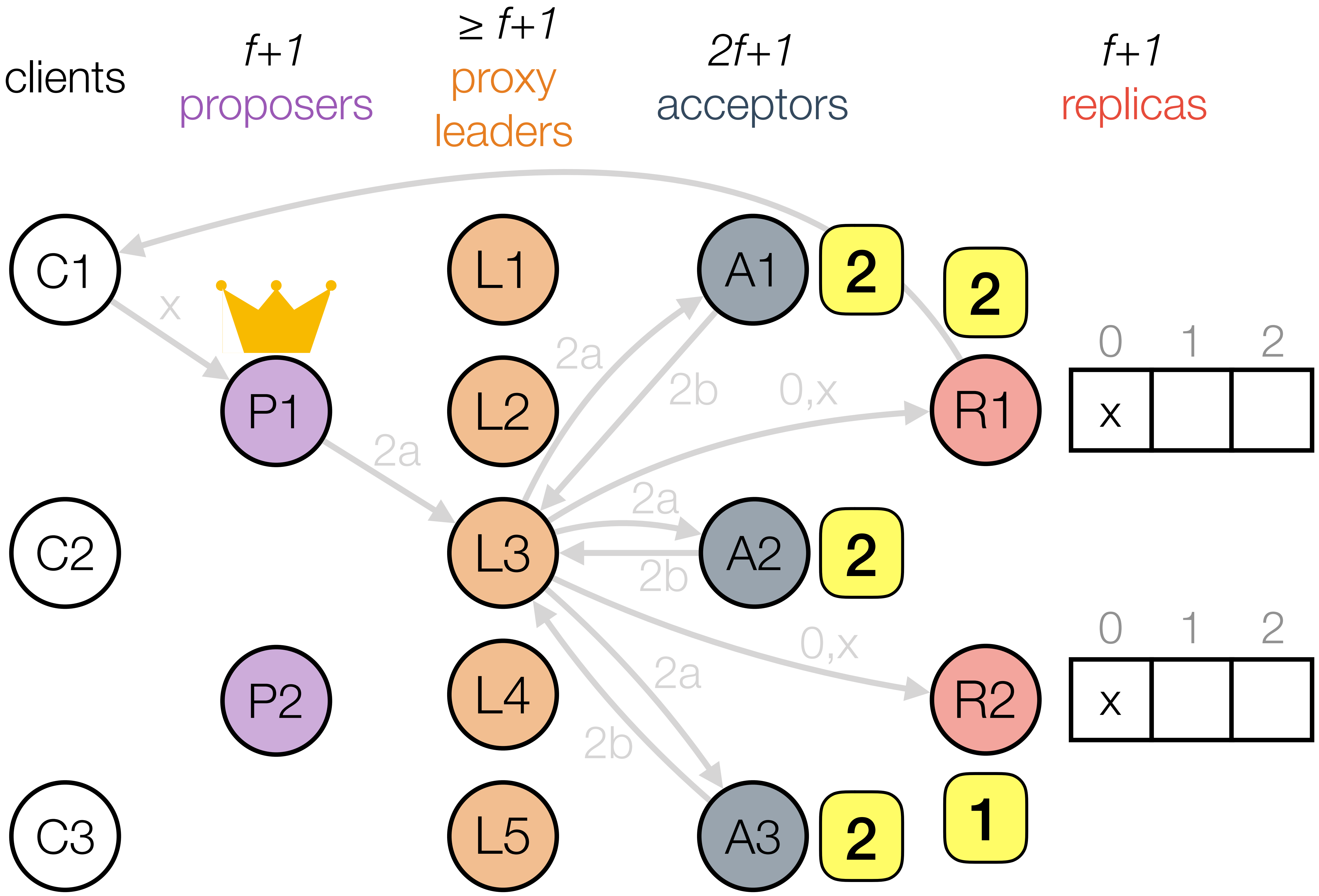


clients
f+1
proposers
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
5f+4
2a
2b
A1
A2
A3
2
2
2
0,x
0,x
R1
R2
2
1
0 1 2
x
0 1 2
x

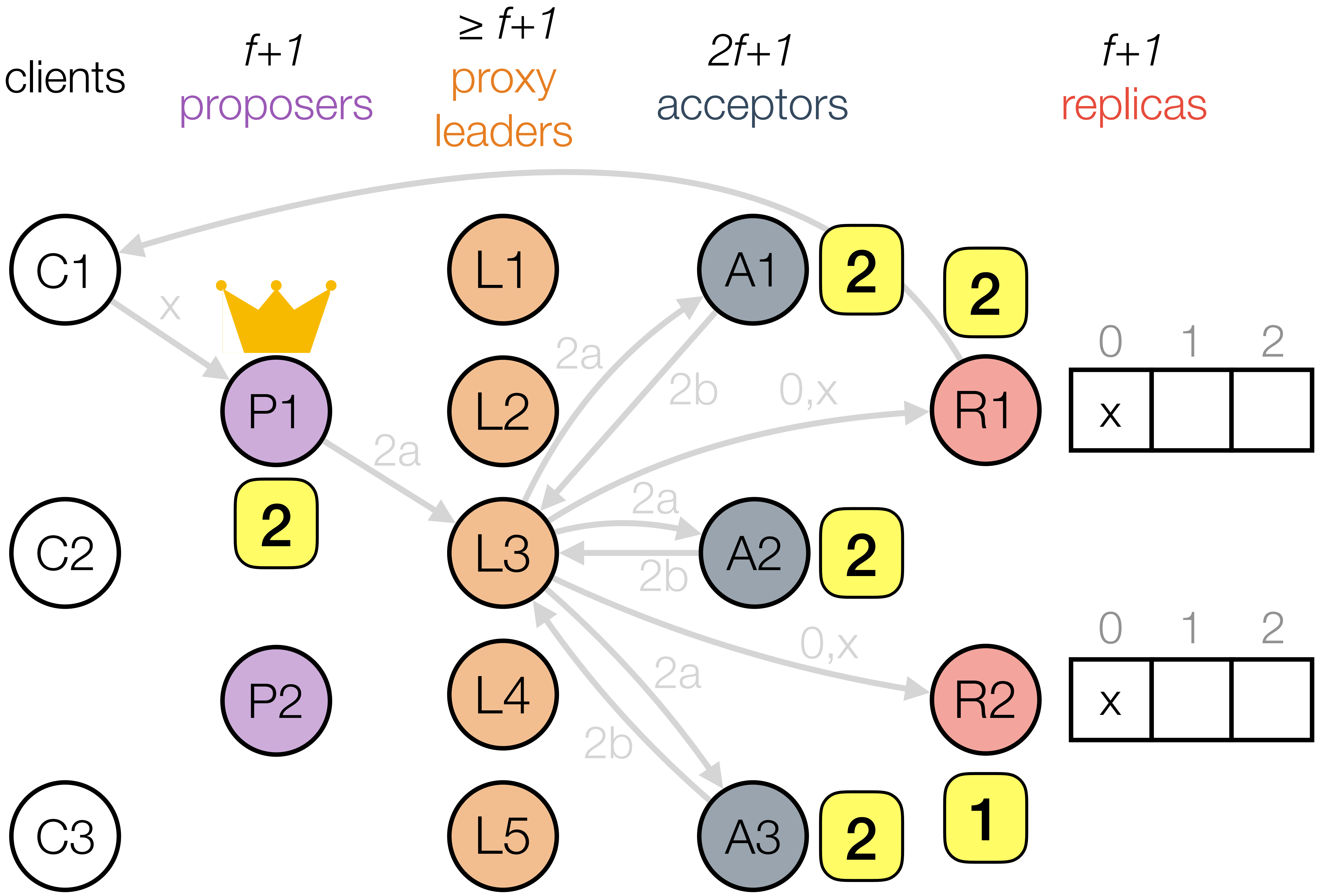

clients
f+1
proposers
≥ *f*+1
proxy
leaders
2*f*+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
2a
2a
2b
2a
2b
2a
2b
A1
A2
A3
0,x
0,x
R1
R2
0
1
2
x
0
1
2
x

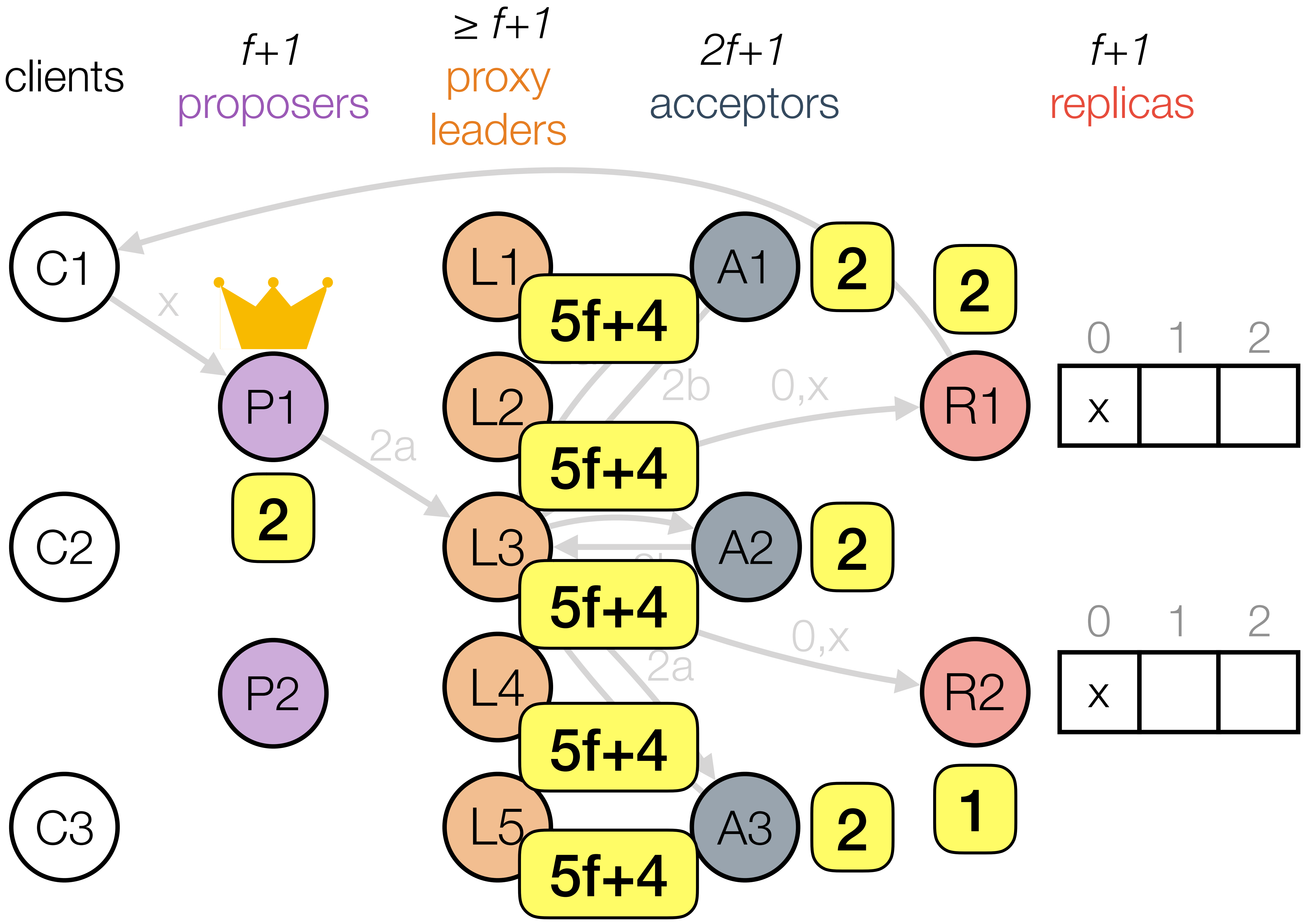


clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
2a
2a
2b
2a
2b
2a
2b
A1
A2
A3
2
2
2
0,x
0,x
R1
R2
0
1
2
x
0
1
2
x



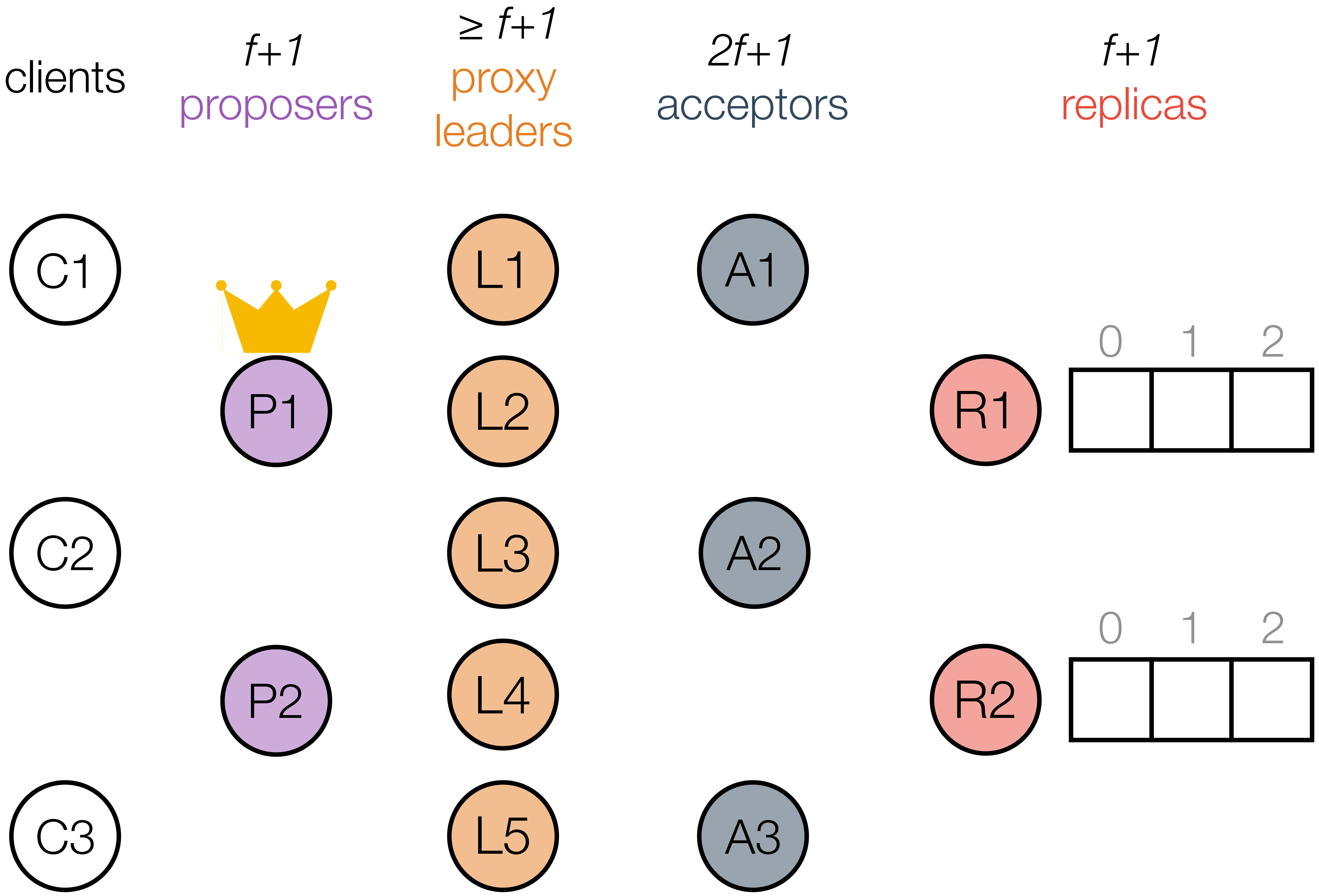
clients
f+1
proposers
≥ f+1
proxy leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
2a
2a
2b
0,x
2a
2b
2a
2b
0,x
A1
A2
A3
2
2
2
2
1
R1
R2
0
1
2
x
0
1
2
x

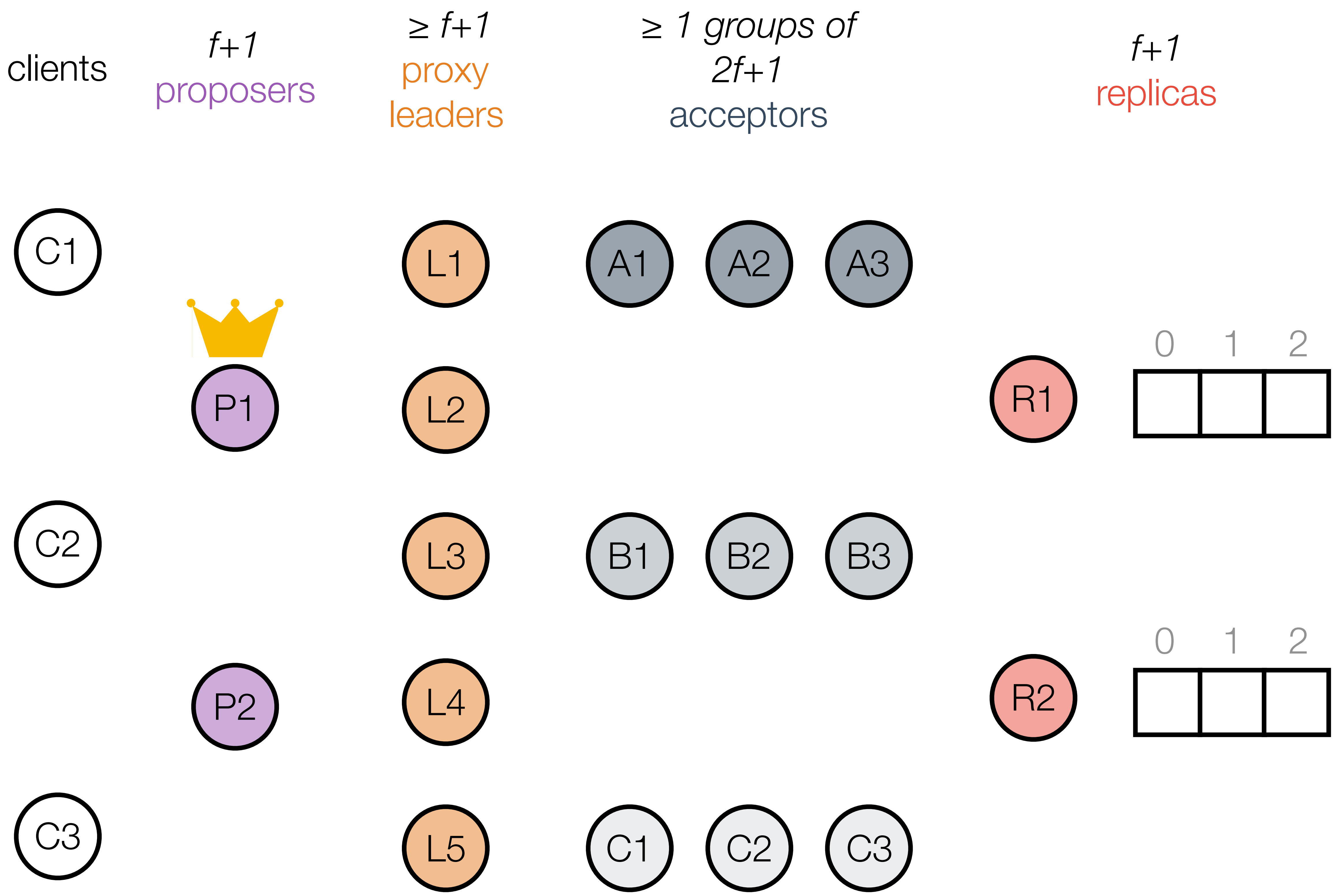

clients
f+1
proposers
≥ f+1
proxy
leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
2
L1
L2
L3
L4
L5
2a
2a
2b
0,x
2a
2b
2a
2b
0,x
A1
A2
A3
2
2
2
R1
R2
2
1
0 1 2
x
0 1 2
x



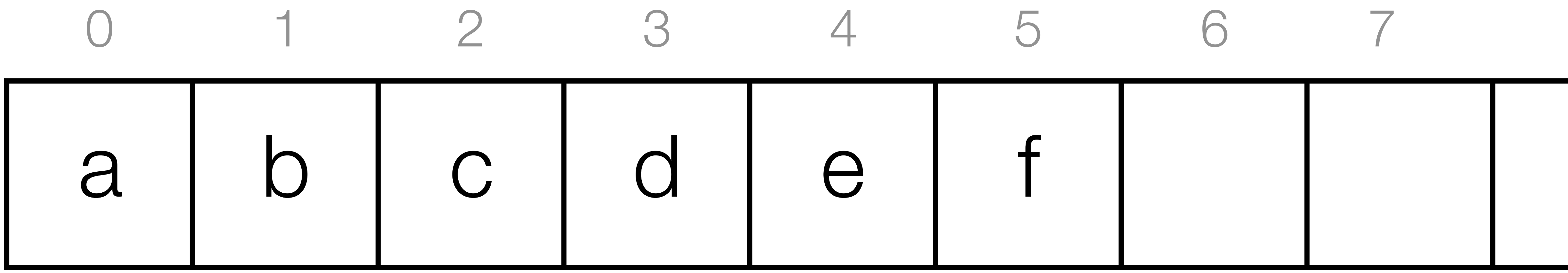
clients
f+1
proposers
≥ f+1
proxy leaders
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
2
2a
L1
L2
L3
L4
L5
5f+4
5f+4
5f+4
5f+4
5f+4
2b
2a
A1
A2
A3
2
2
2
0,x
0,x
R1
R2
2
1
0 1 2
x
0 1 2
x

Compartmentalization:
Decouple and scale

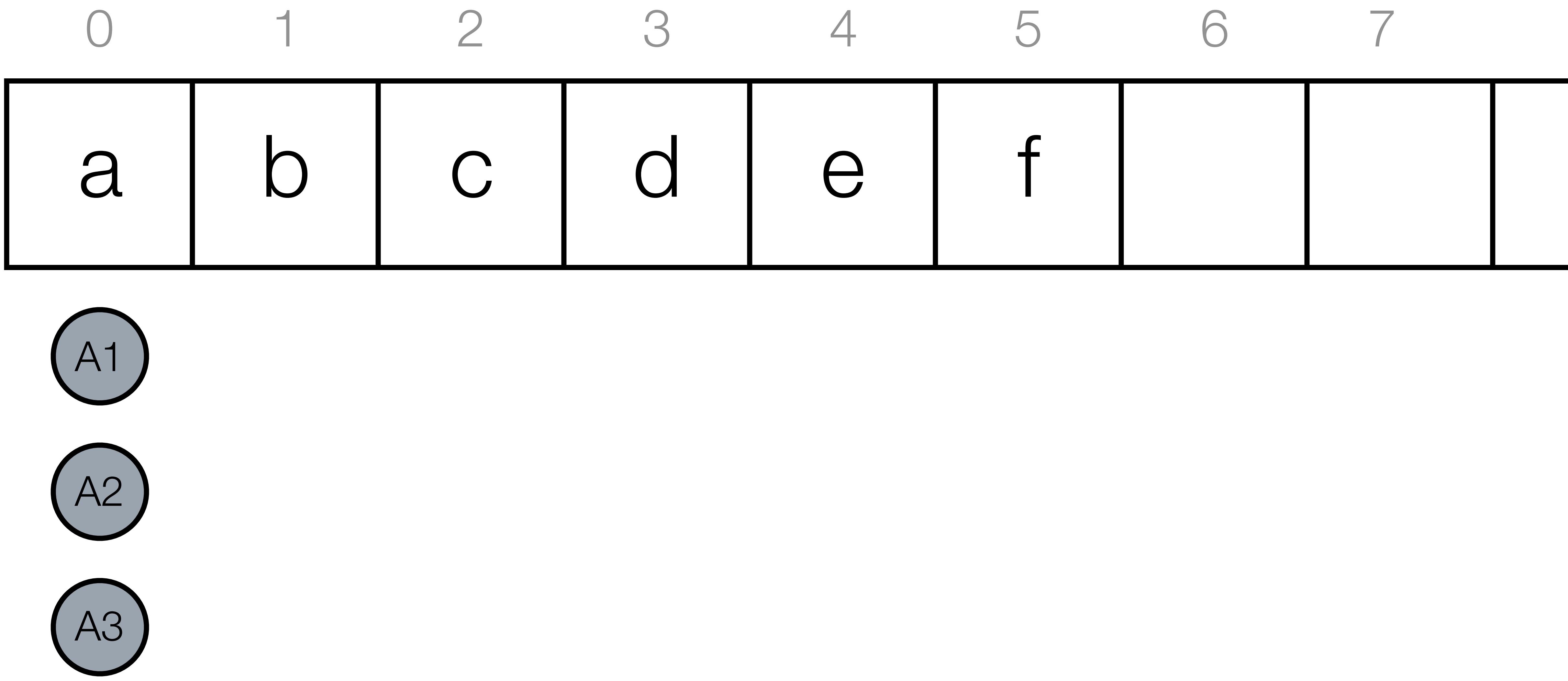

clients
f+1
proposers
≥ *f*+1
proxy
leaders
2*f*+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
R1
0 1 2
R2
0 1 2


clients
$f+1$ proposers
$\geq f+1$ proxy leaders
≥ 1 groups of $2f+1$ acceptors
$f+1$ replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
0
1
2

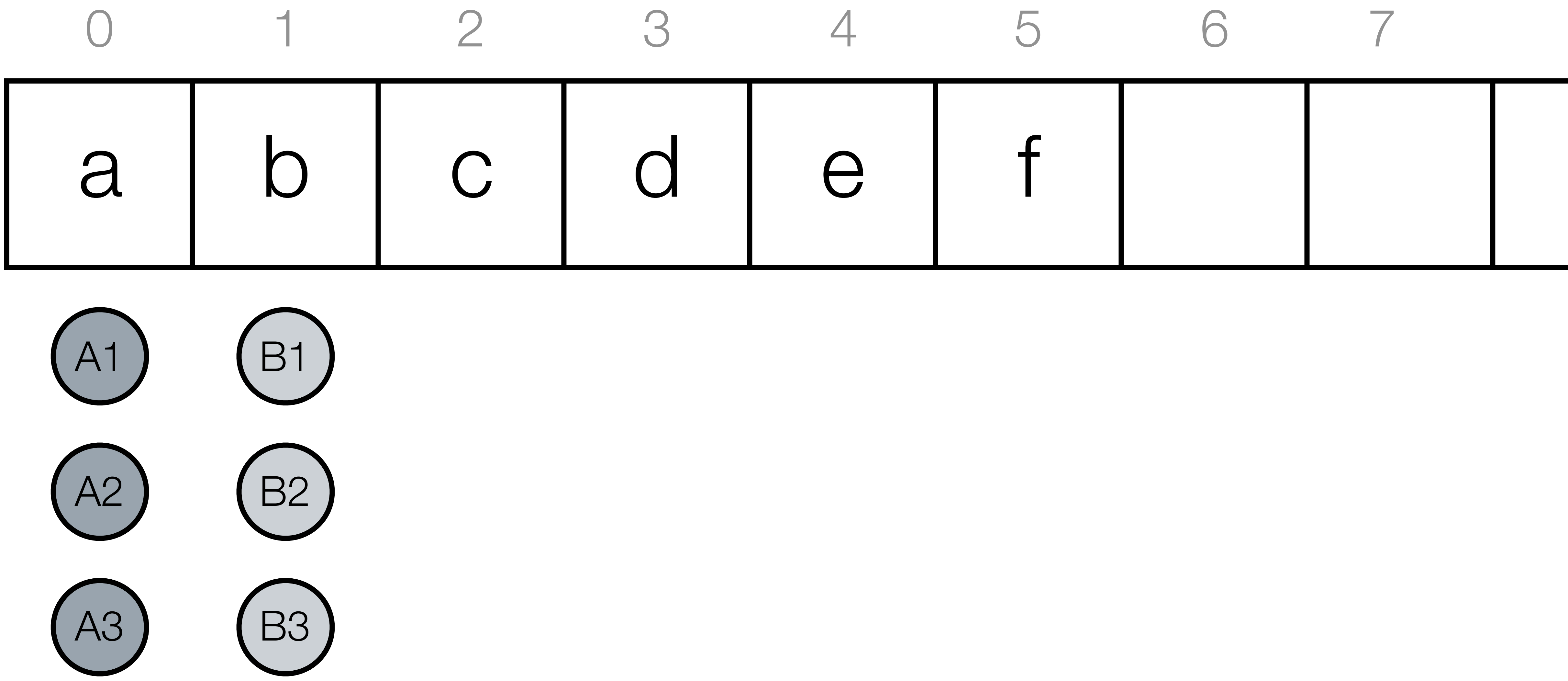
Acceptors



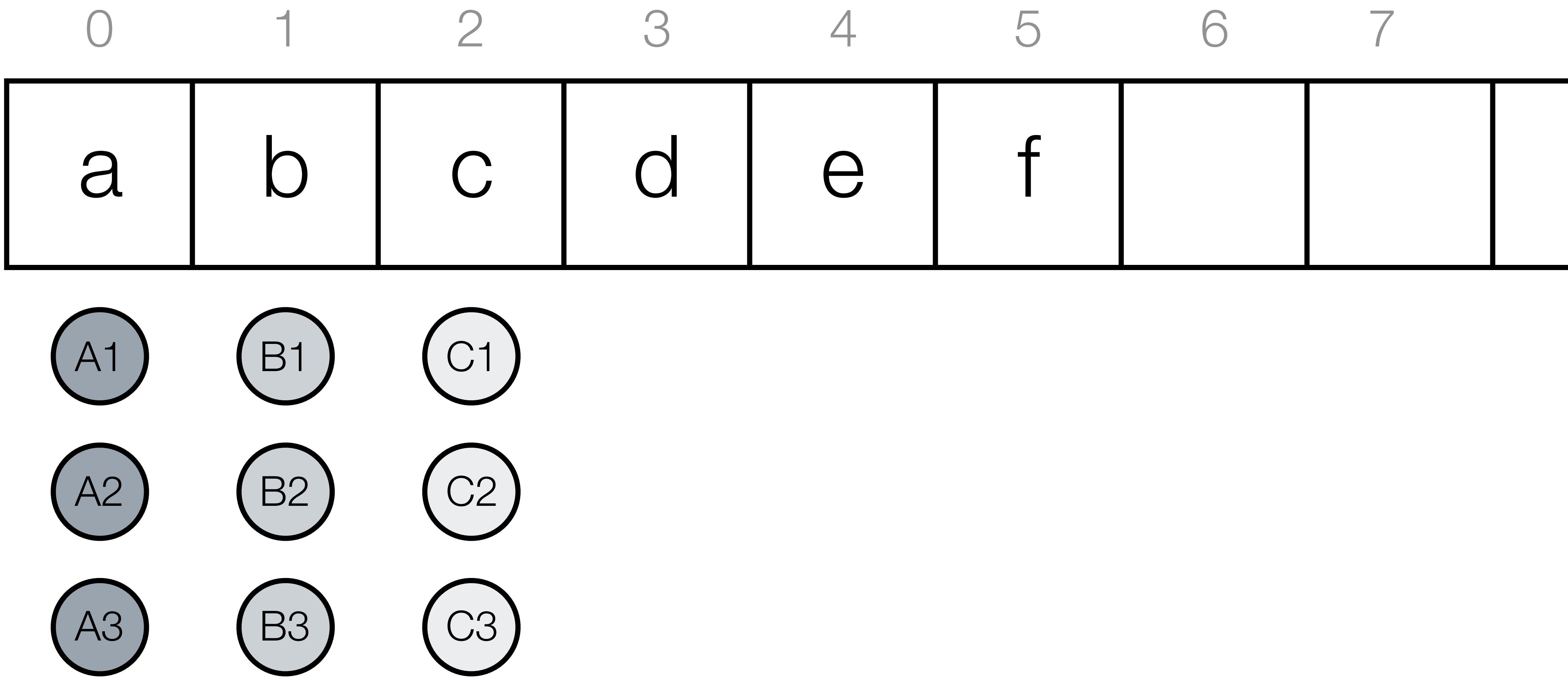
Acceptors



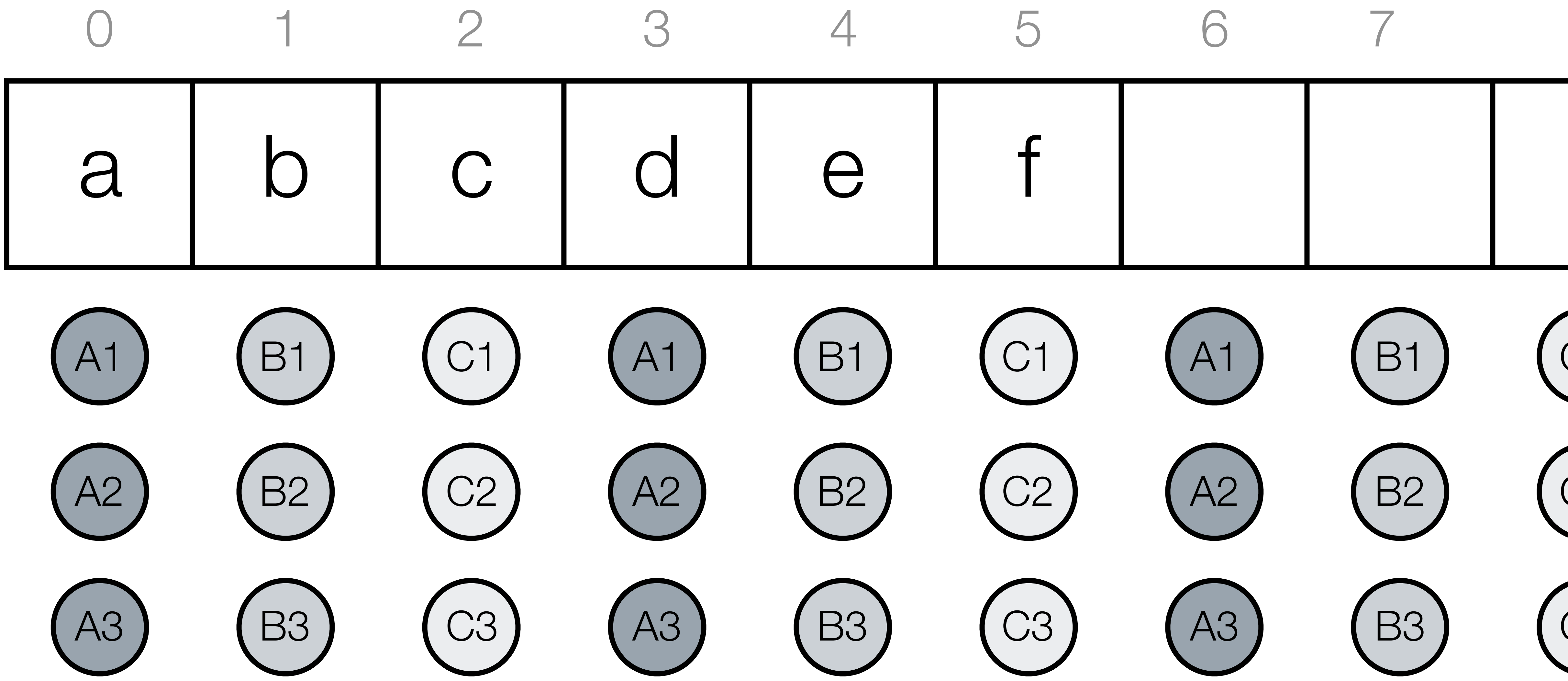
Acceptors

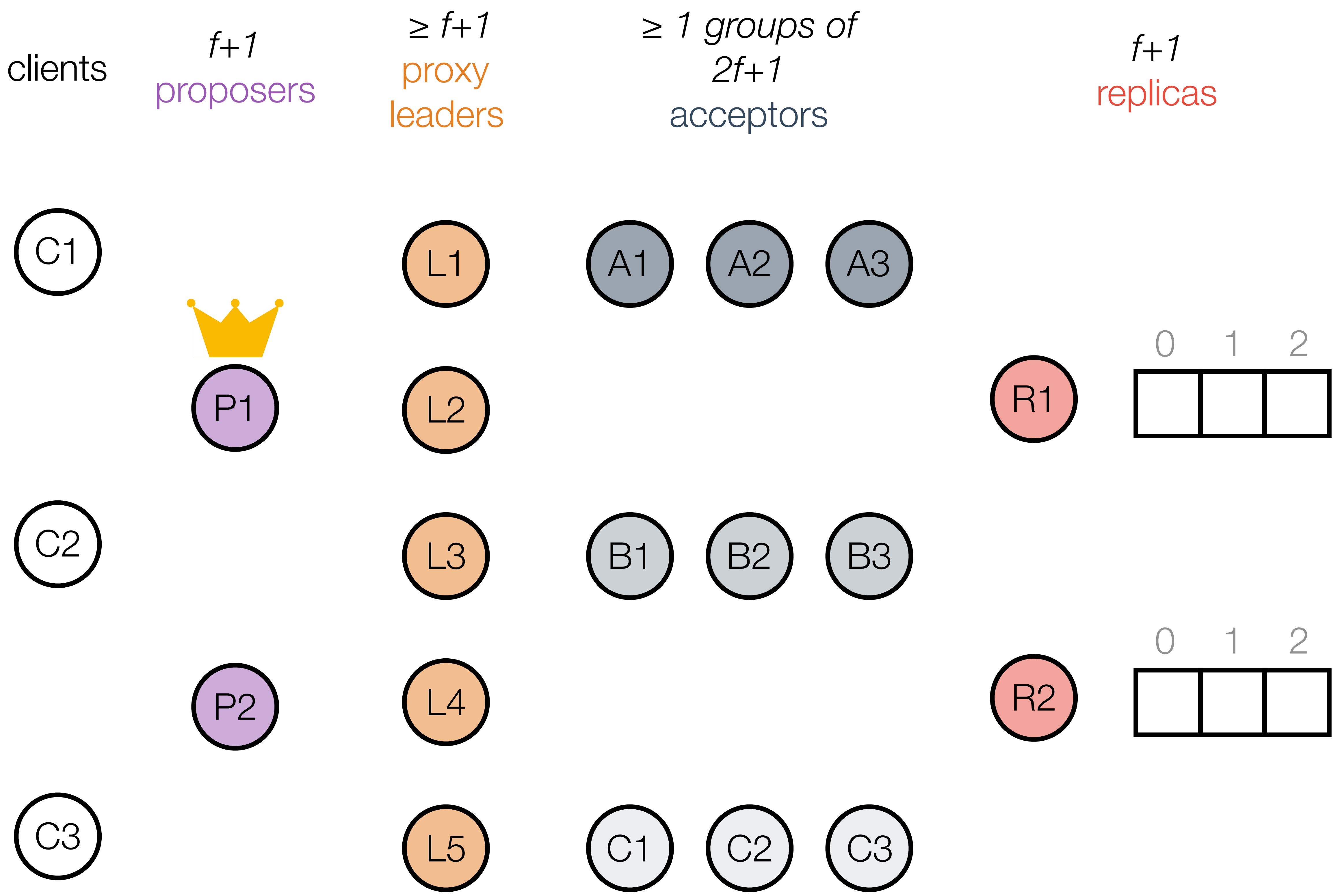


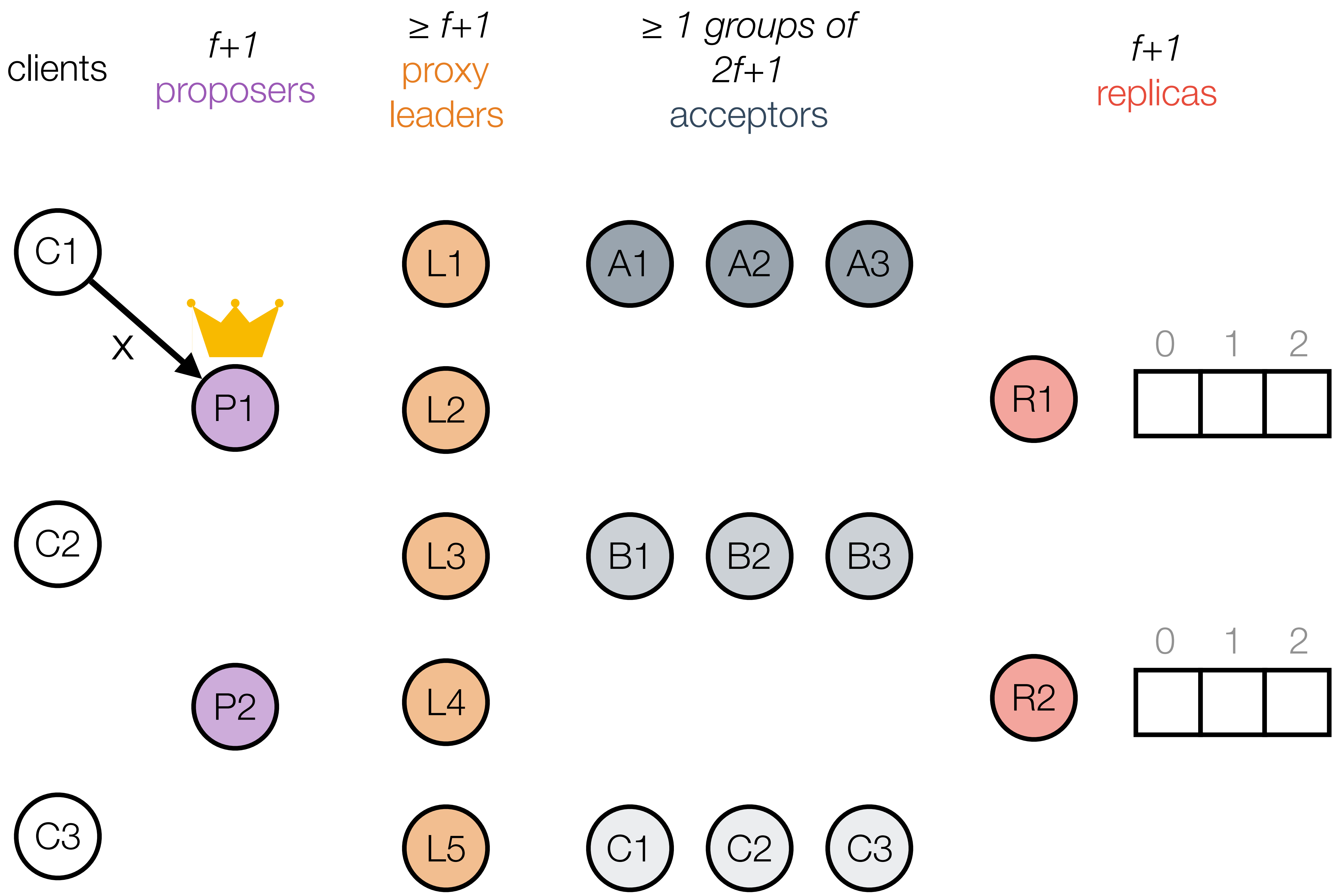
Acceptors

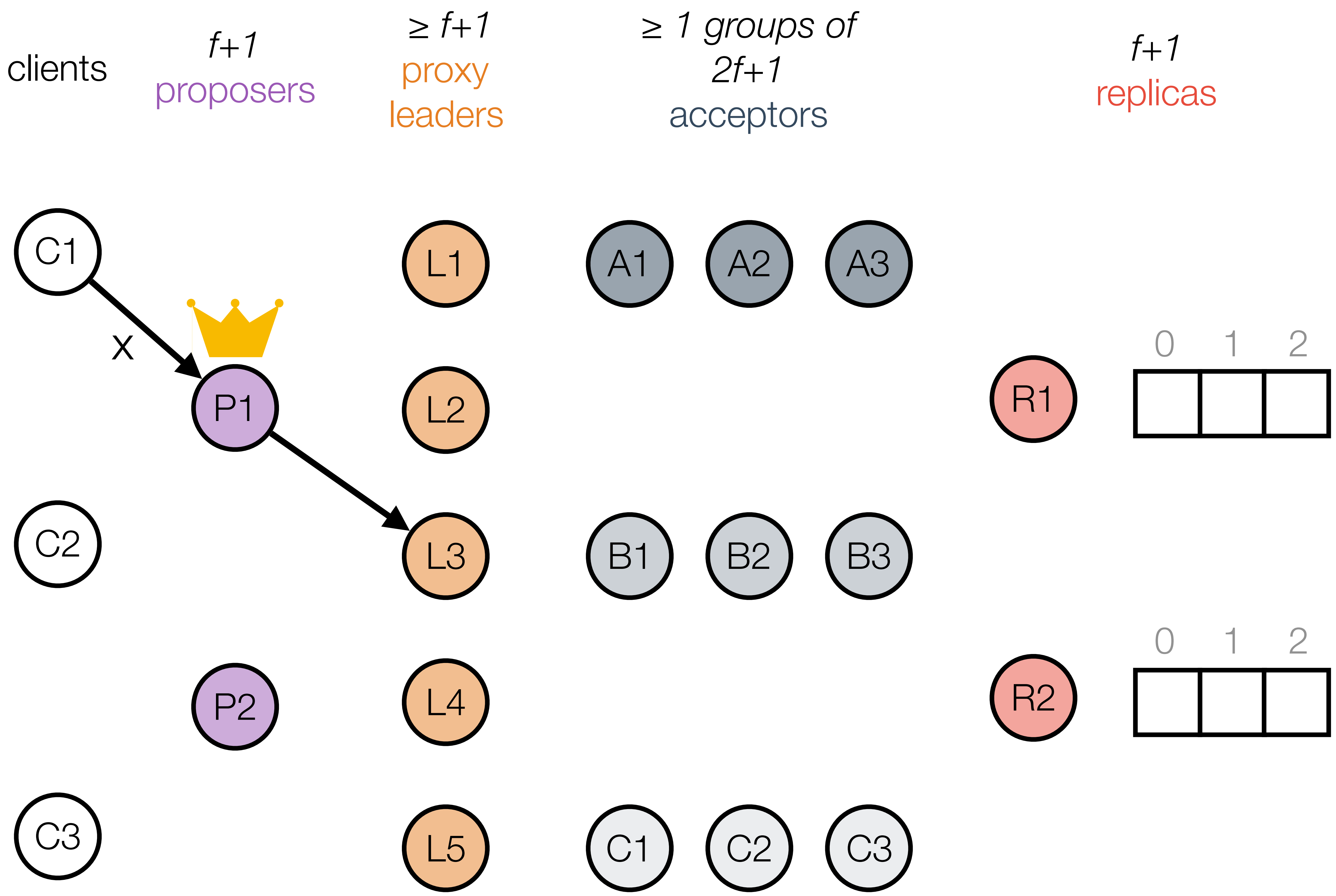


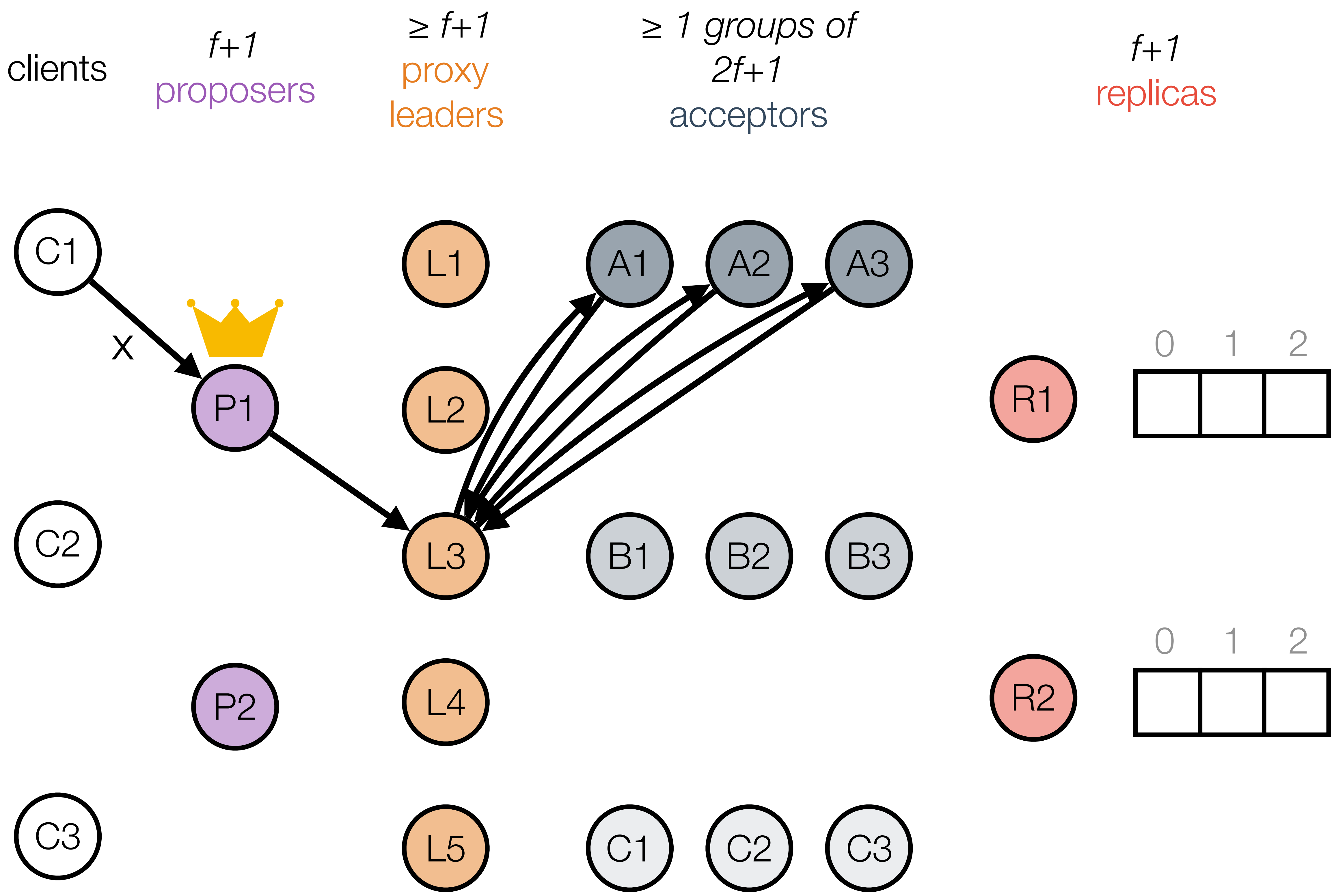
Acceptors

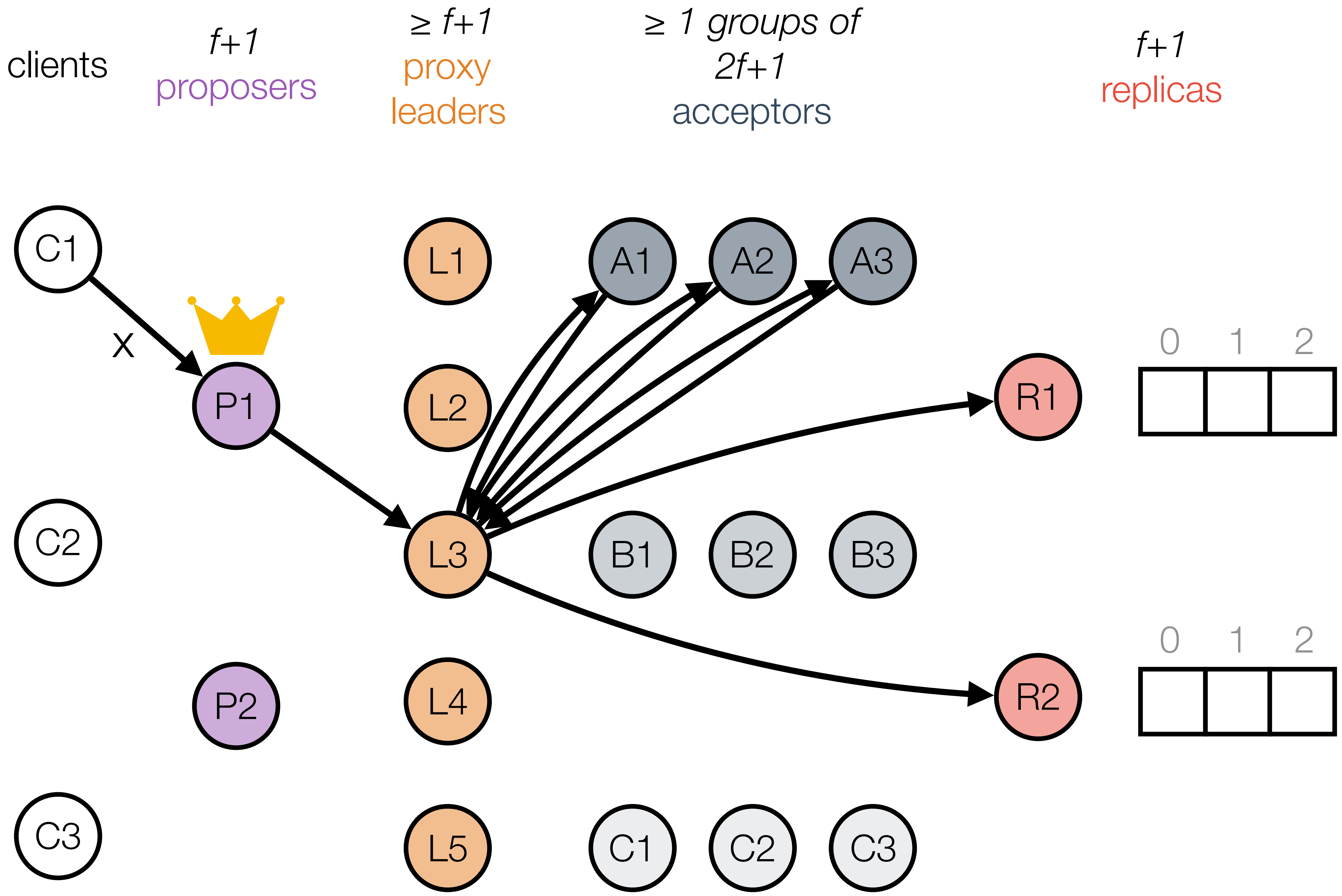

0
1
2
3
4
5
6
7
a
b
c
d
e
f
A1
B1
C1
A1
B1
C1
A1
B1
A2
B2
C2
A2
B2
C2
A2
B2
A3
B3
C3
A3
B3
C3
A3
B3

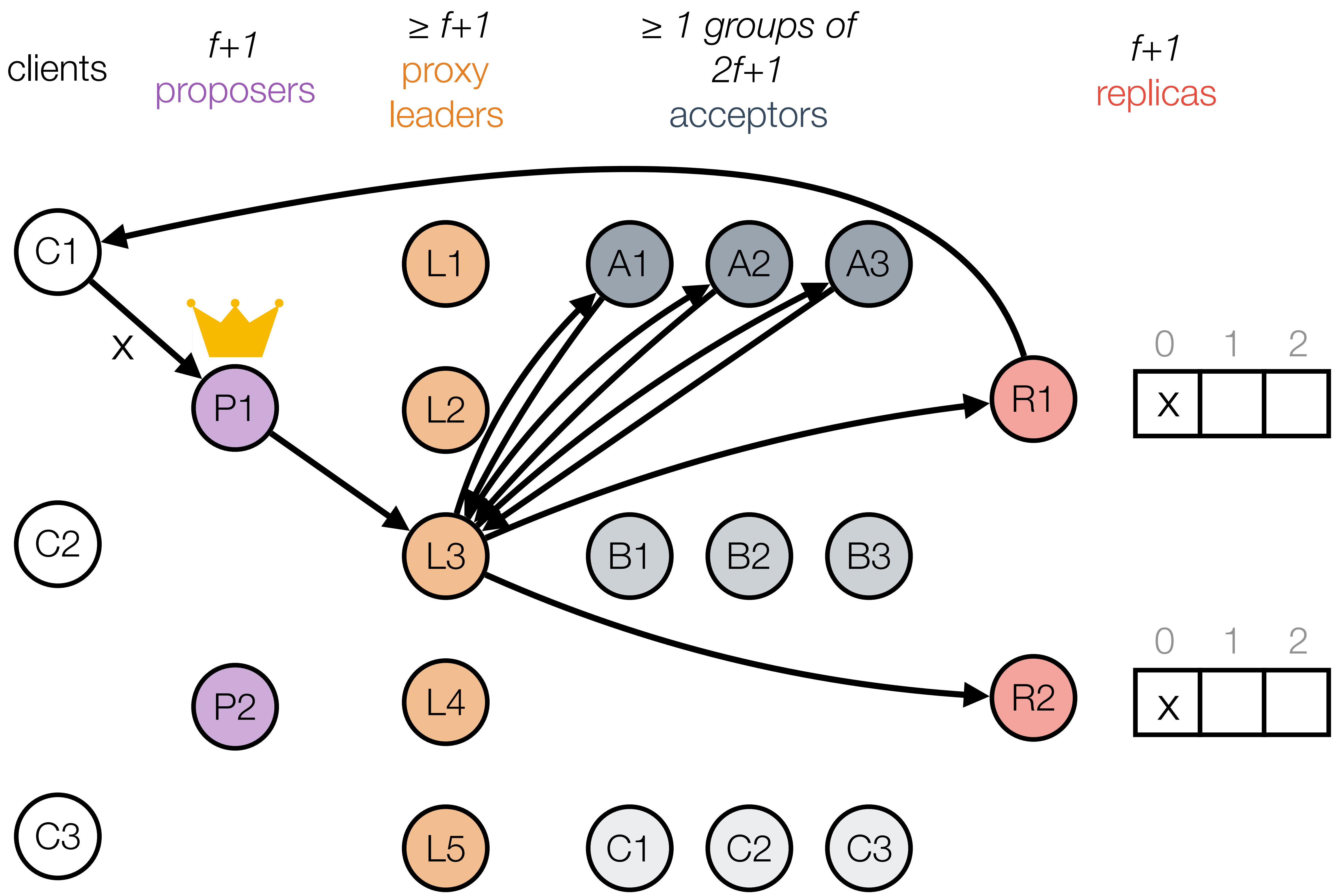

clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
0
1
2

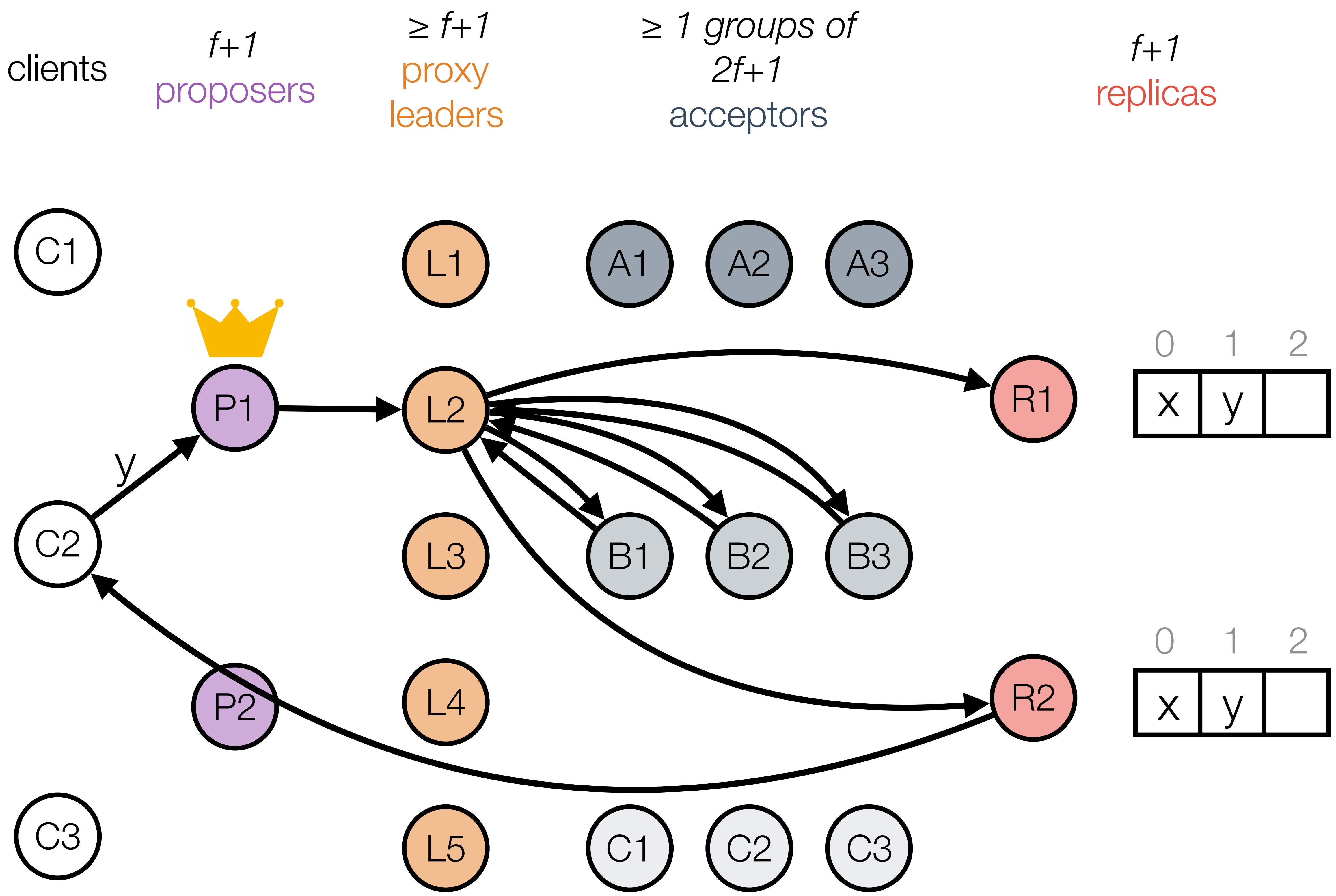

clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
0
1
2

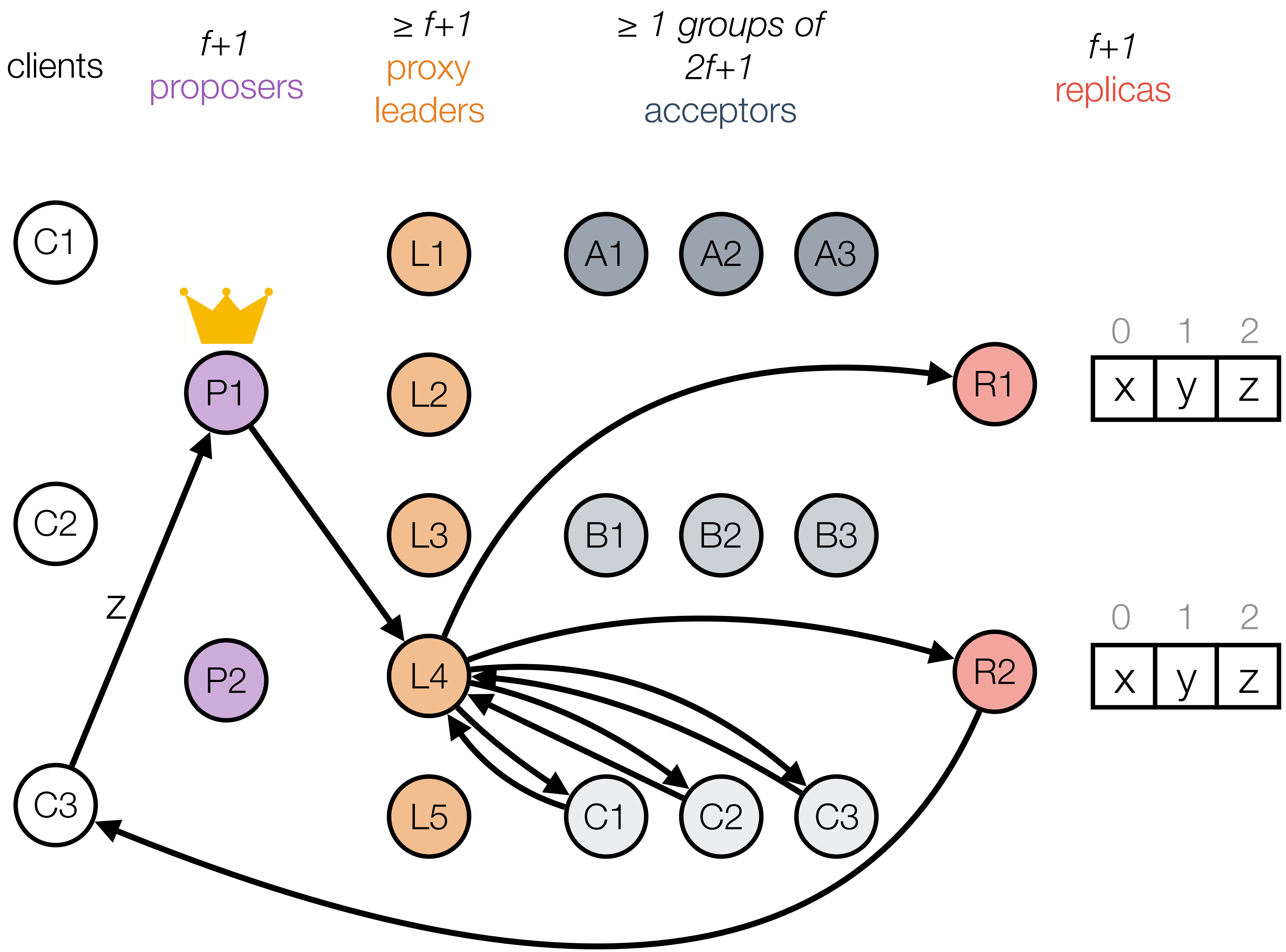

clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
0
1
2


clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
x
P1
P2
C2
C3
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
0 1 2
R2
0 1 2


clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
x
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0 1 2
0 1 2

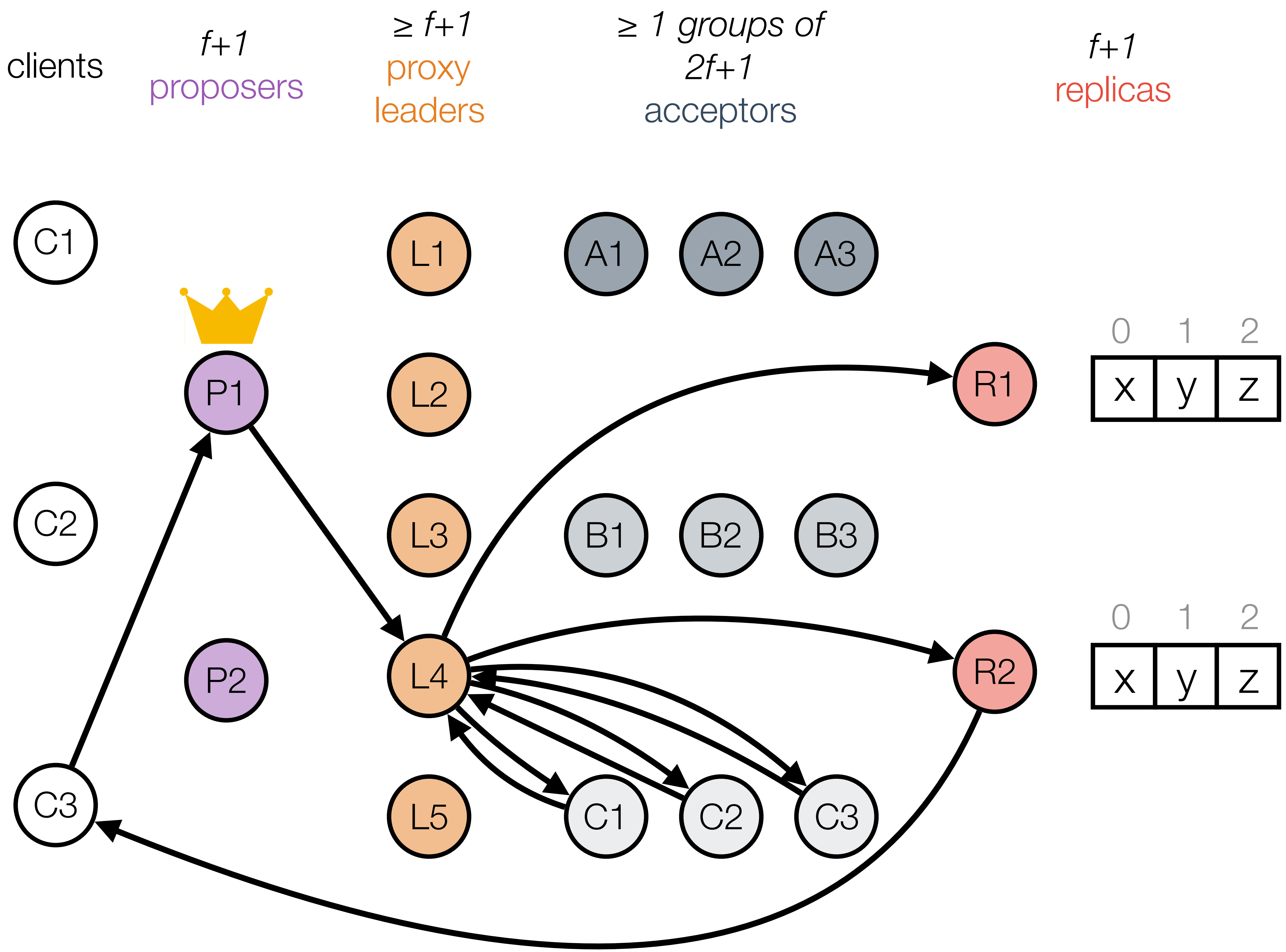

clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
x
P1
L1
L2
L3
A1
A2
A3
B1
B2
B3
R1
0 1 2
x
C2
P2
L4
R2
0 1 2
x
C3
L5
C1
C2
C3


clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
y
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
x
y
0
1
2
x
y

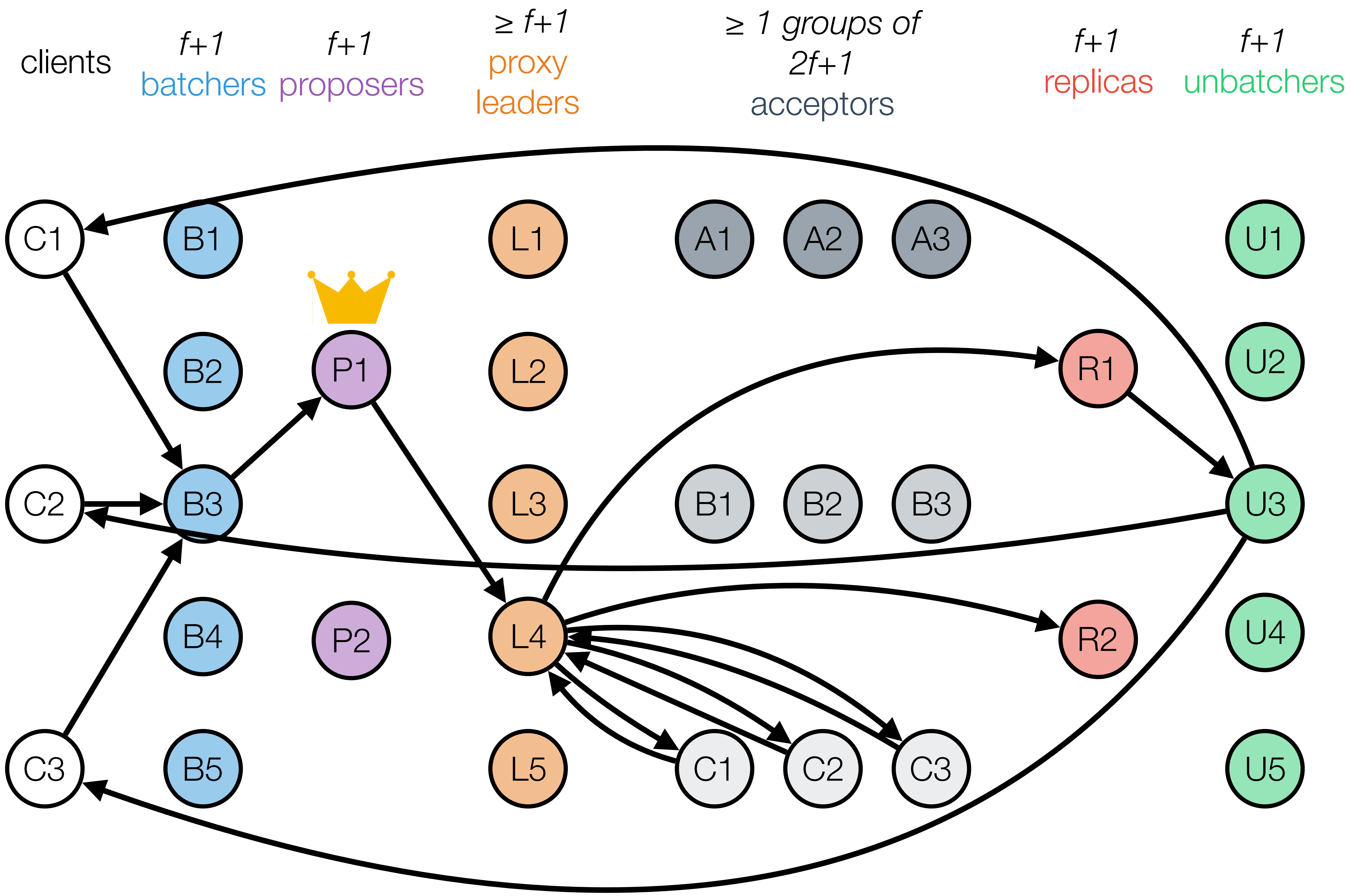

clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
z
0 1 2
x y z
0 1 2
x y z

"Using more than $2f$+1 [acceptors] for f failures is possible but illogical because it requires a larger quorum size with no additional benefit"

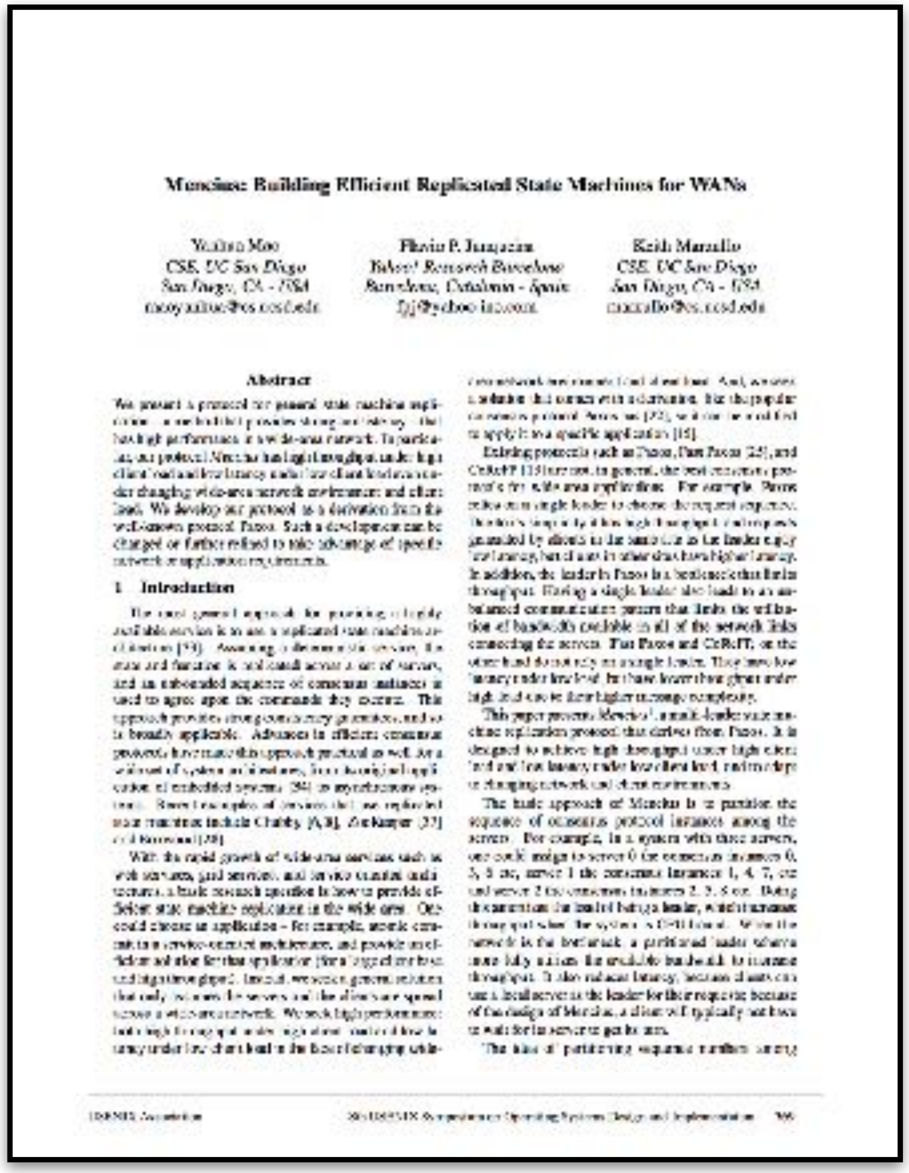
-TAPIR (SOSP 2015)



clients
f+1
proposers
≥ f+1
proxy
leaders
≥ 1 groups of
2f+1
acceptors
f+1
replicas
C1
C2
C3
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
0
1
2
x
y
z
0
1
2
x
y
z


clients
f+1
batchers
f+1
proposers
≥ f+1
proxy leaders
≥ 1 groups of 2f+1
acceptors
f+1
replicas
f+1
unbatchers
C1
C2
C3
B1
B2
B3
B4
B5
P1
P2
L1
L2
L3
L4
L5
A1
A2
A3
B1
B2
B3
C1
C2
C3
R1
R2
U1
U2
U3
U4
U5

Mencius: round robin partition the log between multiple leaders



Mencius: Building Efficient Replicated State Machines for WANs

Yanhua Mao
CSE, UC San Diego
San Diego, CA - USA
maoyanhua@cs.ucsd.edu

Flavio P. Junqueira
Yahoo! Research Barcelona
Barcelona, Catalonia - Spain
fpj@yahoo-inc.com

Keith Marzullo
CSE, UC San Diego
San Diego, CA - USA
marzullo@cs.ucsd.edu

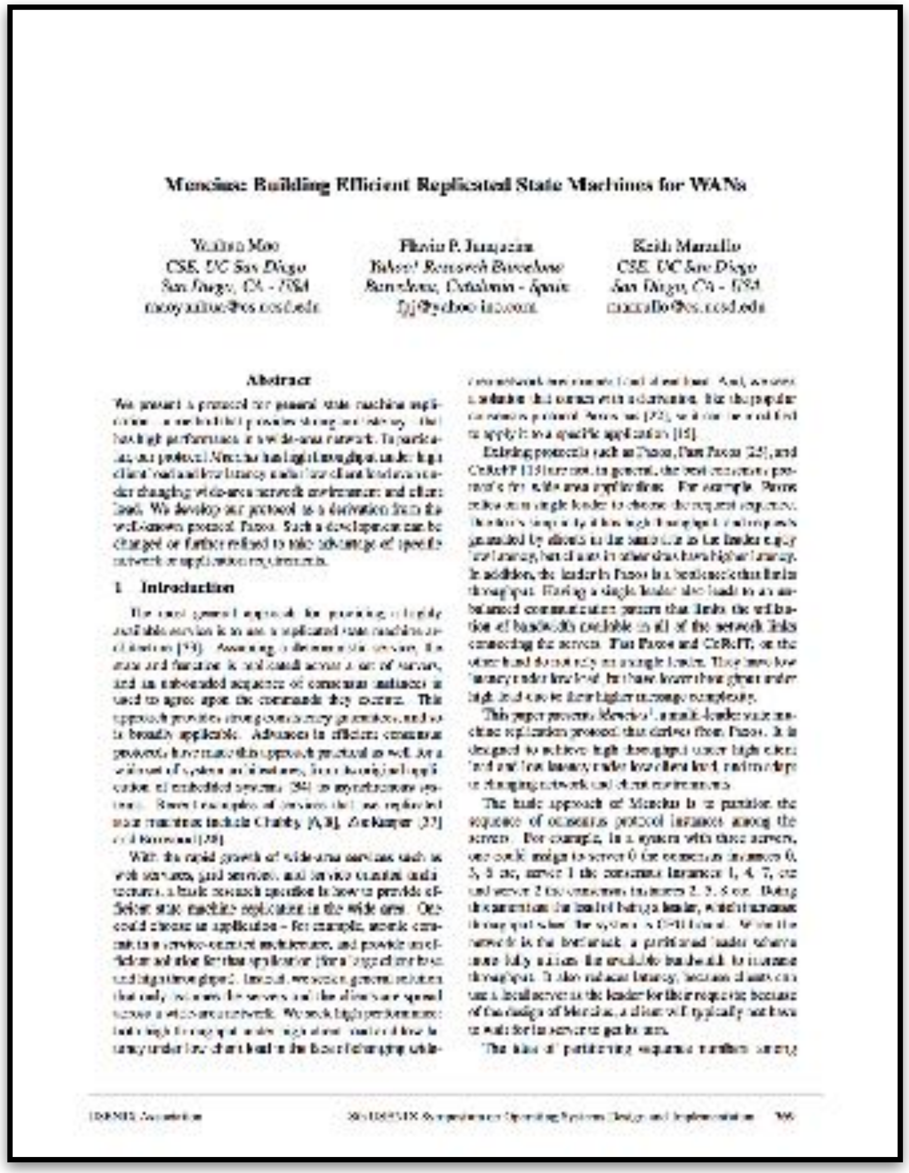
Abstract

[illegible]

1 Introduction

[illegible]

Mencius: round robin partition the log between multiple leaders


Mencius: Building Efficient Replicated State Machines for WANs

Yanhua Mao
CSE, UC San Diego
San Diego, CA - USA
maoyanhua@cs.ucsd.edu

Flavio P. Junqueira
Yahoo! Research Barcelona
Barcelona, Catalonia - Spain
fpj@yahoo-inc.com

Keith Marzullo
CSE, UC San Diego
San Diego, CA - USA
marzullo@cs.ucsd.edu

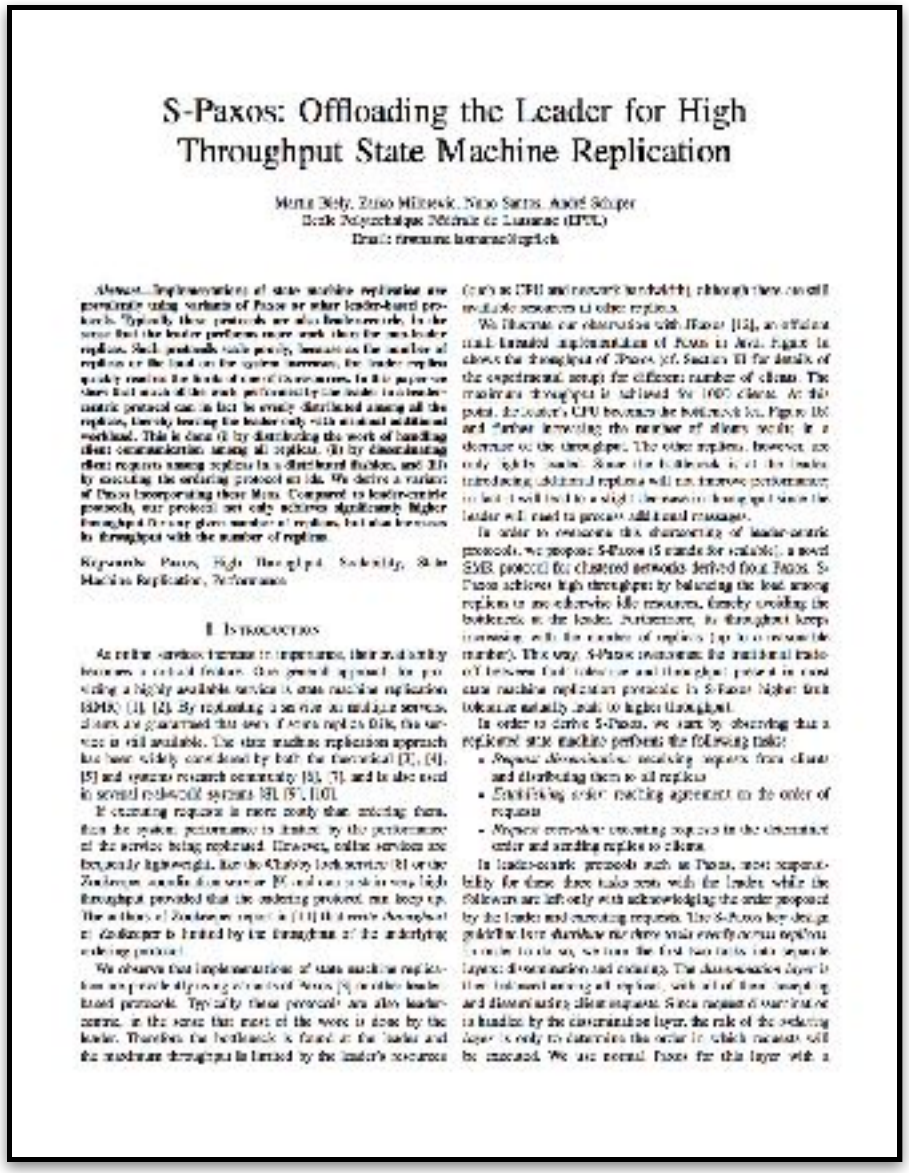
Abstract

[illegible]

1 Introduction

[illegible]

S-Paxos: decouple data flow from control flow

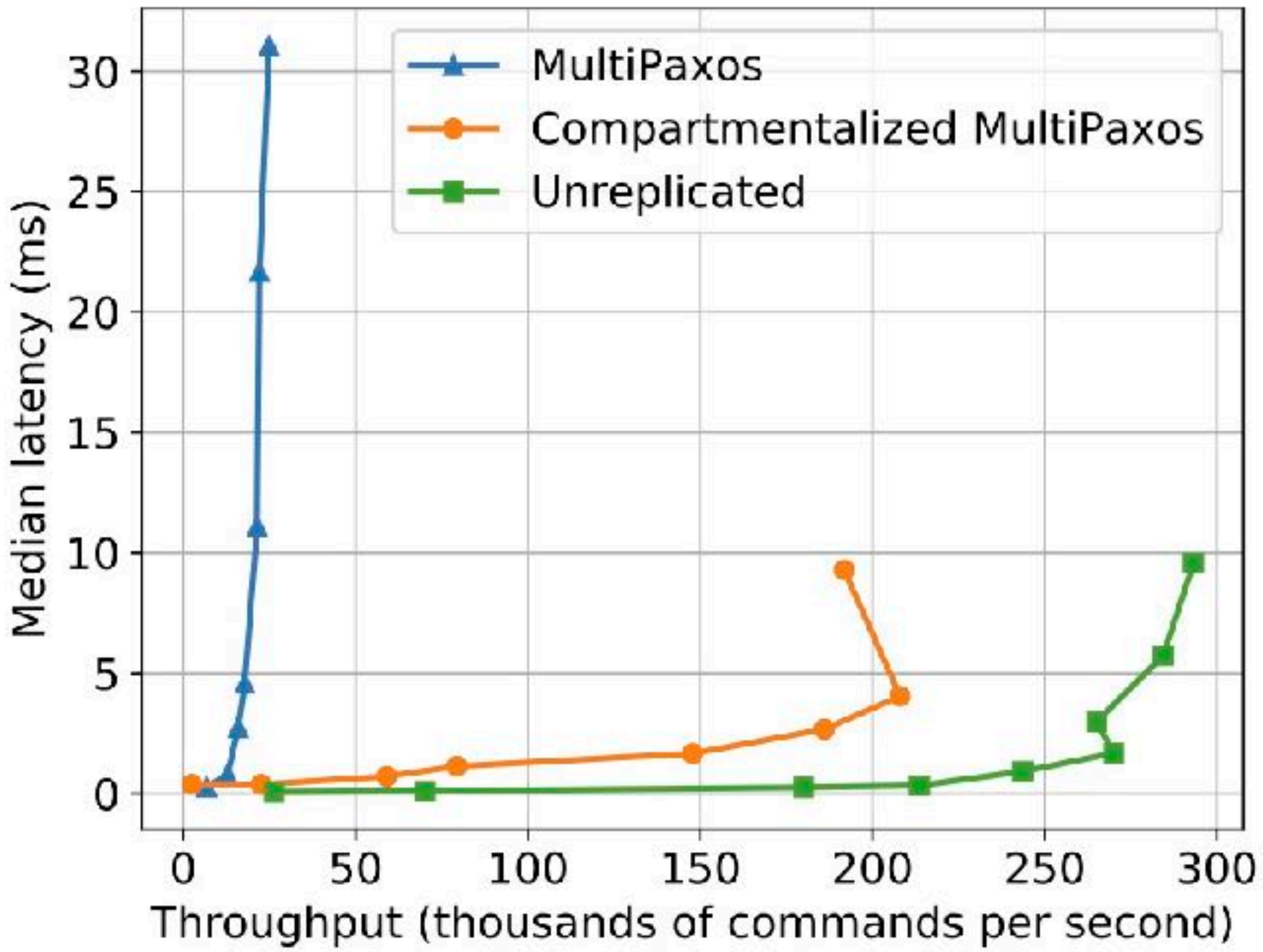

S-Paxos: Offloading the Leader for High Throughput State Machine Replication

Martin Biely, Zarko Milosevic, Nuno Santos, André Schiper
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Email: [illegible]

[illegible]

I. Introduction

[illegible]


MultiPaxos
Compartmentalized MultiPaxos
Unreplicated
Median latency (ms)
Throughput (thousands of commands per second)
0
5
10
15
20
25
30
0
50
100
150
200
250
300

Some protocols **become the bottlenecks** they try to avoid!

mwhittaker.github.io

mwhittaker.github.io

mwhittaker.github.io/frankenpaxos

mwhittaker.github.io

mwhittaker.github.io/frankenpaxos

github.com/mwhittaker/frankenpaxos