RAFT algorithm & Copycat

2015-08-17

Mobile Convergence LAB,
Department of Computer Engineering,
Kyung Hee University.

Consensus Algorithm

각 노드의 상태에 의존하는 어떤 값이 서로 일치하게 됨

분산 컴퓨팅 분야에서

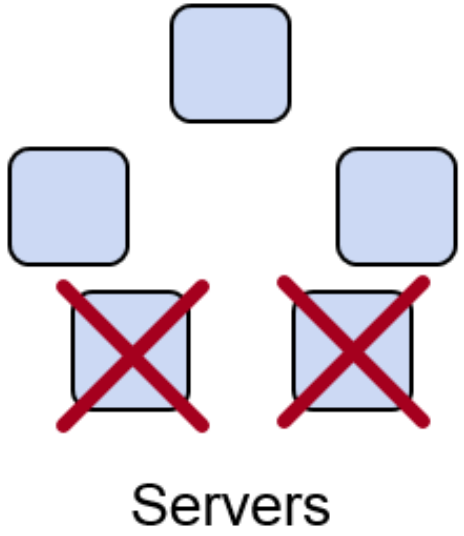
하나의 클러스터링 시스템에서 몇 개의 인스턴스가 오류가

발생하더라도 계속해서 서비스가 제공되도록 해주는 알고리즘

각 노드가 네트워크 상의 이웃 노드들과 관련 정보를 공유하는 상호 규칙

In Consensus…

- Minority of servers fail = No problem
- 정상 작동하는 서버가 과반수 이상이면 시스템은 정상 작동



- Key : Consistent storage system

Paxos

- 1989년도에 발표
- Consensus Algorithm을 구현한 대표적인 프로토콜
- 이해하기 너무 어렵고, 실제 구현하기 어렵다라는 단점이 존재

Why Raft?

- 너무 어려운 Paxos의 대안으로 주목받은 알고리즘
- 세분화되어 이해하기 쉽고, 구현하기 쉽게 개발(연구)됨

In Search of an Understandable Consensus Algorithm

Diego Ongaro and John Ousterhout, *Stanford University*

https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

This paper is included in the Proceedings of USENIX ATC '14:
2014 USENIX Annual Technical Conference.

June 19–20, 2014 • Philadelphia, PA

978-1-931971-10-2

Open access to the Proceedings of
USENIX ATC '14: 2014 USENIX Annual Technical
Conference is sponsored by USENIX.

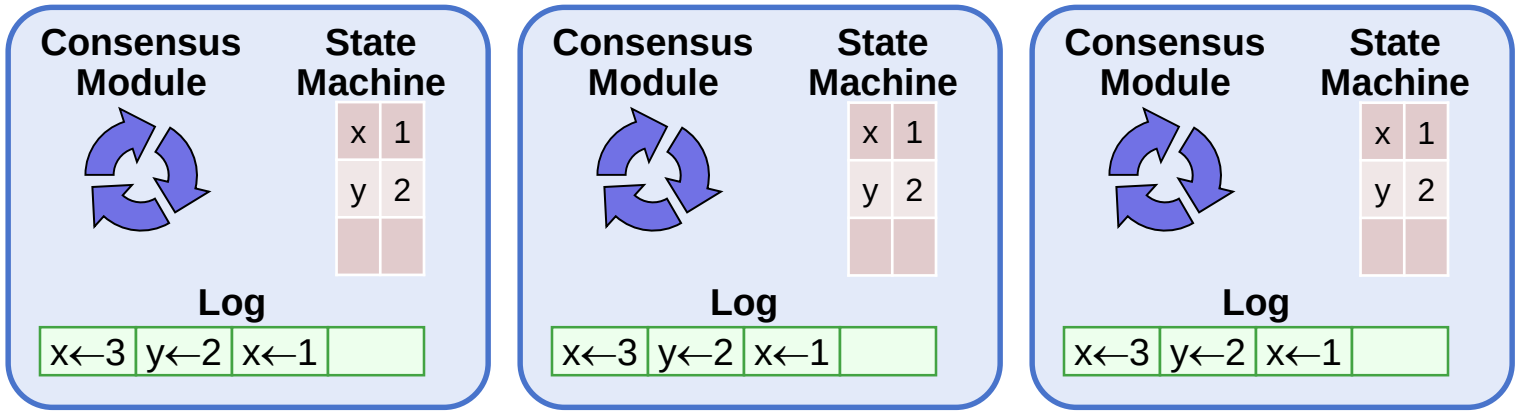
Raft

- "In Search of an Understandable Consensus Algorithm"
- Diego Ongaro & John Ousterhout in Stanford
- Best Paper Award at the 2014 USENIX Annual Technial Conference.
- 이후 박사 학위 논문에서 확장하여 발표
  (Consensus: Bridging Theory and Practice)

USENIX : Unix Users Group / 컴퓨터시스템 분야 세계 최고의 권위의 학술단체

Replicated State Machines (1)

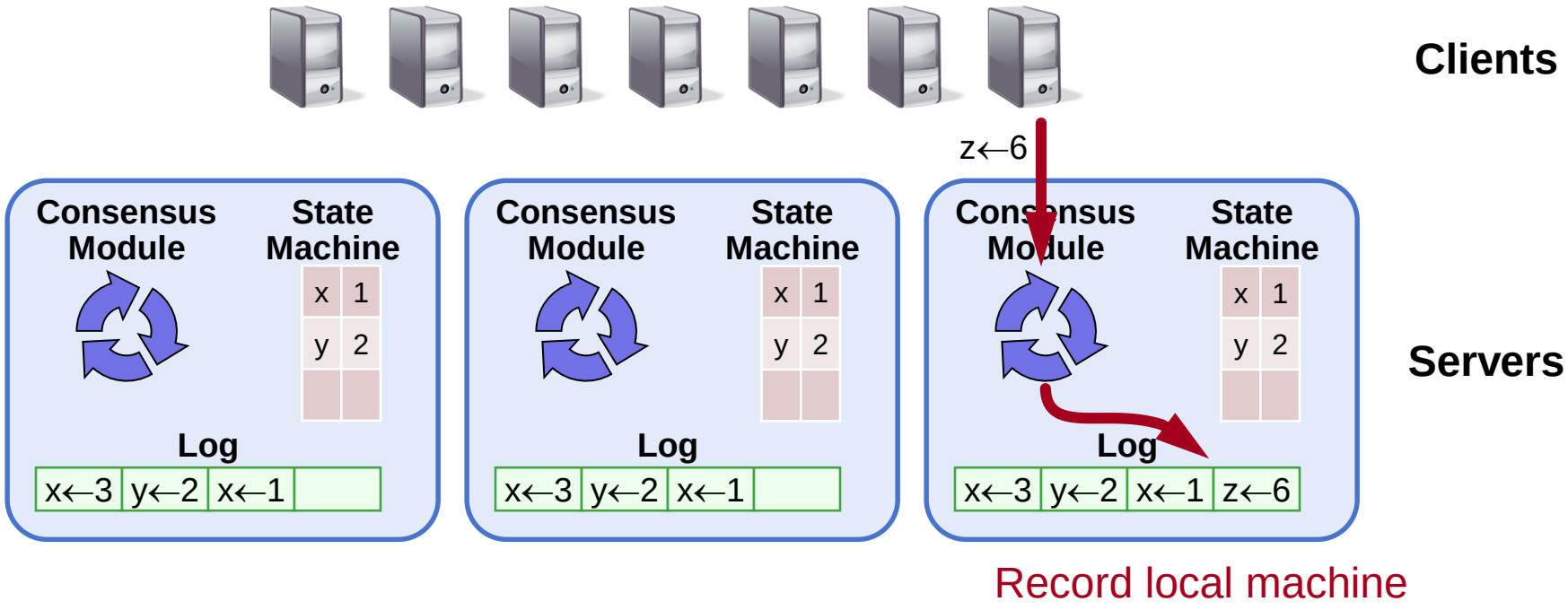
Clients



Servers

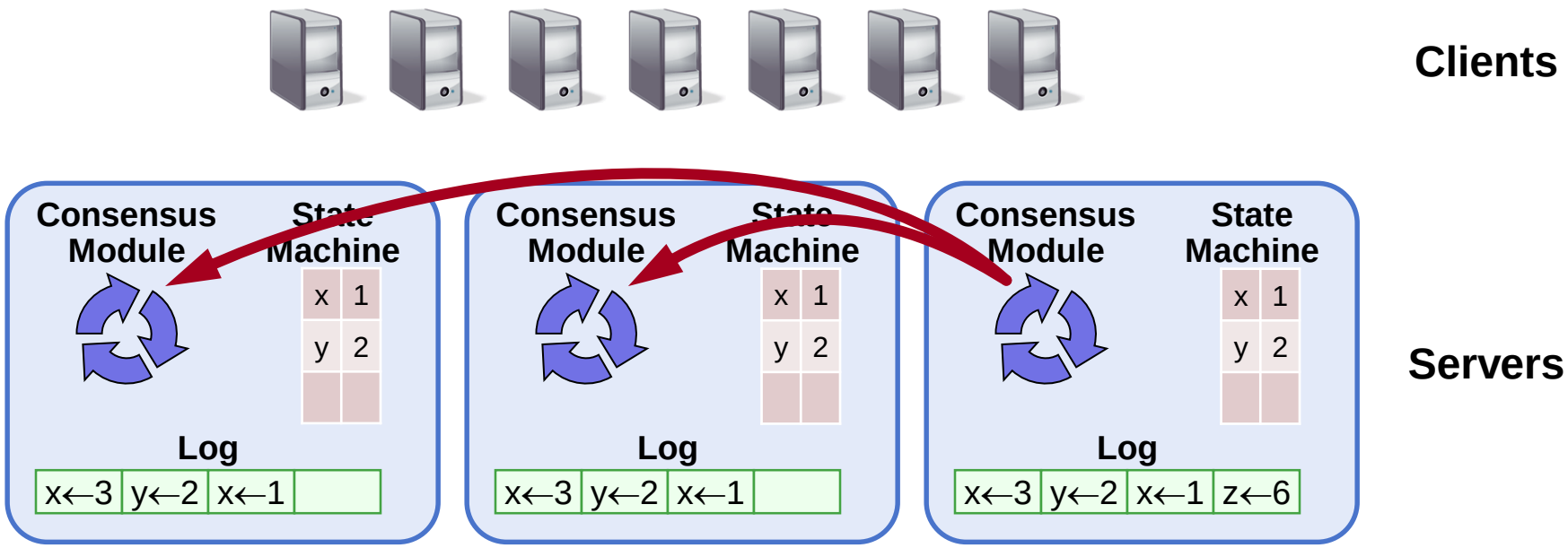
Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

Replicated State Machines (2)



Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

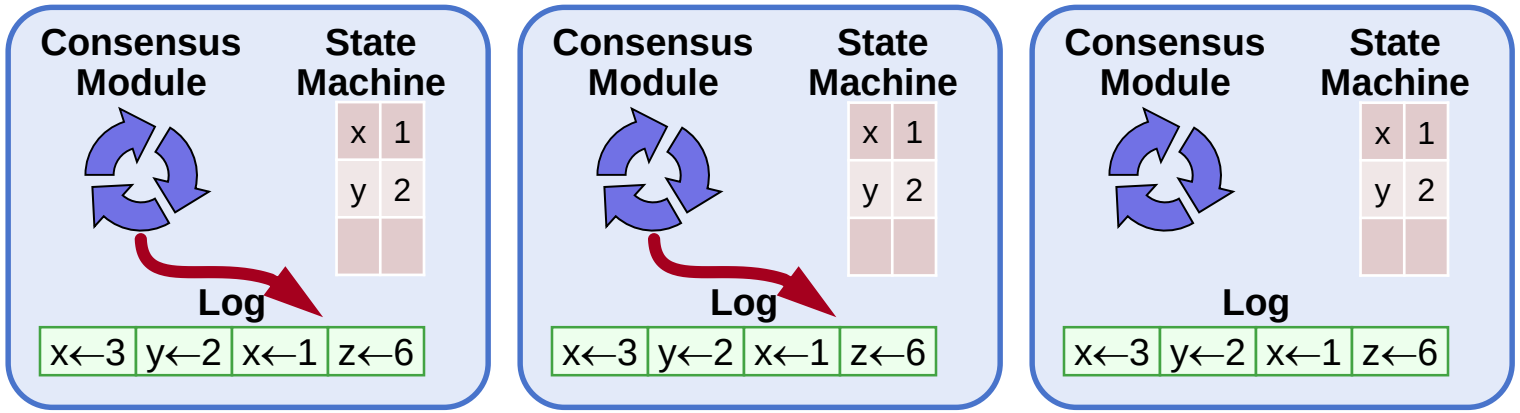
Replicated State Machines (3)



Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

Replicated State Machines (4)

Clients



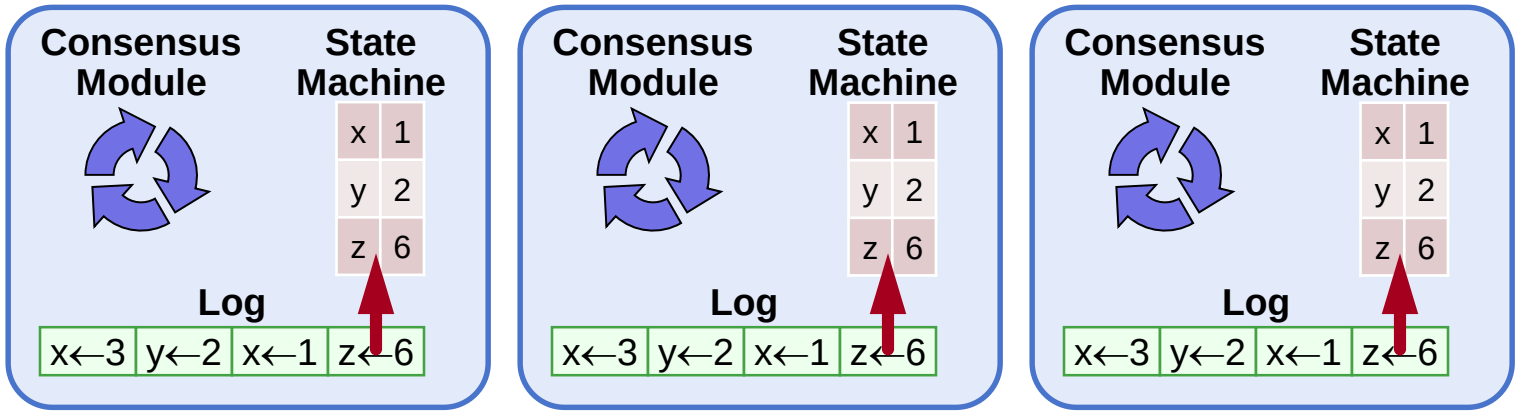
Servers

Replicate log

Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

Replicated State Machines (5)

Clients

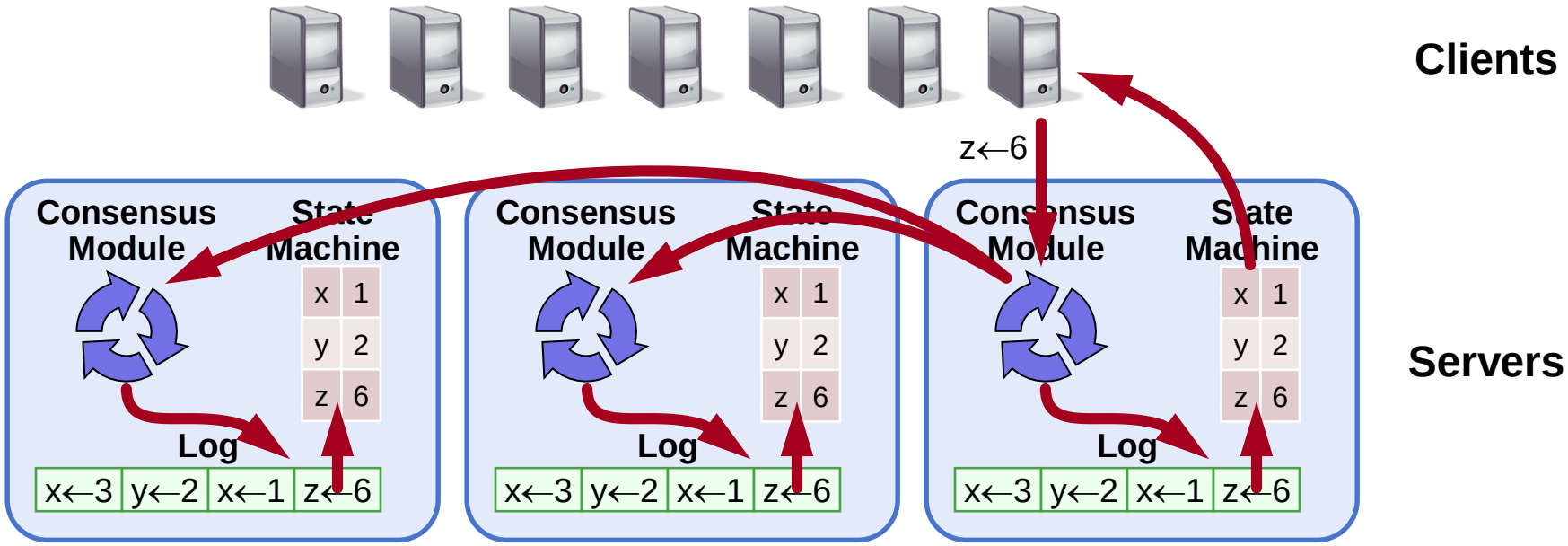


Servers

Safely replicate log
execute the command

Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

Replicated State Machines (6)



Consensus Module	Manage & Replicate the logs
Log	collection of commands
State Machine	Execute the commands & Produce result

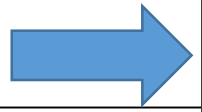
Raft 알고리즘의 핵심

- Leader Election
  - leader의 failure 발생 시, 새 leader 가 반드시 선출되야 한다.
- Log Replication
  - client로부터 log entry를 받으면 클러스터의 노드에 복사해준다.
- Safety
  - consistency(일관성), leader election 관련 안전성

RPC for Raft

AppendEntries RPC

- Arguments
  - term
  - leaderID
  - prevLogIndex
  - prevLogTerm
  - entries[]



entries값(data값)을 empty이면 Heartbeat

RequestVote RPC

- Arguments
  - term
  - candidateID
  - lastLogIndex
  - lastLogTerm

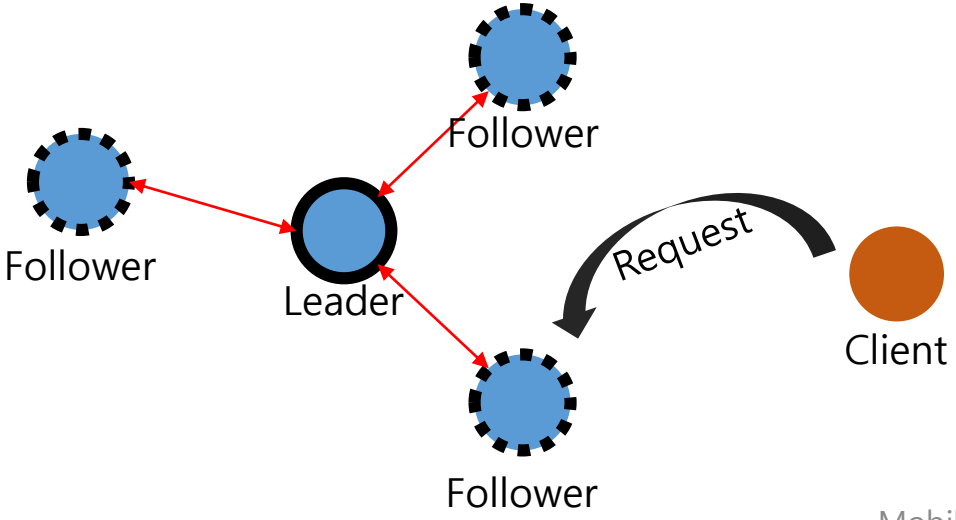
RPC : Remote Procedure Call

Raft에서의 세 가지 상태

- Follower state
- Candidate state
- Leader state

Raft에서의 세 가지 상태 (cont)

- Follower state
  - passive node; 모든 노드의 요청에 대한 응답만 수행 (issue no RPC)
  - 클라이언트가 follower에 접속하면 leader에게 리다이렉트



Raft에서의 세 가지 상태 (cont)

- Candidate state
  - follower가 timeout된 상태
  - leader에게 heartbeat를 받지 못 했을 때
  - ➔candidate 상태로 전이

Raft에서의 세 가지 상태 (cont)

- Leader state
  - 모든 클라이언트의 요청을 처리
  - 다른 노드(서버)의 Log replication 담당
  - 반드시 가용한 leader 가 존재

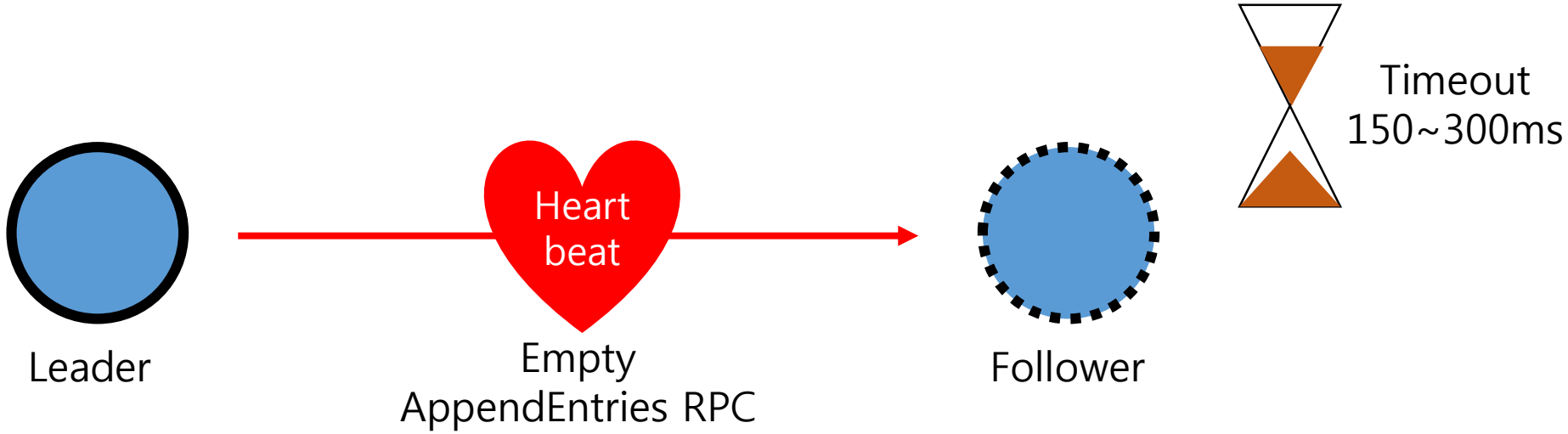
Leader Election

- 기본적으로 노드들은 follower 로 시작

- leader는 heartbeat 이용
  - heartbeat == Empty AppendEntries RPC
  - 150ms < timeout < 300ms
  - when timeout, follower -> candidate

- candidate는 과반수의 표를 획득하면 leader 로 상태 전이

Leader Election (cont)

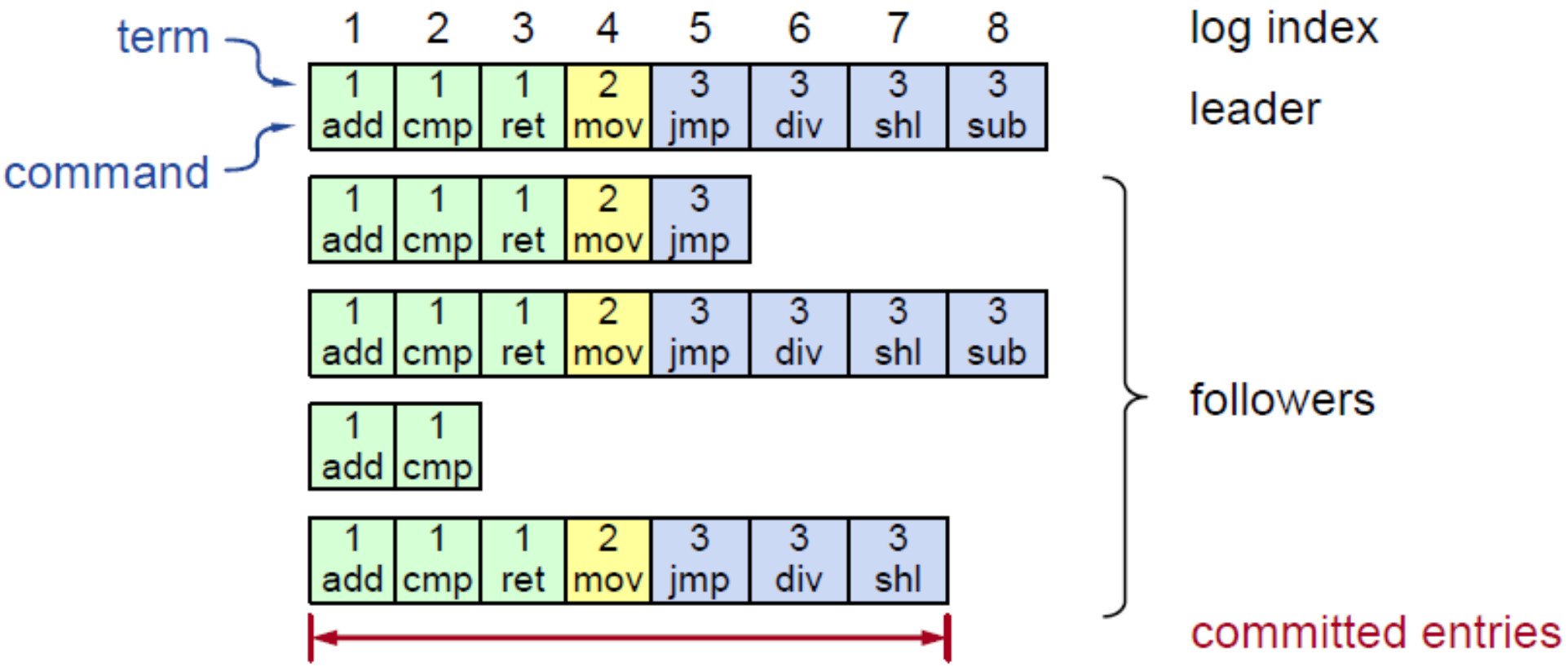


Leader Election (cont)

- 일반적인 heartbeat
- leader가 failure 되었을 경우 or leader 의 응답이 늦을 경우
- http://raftconsensus.github.io/

Log replication

- leader 가 AppendEntries RPC로 수행
- 다른 노드로 복사(동기화)
- https://youtu.be/4OZZv80WrNk


term
command
1 2 3 4 5 6 7 8
log index
leader
followers
committed entries
1 add
1 cmp
1 ret
2 mov
3 jmp
3 div
3 shl
3 sub

Copycat

Copycat! Why?

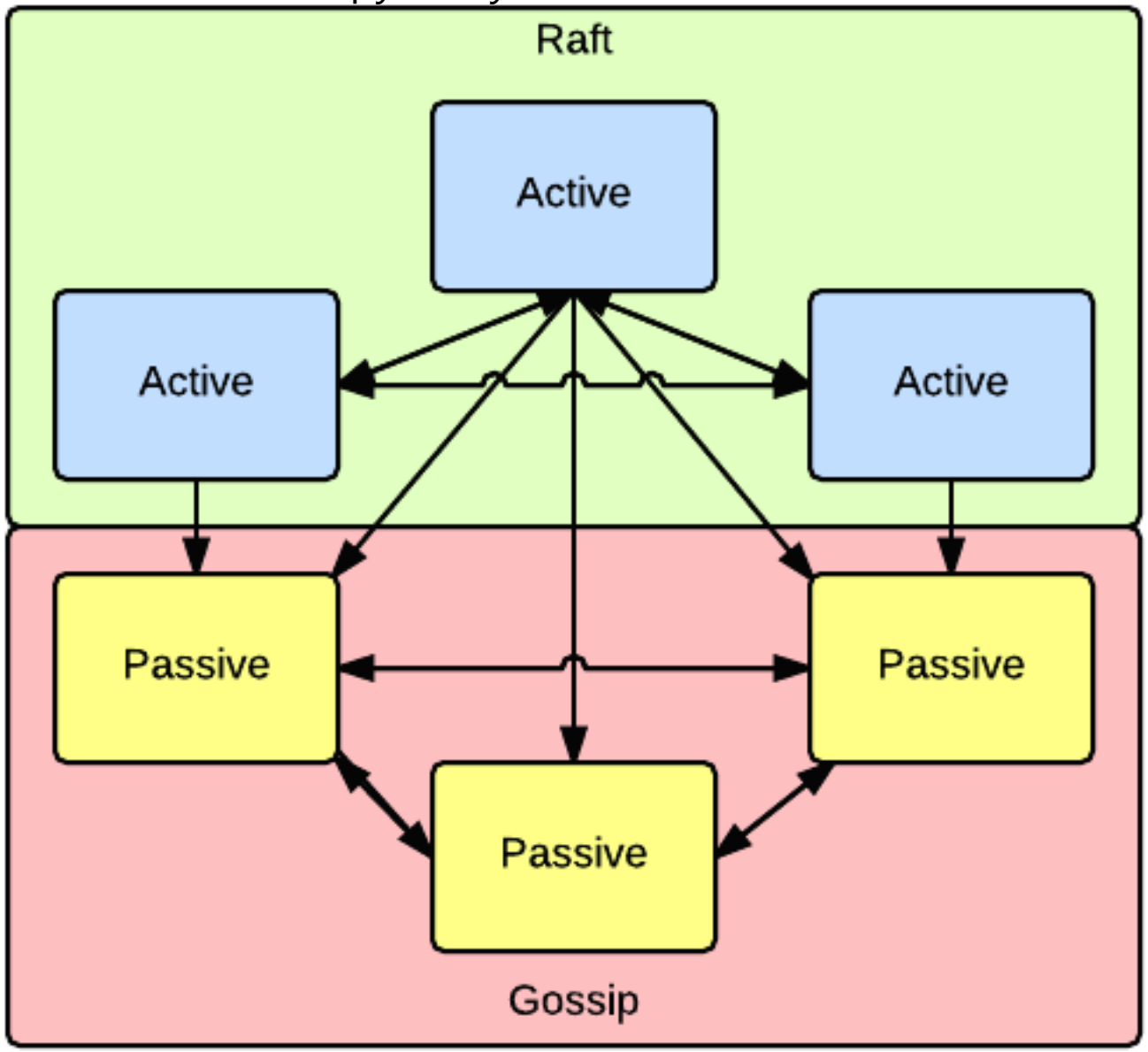
By Madan Jampani

- we chose to use Copycat because:
  - 순수 자바 기반의 구현물
  - 라이선스가 부합한 오픈소스
  - 확장성과 커스텀 가능성(customizability)
  - 최근까지 커밋, 지속적인 발전
  - Well documentation

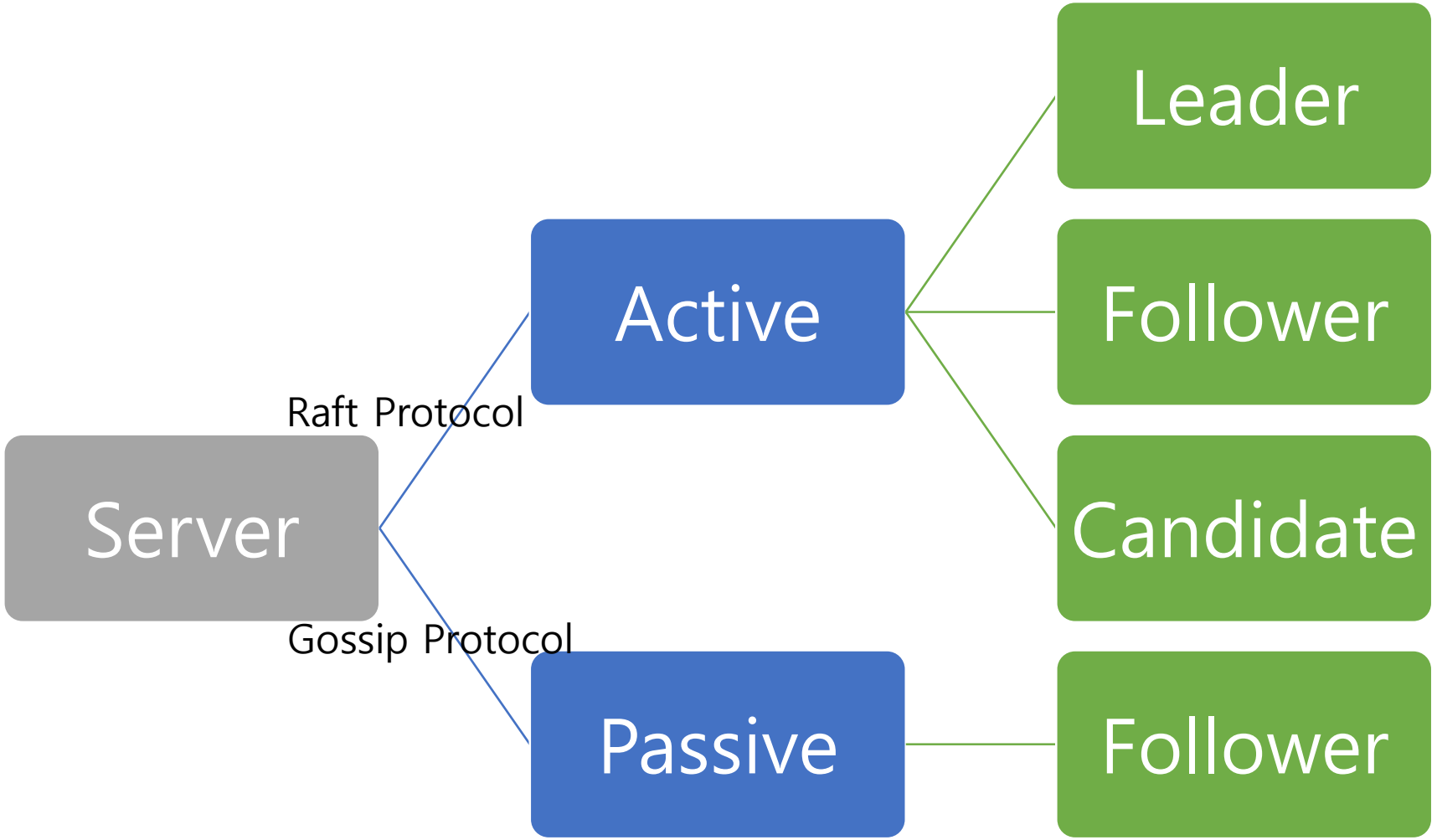
copycat

- distributed coordination framework
- 수많은 Raft consensus protocol 구현물 중 하나
- Raft implement + α

Copycat System Architecture



- Active member
  - 리더가 될 수 있는 멤버
  - Raft protocol
  - Synchronous log replication
- Passive member
  - 리더 선출에 참여하지 않는 follower
  - Gossip protocol
  - Asynchronous log replication

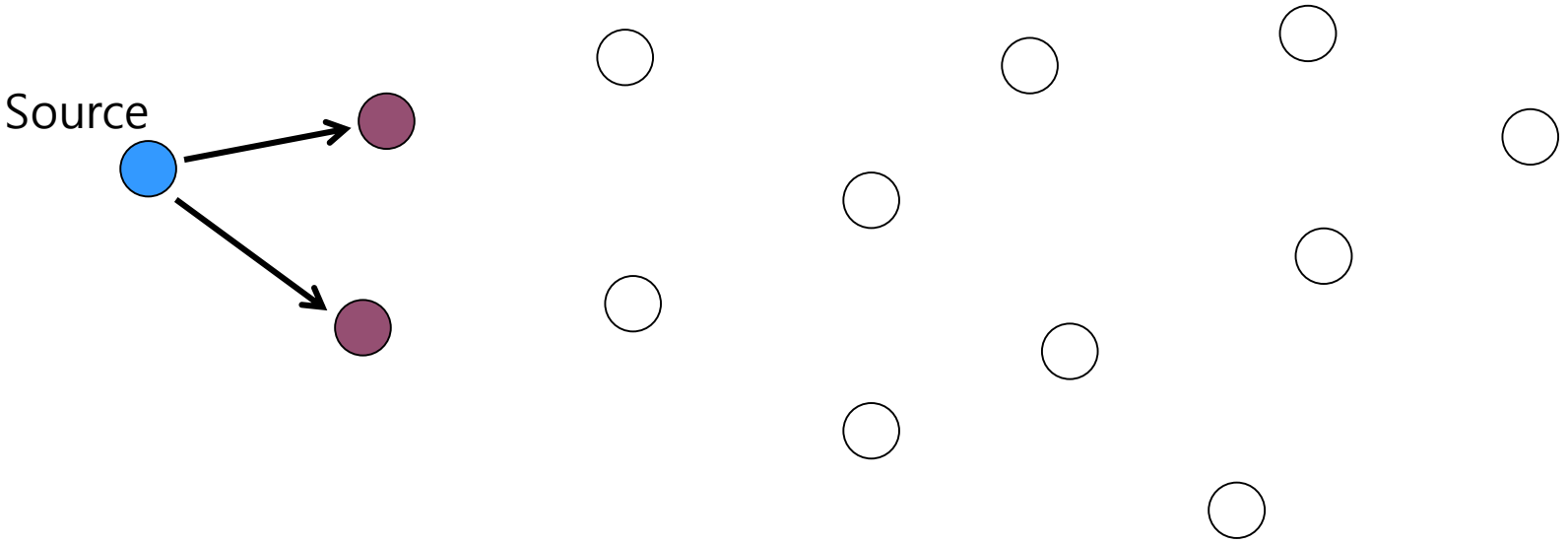

Server
Raft Protocol
Gossip Protocol
Active
Passive
Leader
Follower
Candidate
Follower

Gossip protocol

- =epidemic protocol
- messages broadcast
- 주기적으로 랜덤한 타겟을 골라 gossip message 전송, 이것을 받아 infected 상태가 된 노드도 똑같이 행동

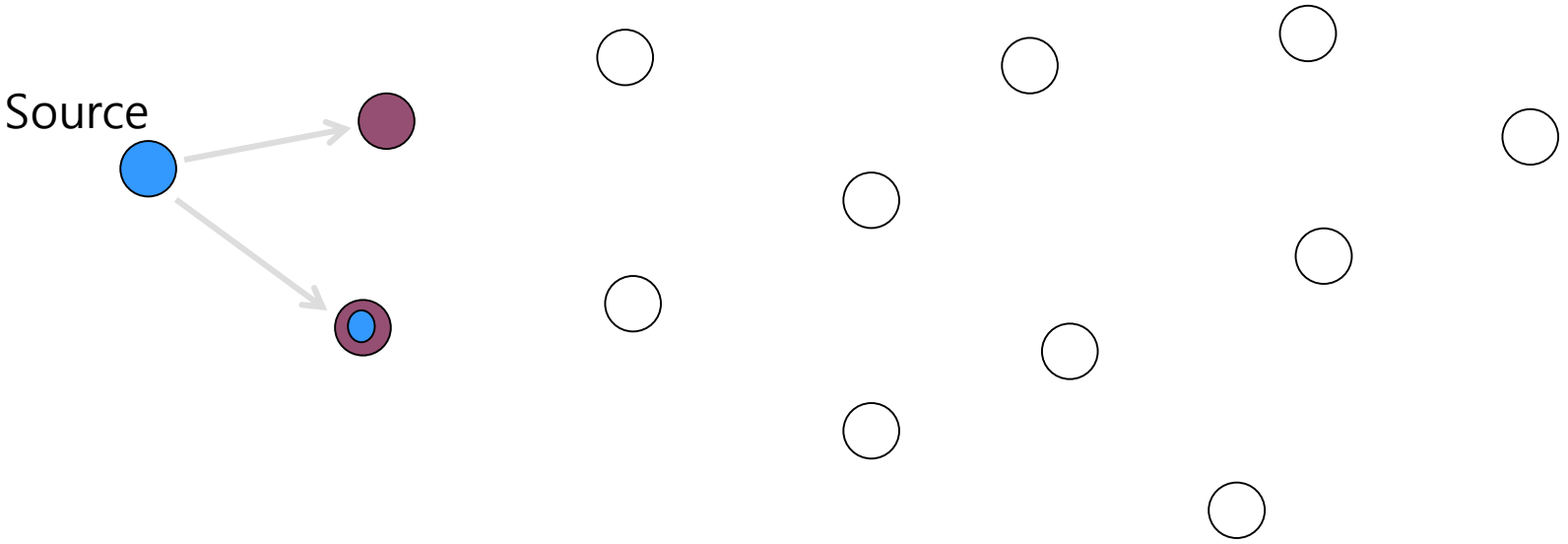
Gossip protocol (1)

- Gossiping = Probabilistic flooding
  - Nodes forward with probability, p



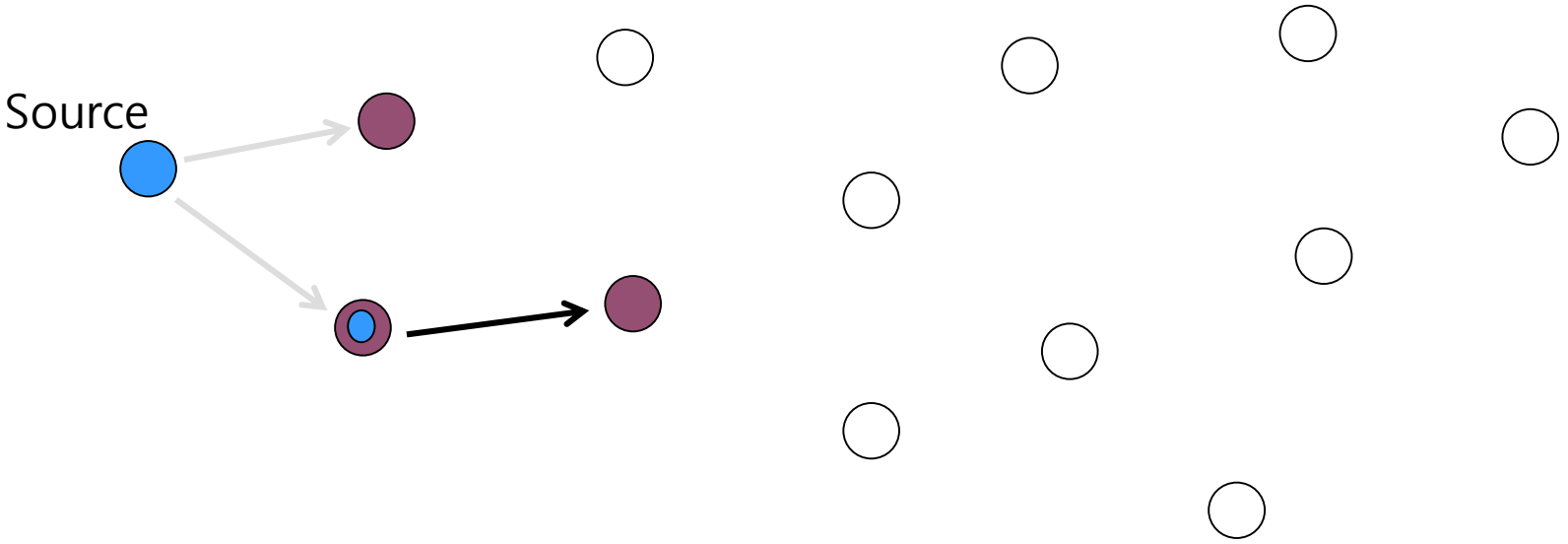
Gossip protocol (2)

- Gossip based broadcast
  - Nodes forward with probability, p



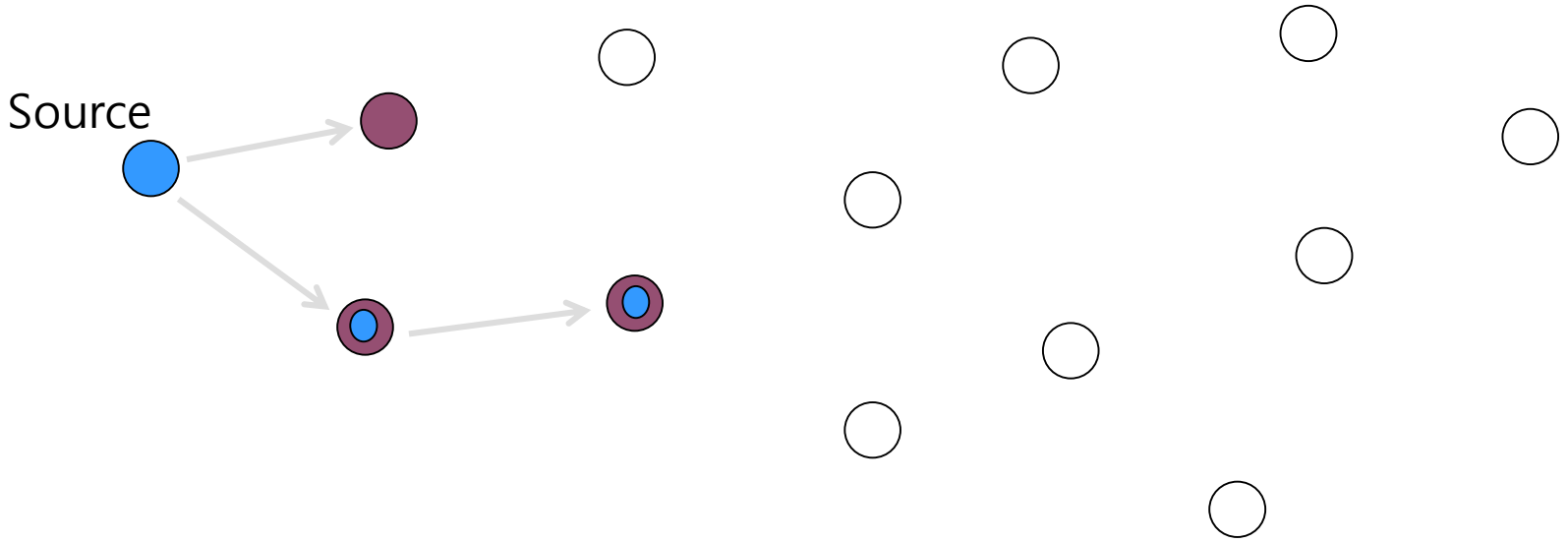
Gossip protocol (3)

- Gossip based broadcast
  - Nodes forward with probability, p



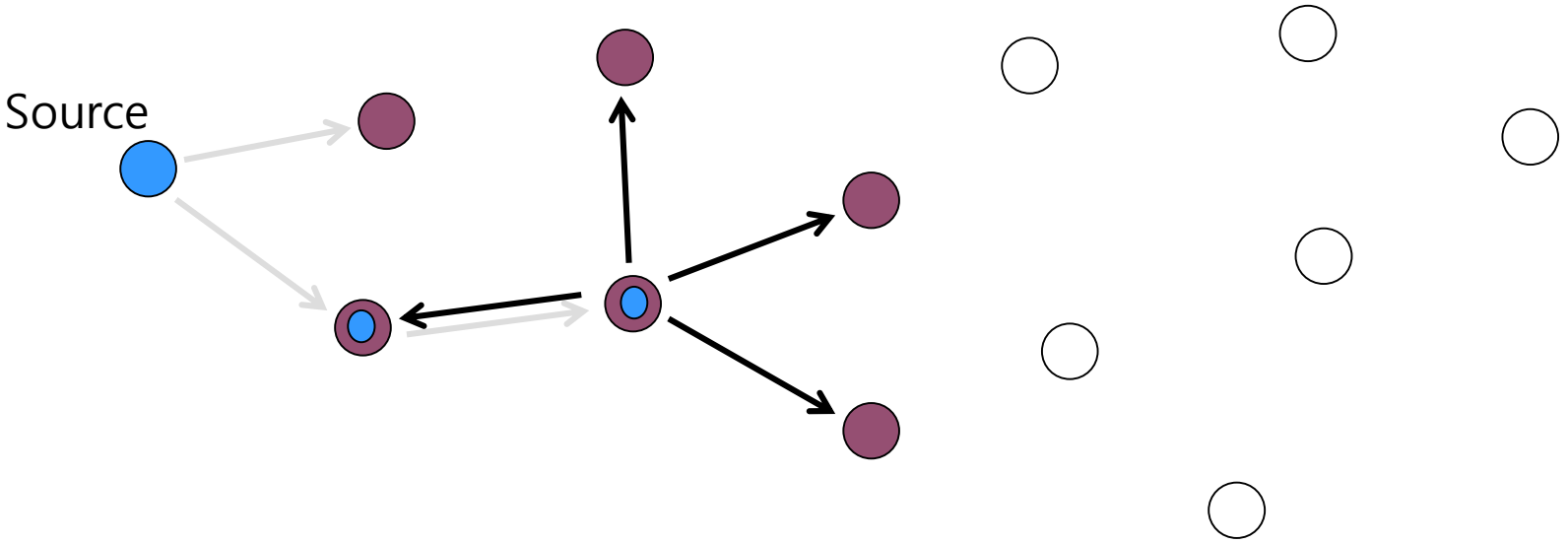
Gossip protocol (4)

- Gossip based broadcast
  - Nodes forward with probability, p



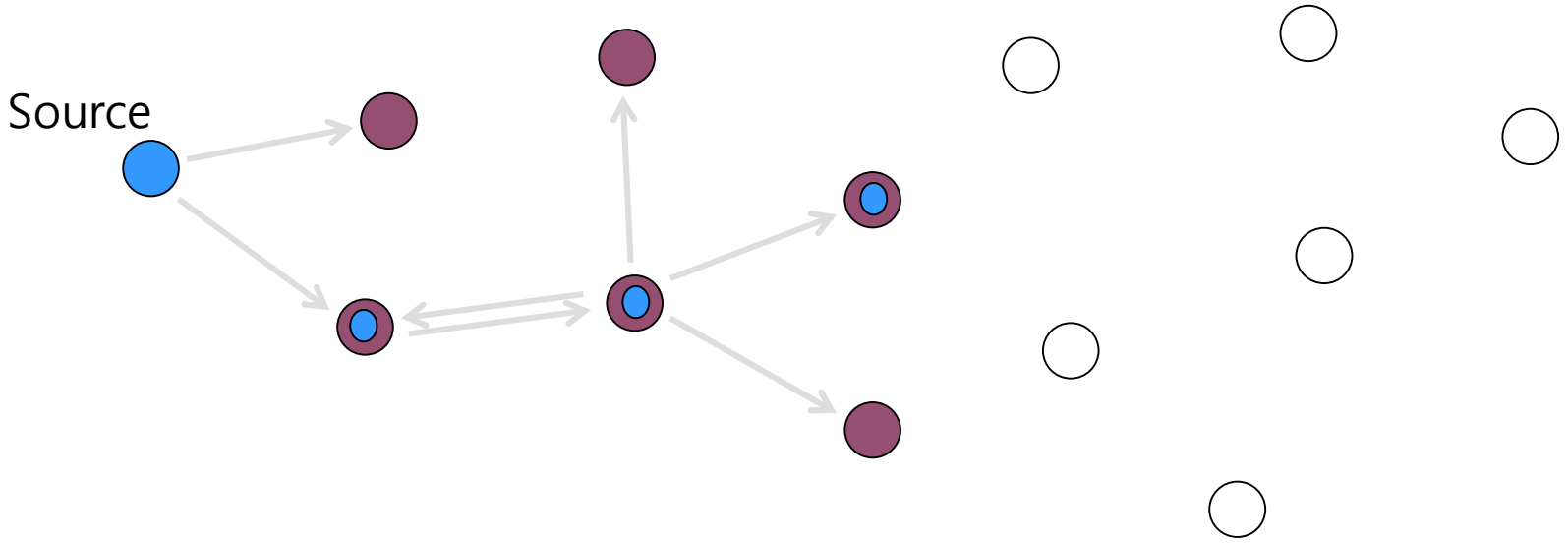
Gossip protocol (5)

- Gossip based broadcast
  - Nodes forward with probability, p



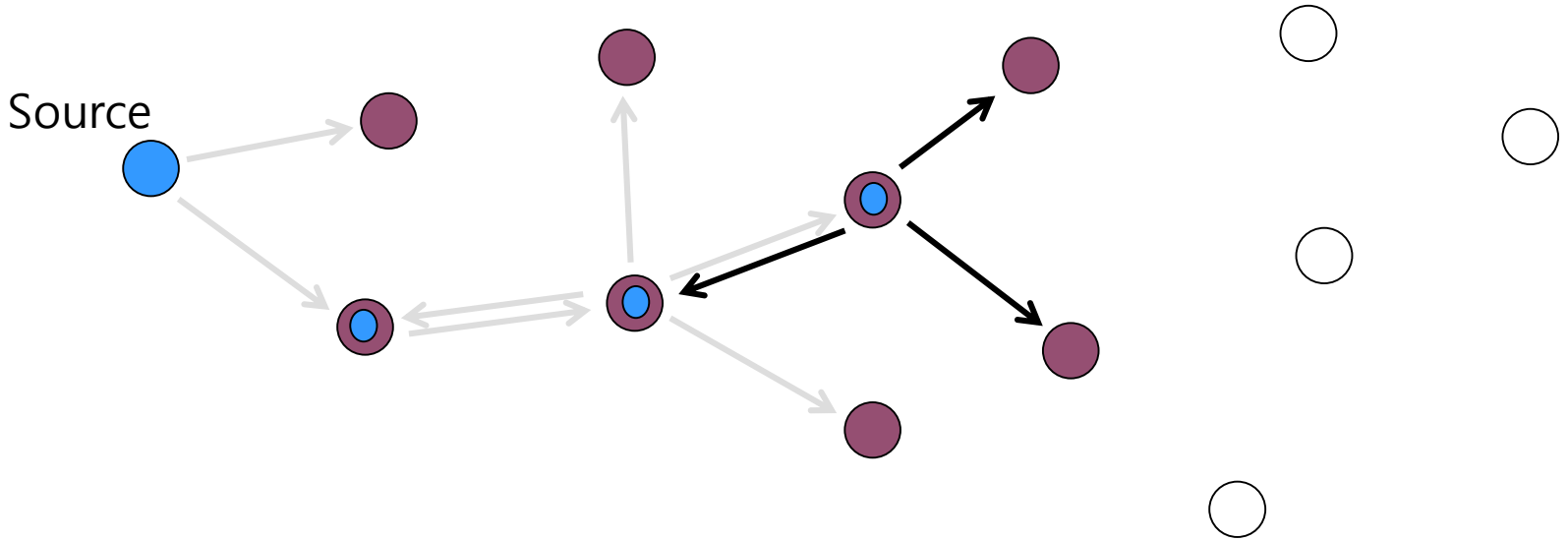
Gossip protocol (6)

- Gossip based broadcast
  - Nodes forward with probability, p



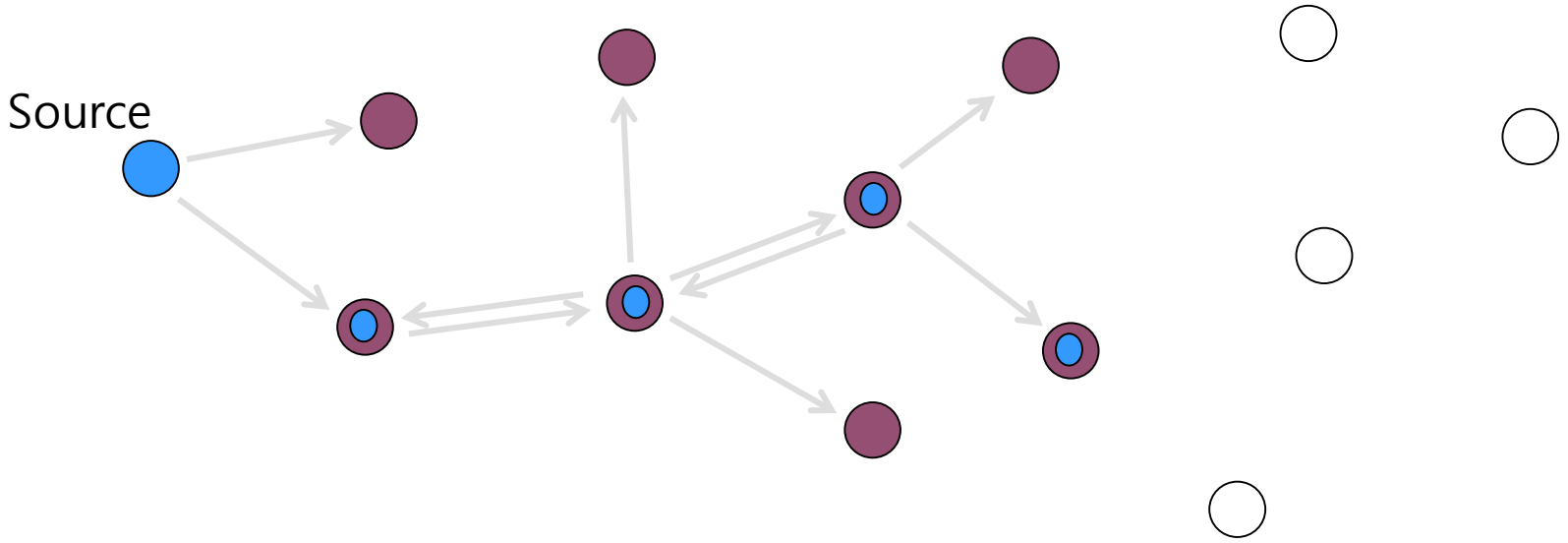
Gossip protocol (7)

- Gossip based broadcast
  - Nodes forward with probability, p



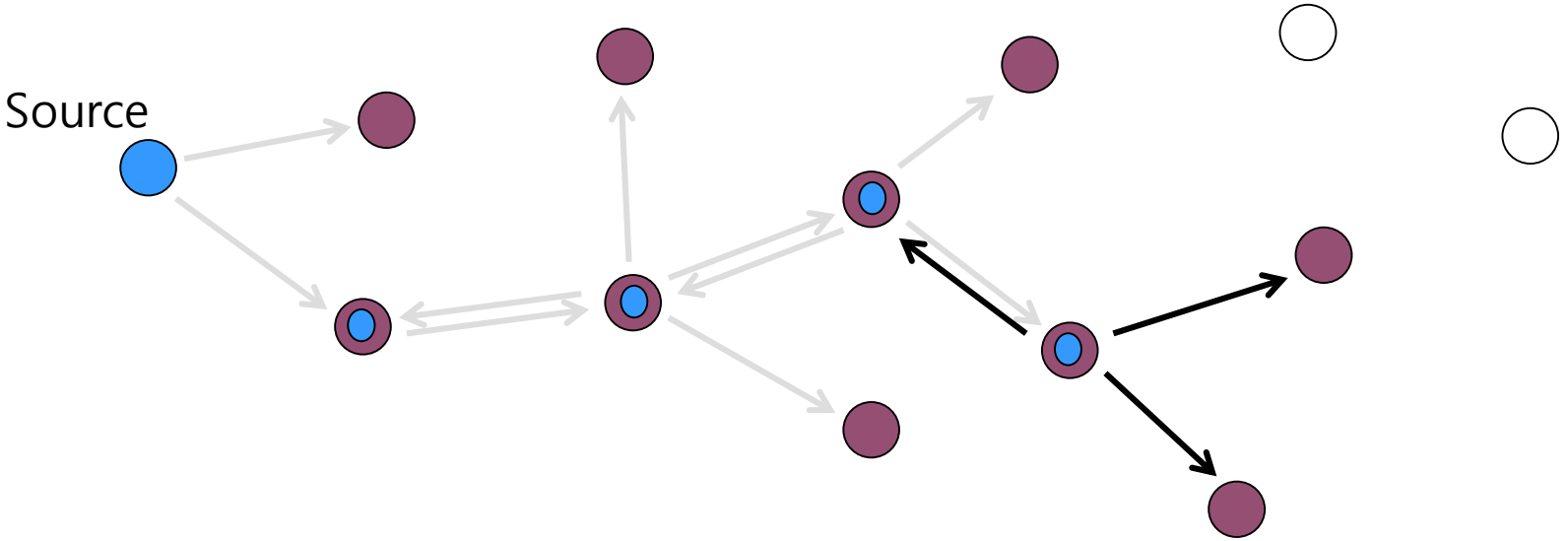
Gossip protocol (8)

- Gossip based broadcast
  - Nodes forward with probability, p



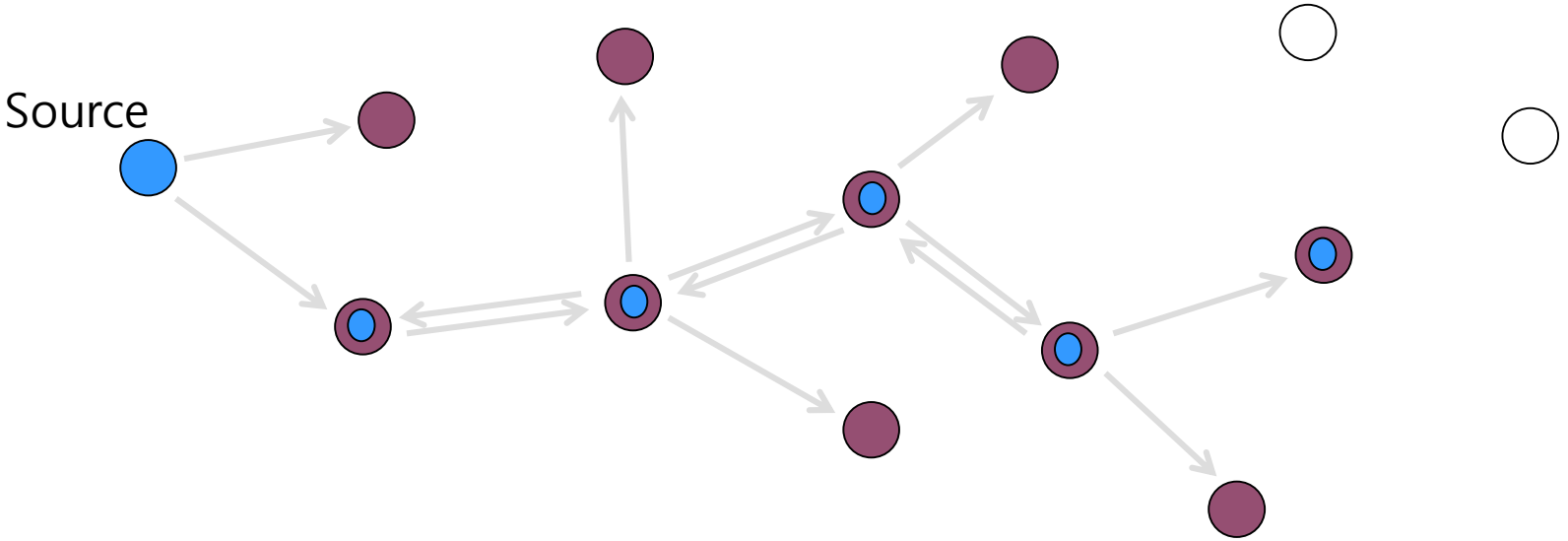
Gossip protocol (9)

- Gossip based broadcast
  - Nodes forward with probability, p



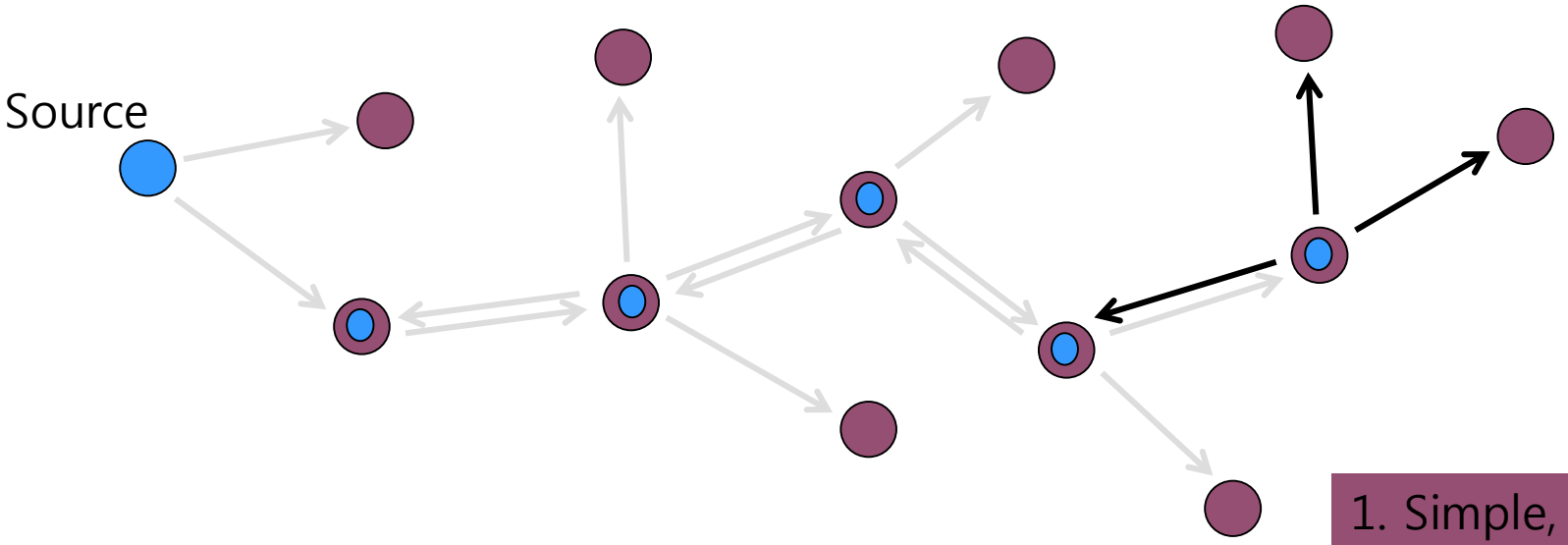
Gossip protocol (10)

- Gossip based broadcast
  - Nodes forward with probability, p



Gossip protocol (11)

- Gossip based broadcast
  - Nodes forward with probability, p



1. Simple,
2. Fault tolerant
3. Load-balanced