GridGain

Networking in Apache Ignite

Stanislav Lukyanov

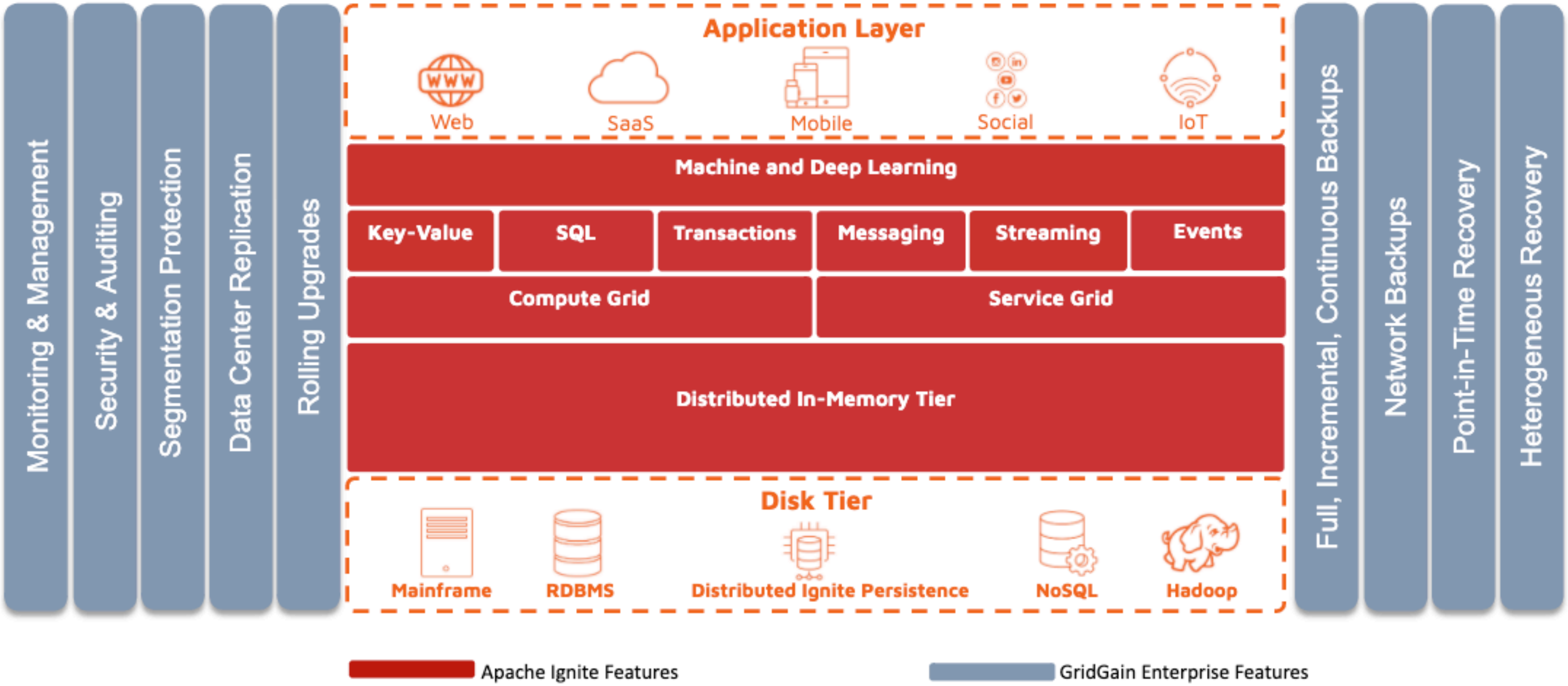
Sep 9, 2020

Functions of Networking in a Distributed System

What can a network component do?

- Transfer data
- Deliver cluster-wide events and messages
- Find nodes and connect to them
- Detect failed nodes

Reference System: Apache Ignite



Data Transfer

Transferring Data

What can we say about this function?

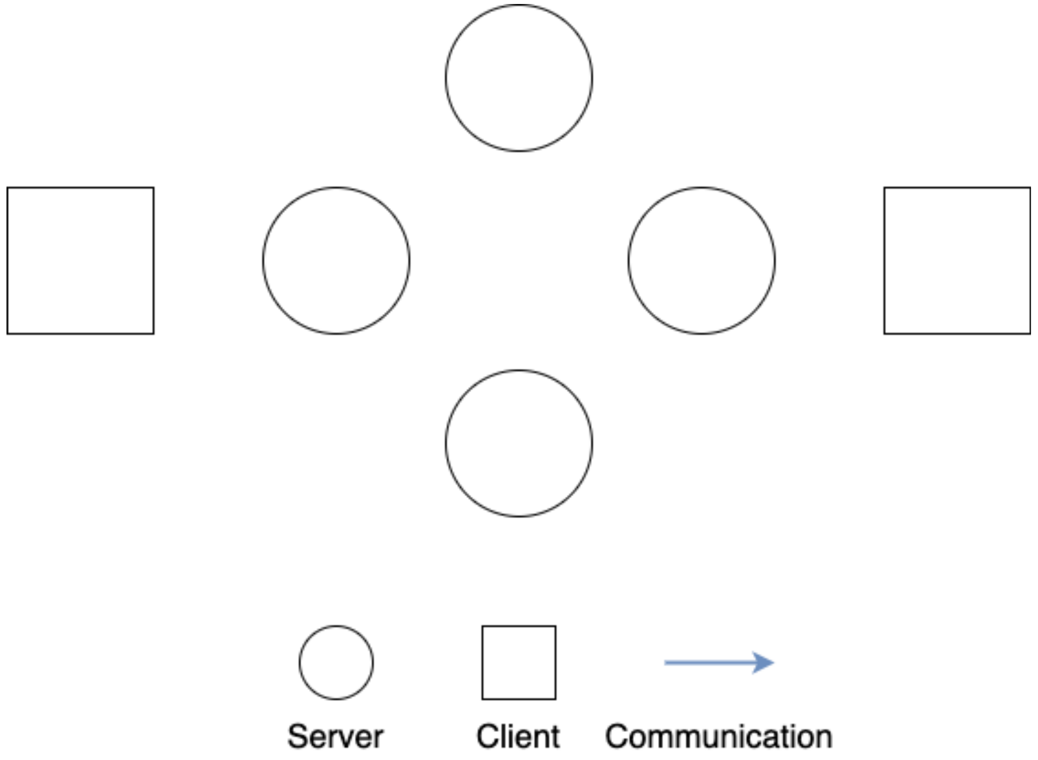
- Defines performance of the cluster
- 99.9% of the networking traffic in a cluster under load
- Key – speed (latency and throughput)

Apache Ignite: Communication

High-speed data transferring component

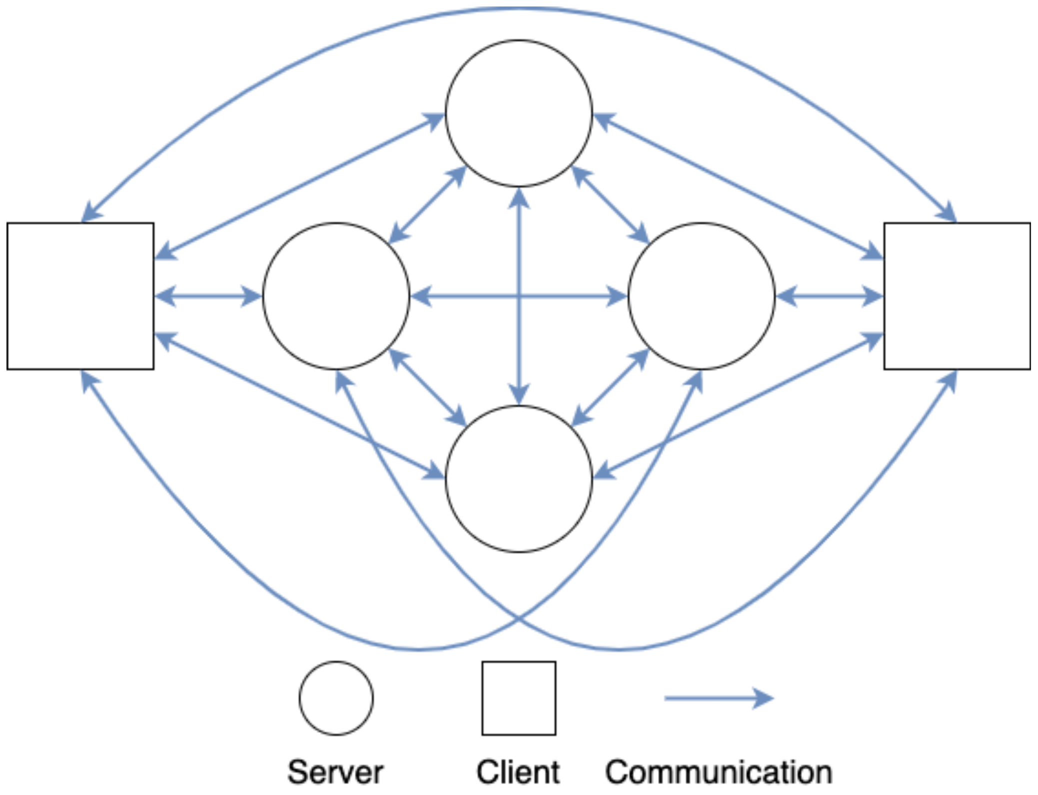
- Almost all operations in the cluster use Communication
  - Data modifications, SQL, Distributed Computing, etc
- Based on java.nio
  - Optimized multithreaded IO
- Peer-to-peer connections
  - One hop – minimal latency
  - Need to budget connections – see next slides

Communication: Typical Topology



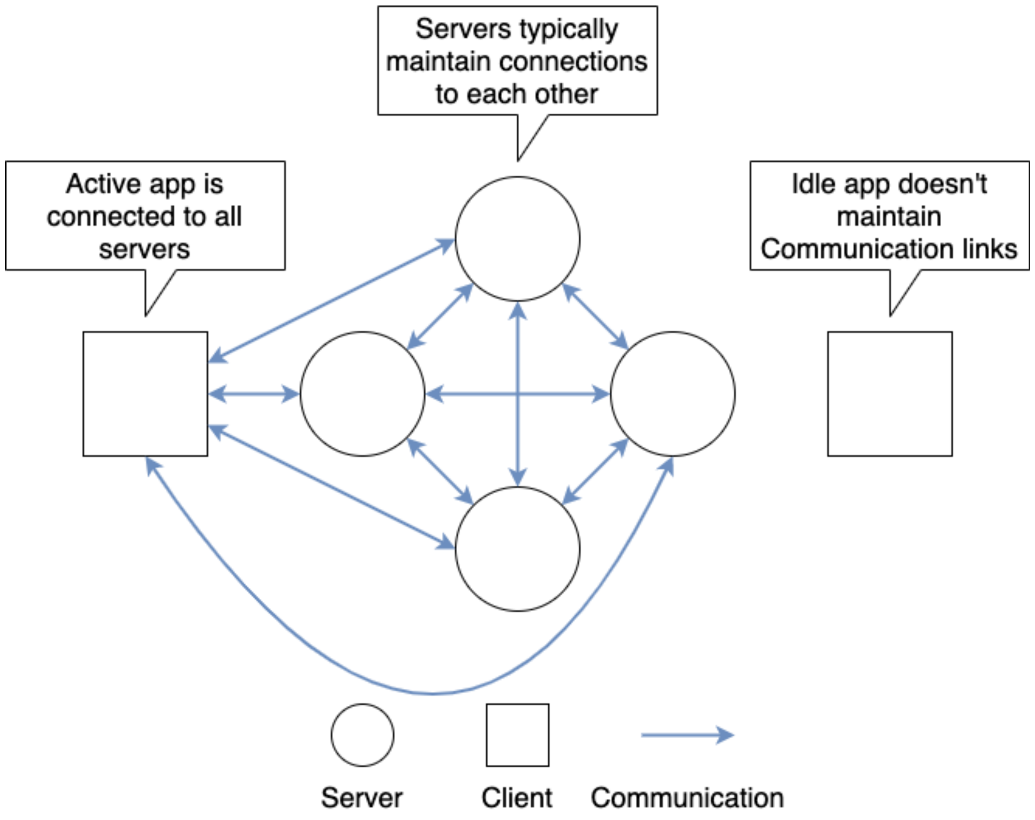
Communication: Typical Topology

- Logically forms a fully-connected topology
  - All nodes are equal



Communication: Typical Topology

- Logically forms a fully-connected topology
  - All nodes are equal
- But connections are a precious resource
  - Creates connections ad-hoc
  - If a connection is idle – drop it and recreate when needed



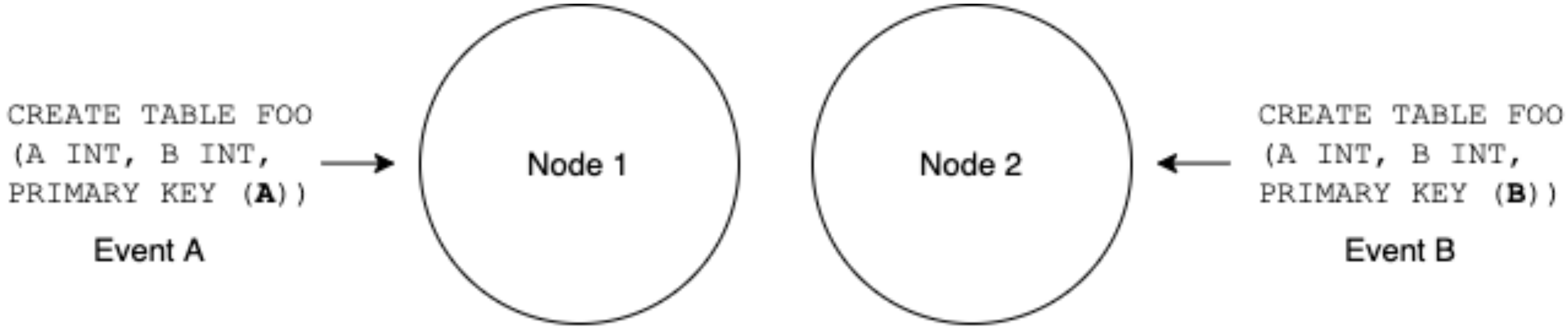
GridGain

Cluster Events Delivery

Cluster Events Delivery

What can we say about this function?

- Base for other distributed protocols
- No need for high throughput
- Distributed protocols may require global ordering of events



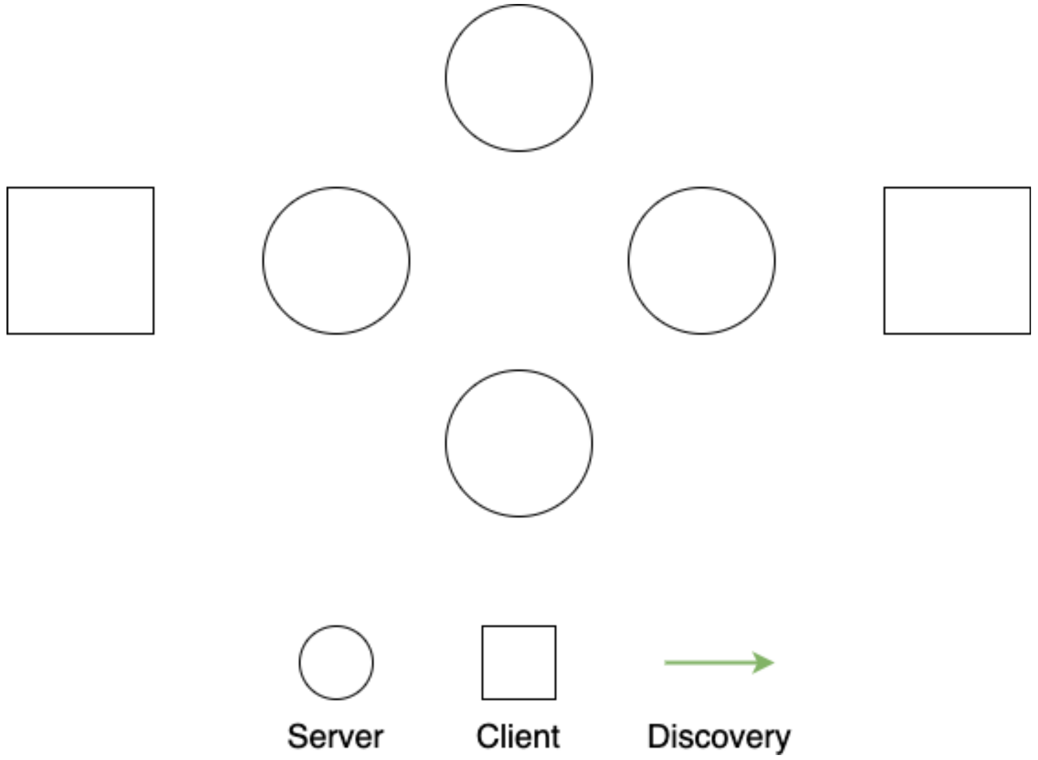
Who was the first, A or B?

Apache Ignite: Discovery

Delivery of cluster-wide events and more

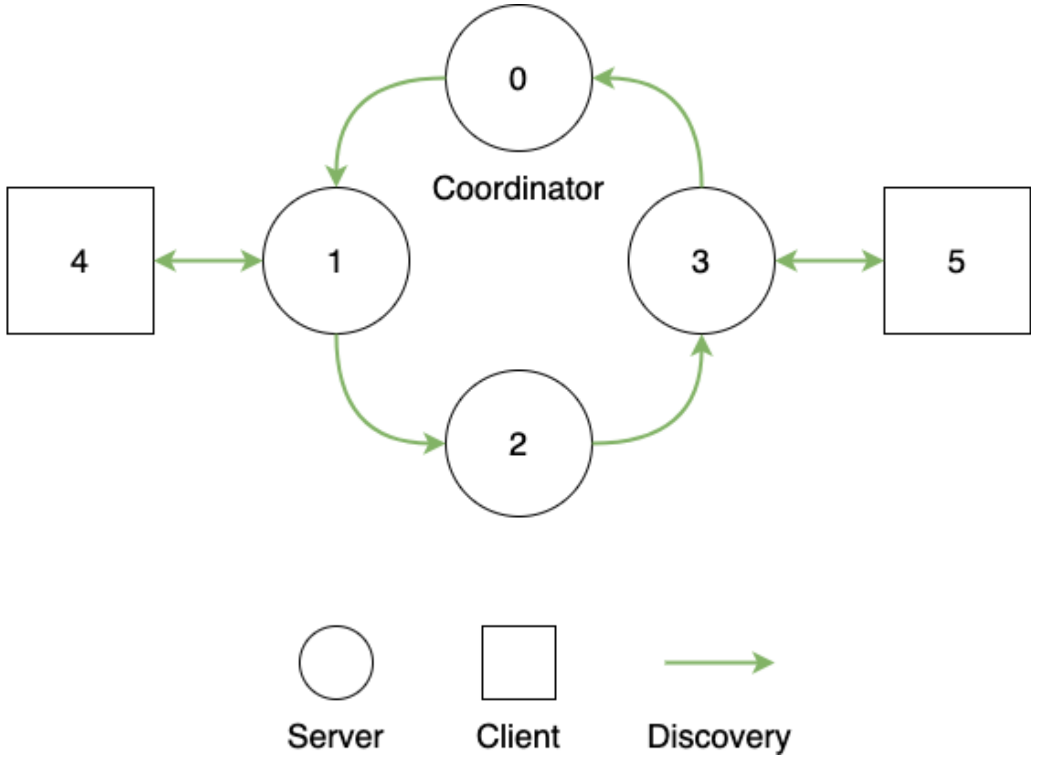
- Delivers ordered events
- Allows nodes to find each other and join the cluster
- Detects failed nodes

Discovery: Typical Topology



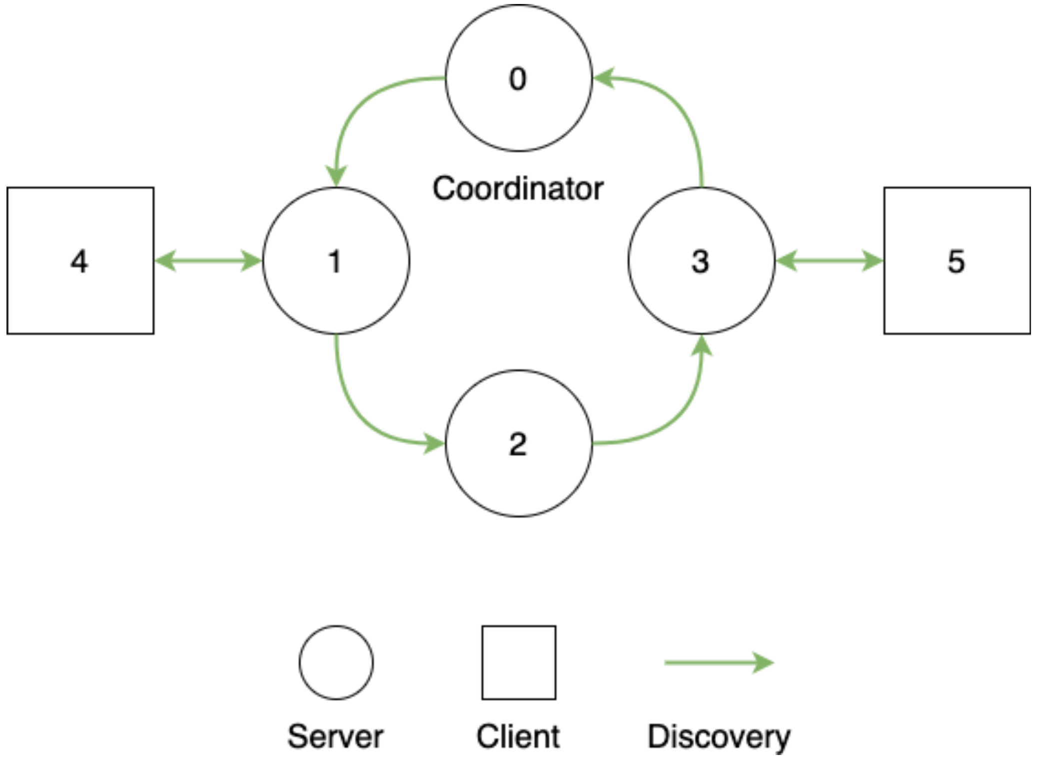
Discovery: Typical Topology

- Servers form a one-way ring
  - Makes it easier to implement global ordering and delivery – see next slides
- Each client is connected to one server
  - Allows the core of the cluster to be more independent from applications
- The oldest server is called "coordinator"
  - Responsible for ordering
- Single-threaded processing



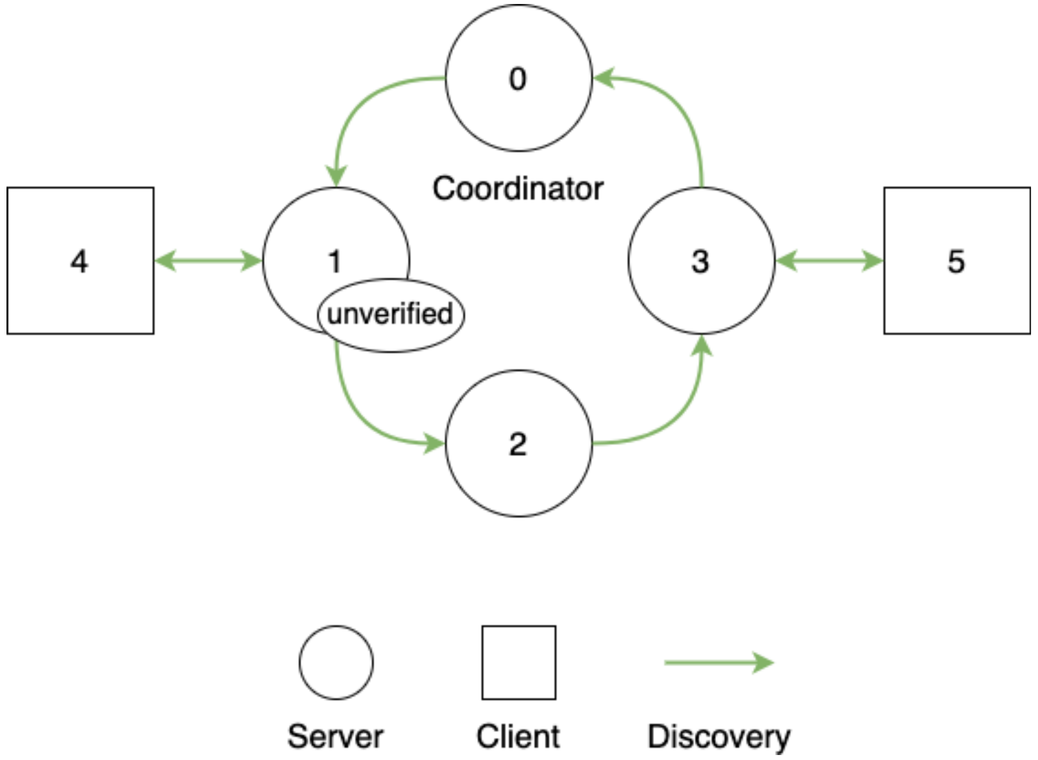
GridGain

Discovery: Ring Message Delivery Protocol



Discovery: Ring Message Delivery Protocol

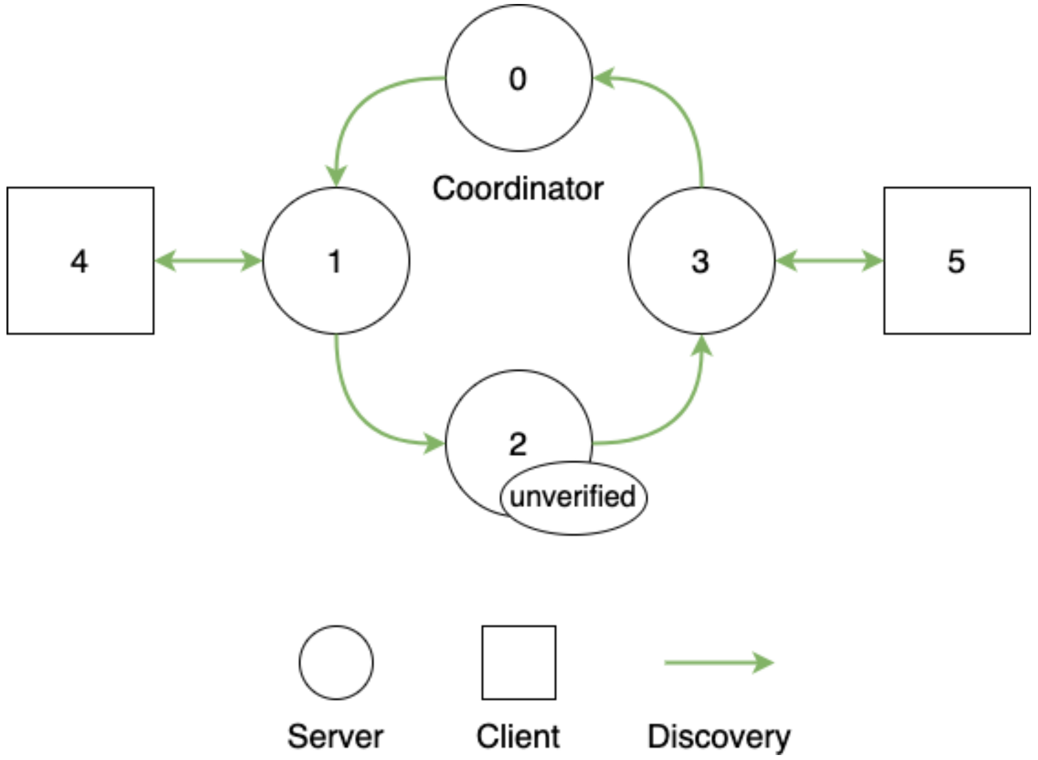
1. A message starts "unverified"



GridGain

Discovery: Ring Message Delivery Protocol

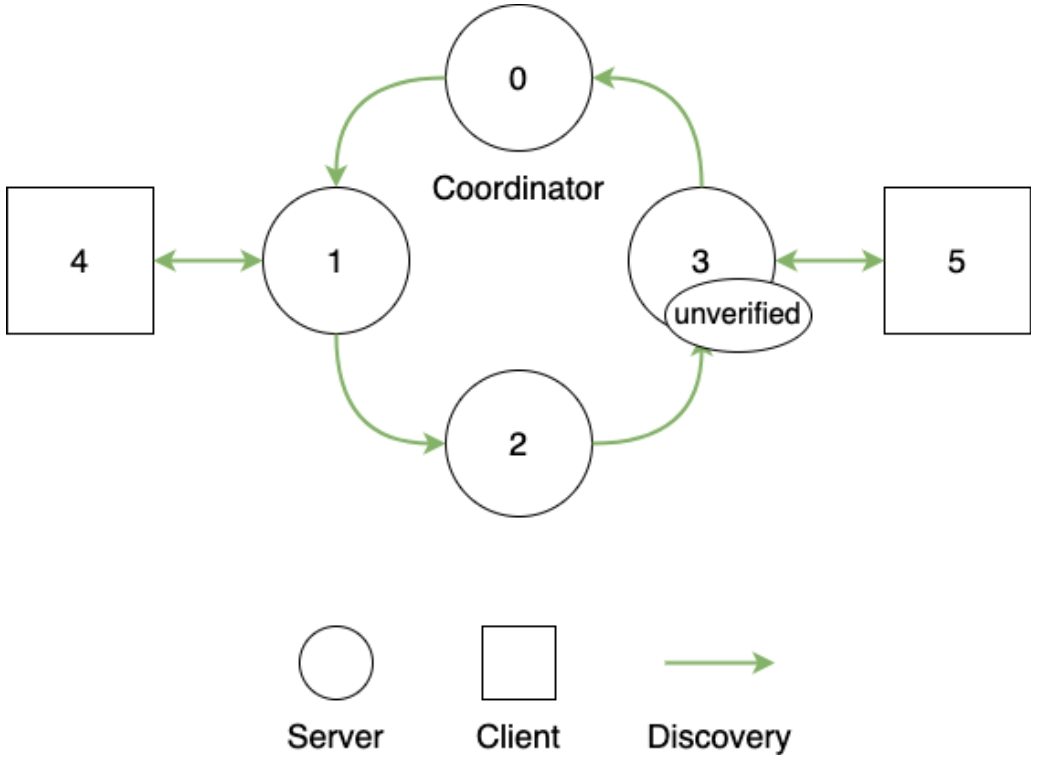
1. A message starts “unverified”
2. Travels to the coordinator
   - Servers don’t actually “process” the message, just send it further



GridGain

Discovery: Ring Message Delivery Protocol

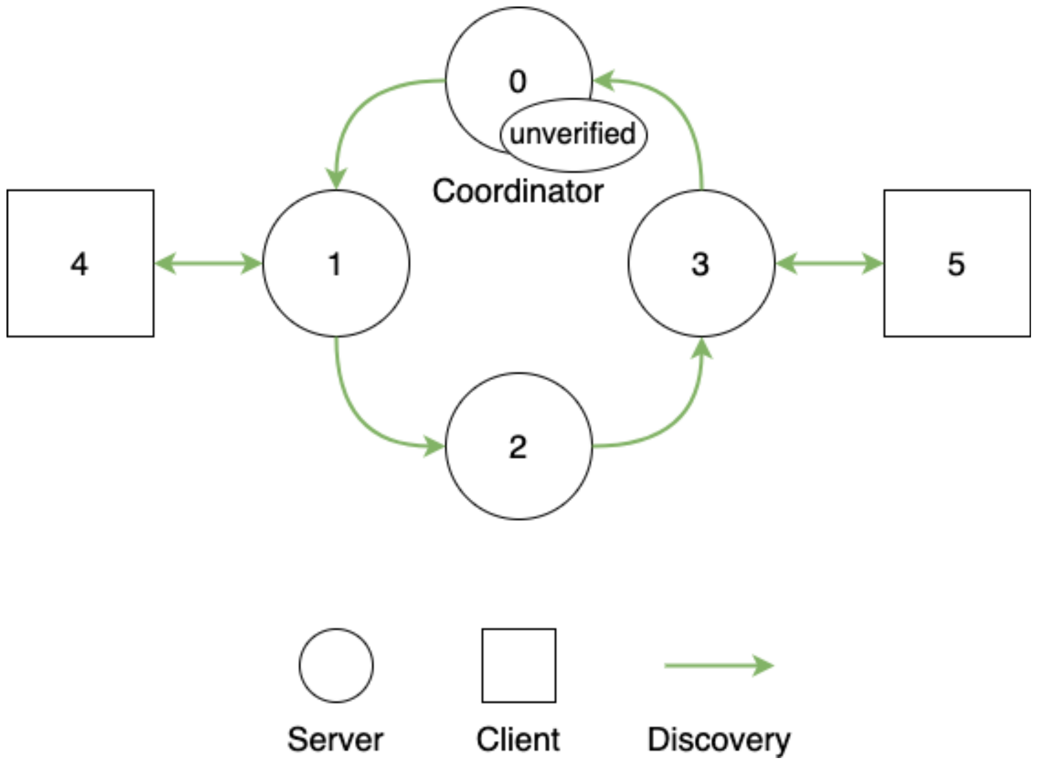
1. A message starts "unverified"
2. Travels to the coordinator
   - Servers don't actually "process" the message, just send it further



GridGain

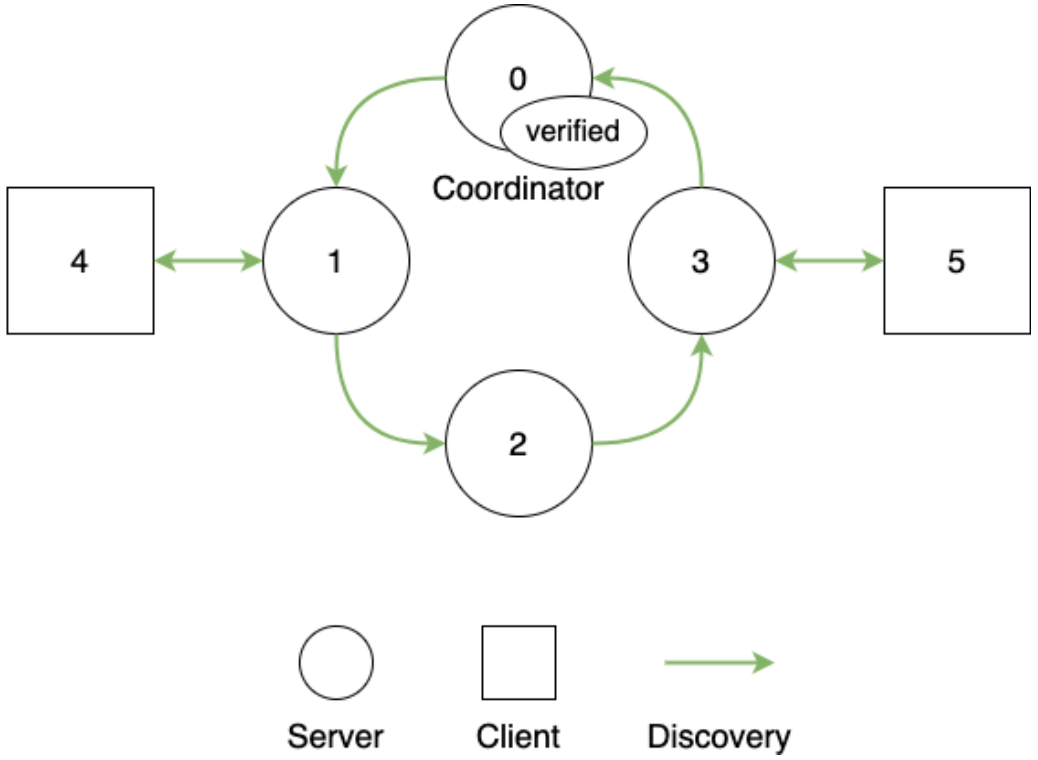
Discovery: Ring Message Delivery Protocol

1. A message starts “unverified”
2. Travels to the coordinator
   - Servers don’t actually “process” the message, just send it further



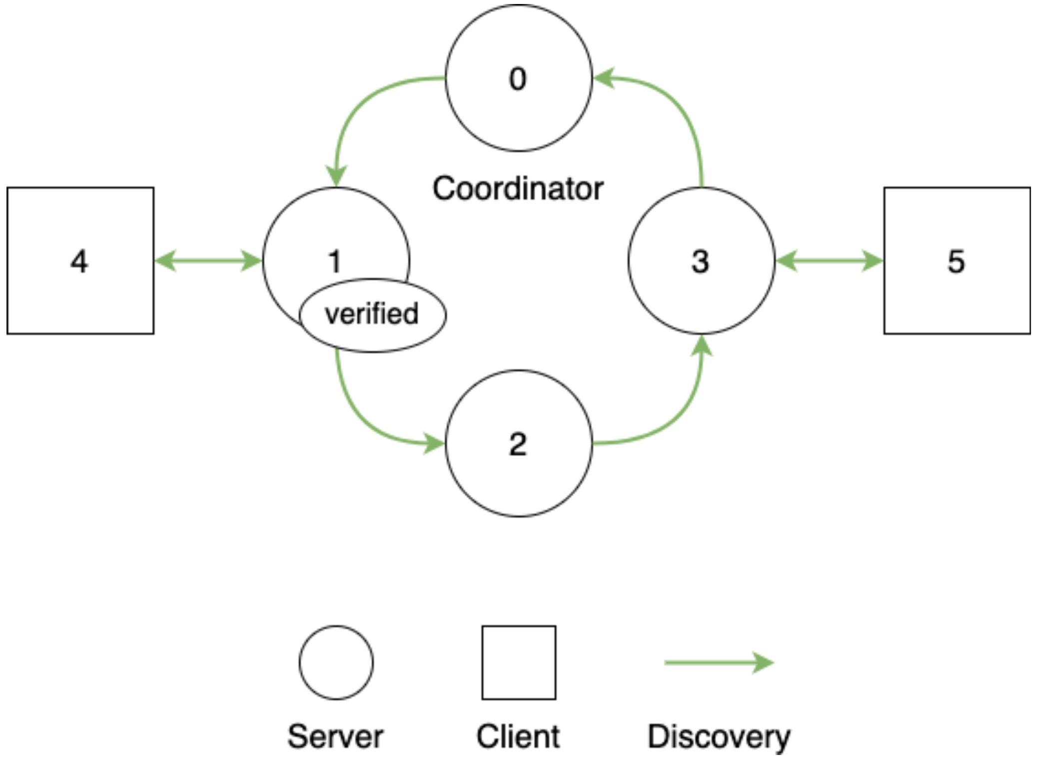
Discovery: Ring Message Delivery Protocol

1. A message starts "unverified"
2. Travels to the coordinator
   - Servers don't actually "process" the message, just send it further
3. Coordinator verifies the message
   - Now the message is **ordered**



Discovery: Ring Message Delivery Protocol

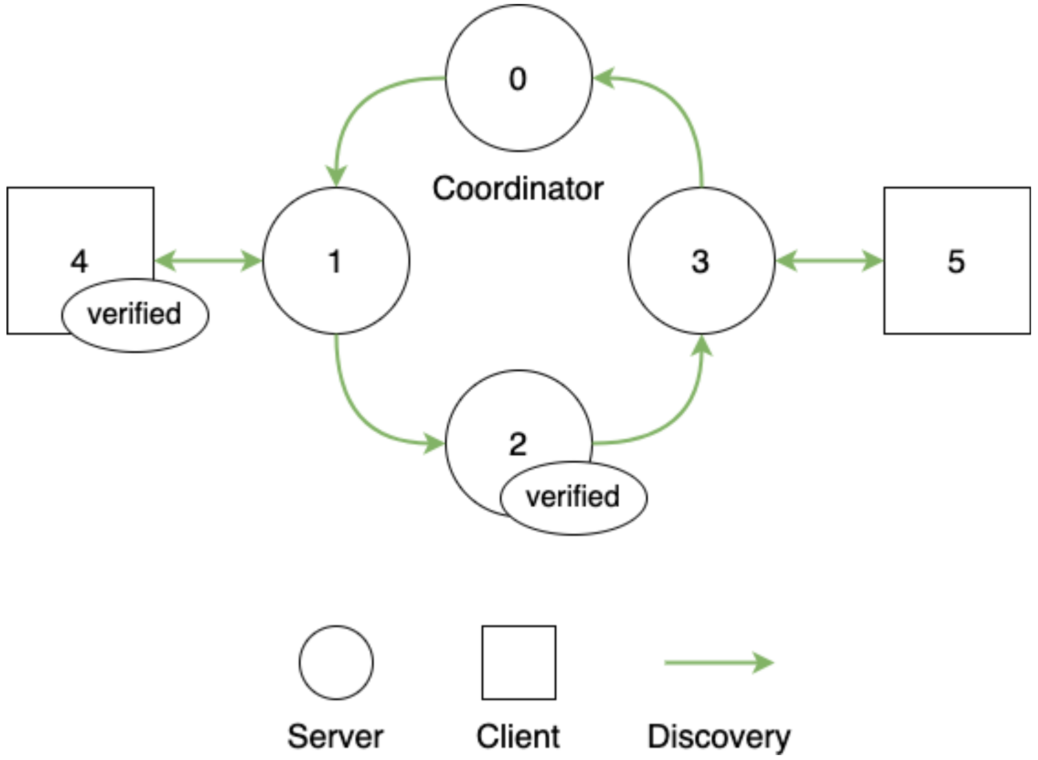
1. A message starts “unverified”
2. Travels to the coordinator
   - Servers don’t actually “process” the message, just send it further
3. Coordinator verifies the message
   - Now the message is **ordered**
4. Coordinator sends the message across the ring again



GridGain

Discovery: Ring Message Delivery Protocol

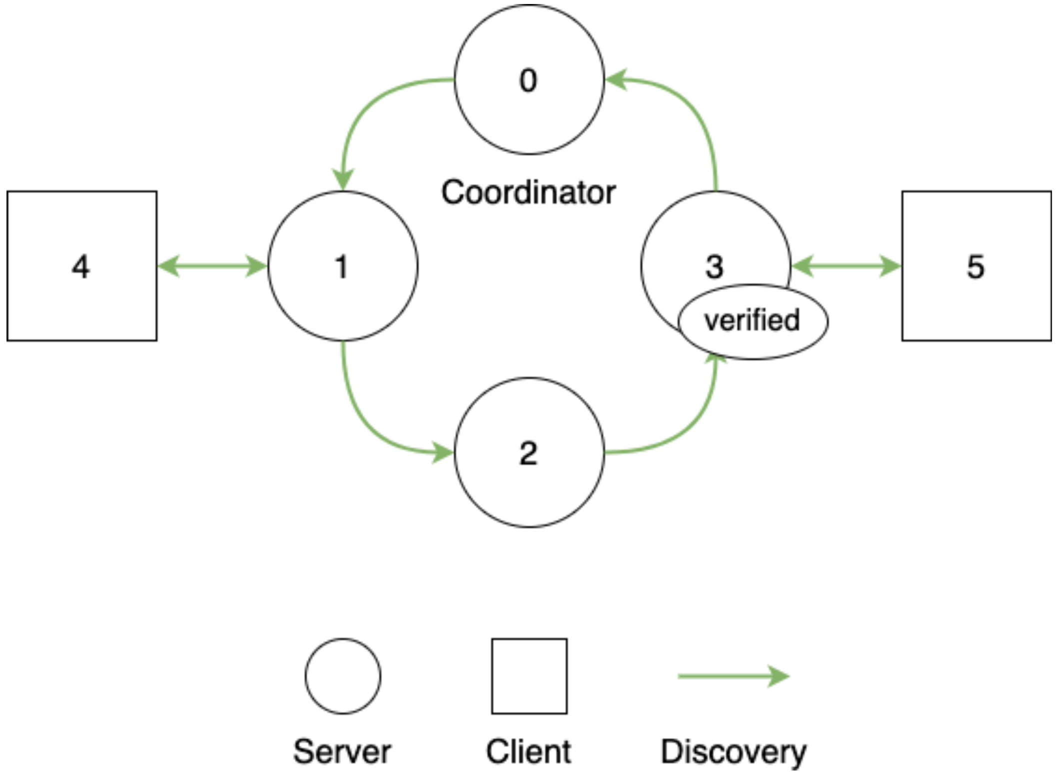
1. A message starts “unverified”
2. Travels to the coordinator
   - Servers don’t actually “process” the message, just send it further
3. Coordinator verifies the message
   - Now the message is **ordered**
4. Coordinator sends the message across the ring again
5. Now servers process the message + send it to clients



GridGain

Discovery: Ring Message Delivery Protocol

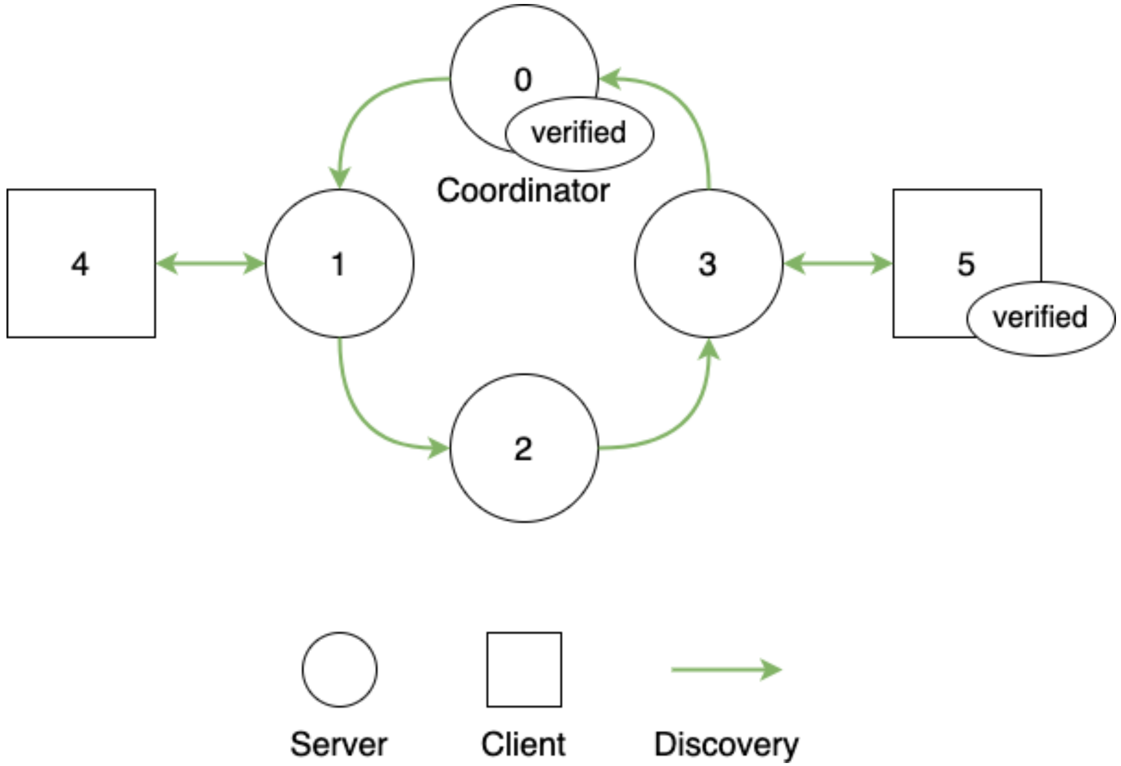
1. A message starts "unverified"
2. Travels to the coordinator
   - Servers don't actually "process" the message, just send it further
3. Coordinator verifies the message
   - Now the message is **ordered**
4. Coordinator sends the message across the ring again
5. Now servers process the message + send it to clients



GridGain

Discovery: Ring Message Delivery Protocol

1. A message starts “unverified”
2. Travels to the coordinator
   - Servers don’t actually “process” the message, just send it further
3. Coordinator verifies the message
   - Now the message is **ordered**
4. Coordinator sends the message across the ring again
5. Now servers process the message + send it to clients

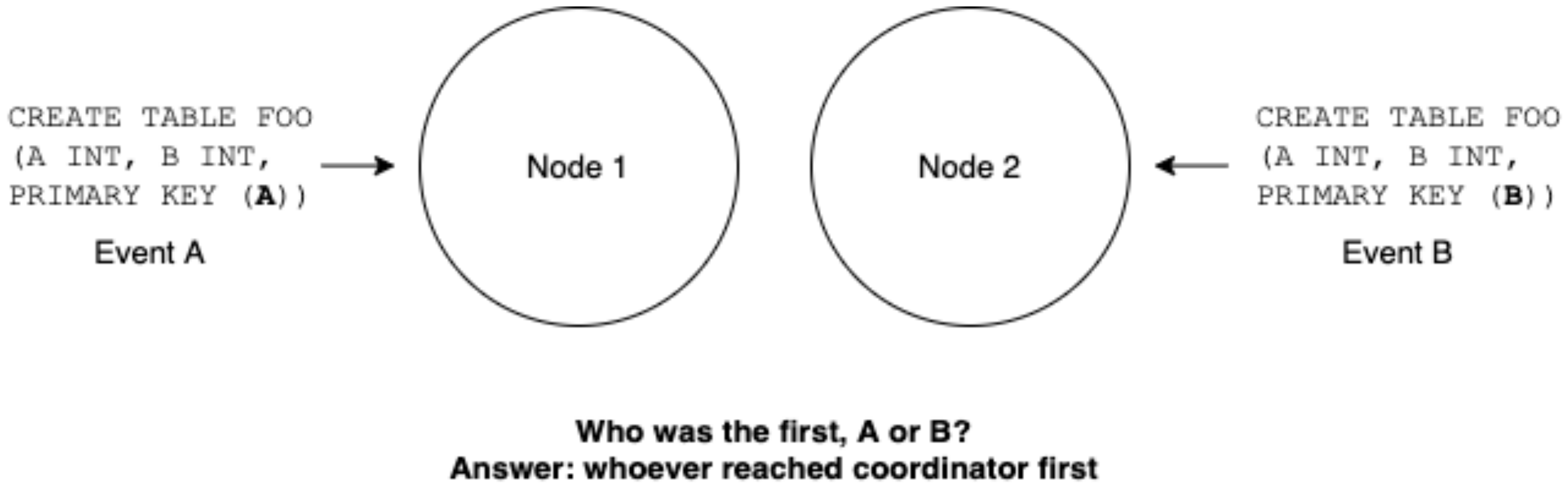


GridGain

Discovery: Ring Message Delivery Protocol

Key points

- Each message makes **two circles** around the ring
- Event A is considered to have happened **earilier** than event B if coordinator **verified A before it verified B**



Connecting to a Cluster

Connecting to a Cluster

What does a node need to do to connect to a cluster?

- Find a running cluster
  - Which IPs to connect to?
- Handshake
- Distribute information about itself in the cluster

GridGain

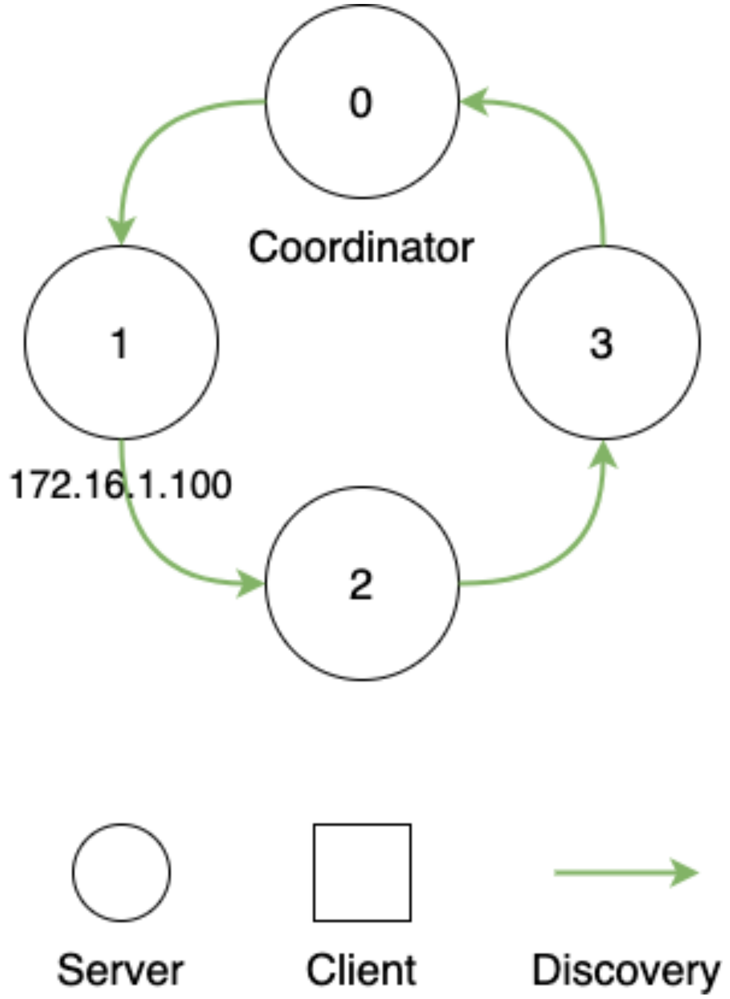
Discovery: IP Finder

A Discovery subcomponent that knows where to look for a cluster

- Provides a list of IPs to try to connect to via Discovery
- Has many implementations
  - Static – a hardcoded list of IPs
  - Multicast-based search
  - Shared file
  - Discovery via K8S and Cloud-specific tools

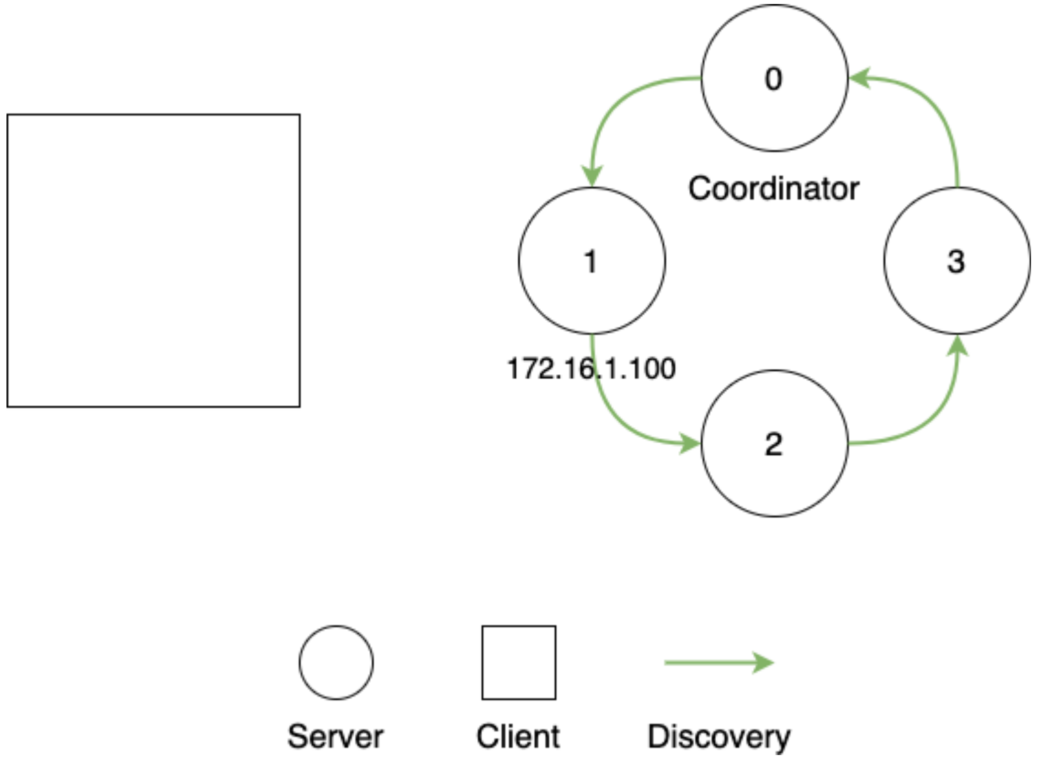
GridGain

Discovery: Cluster Join



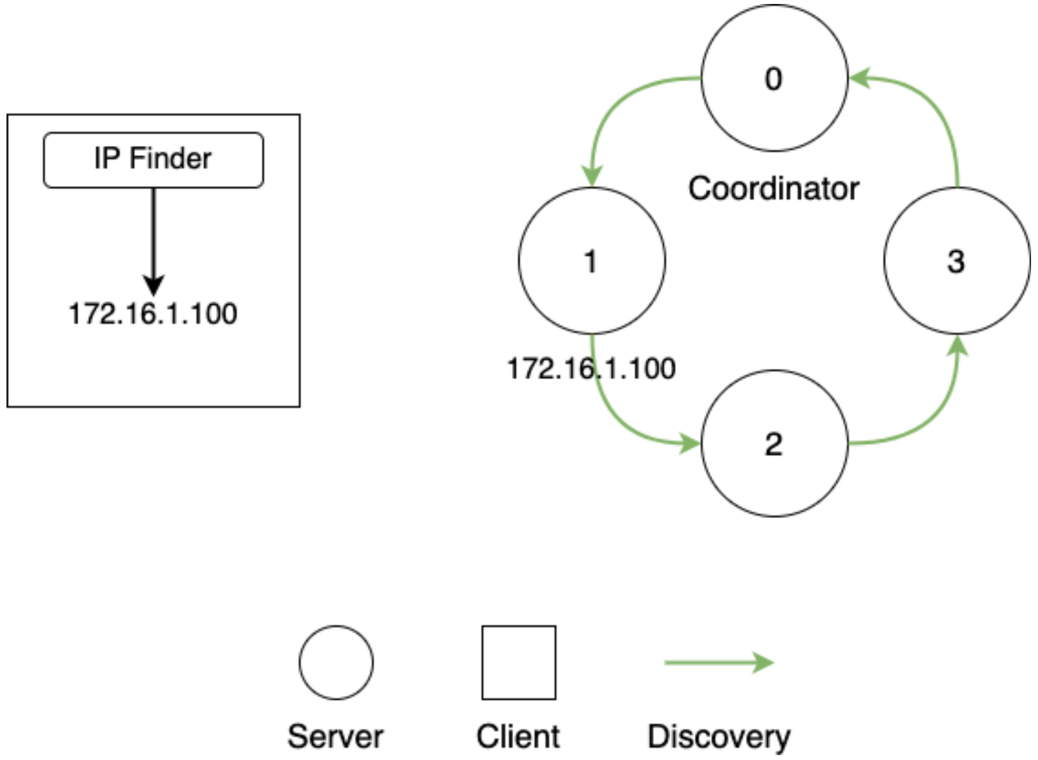
Discovery: Cluster Join

1. A node starts



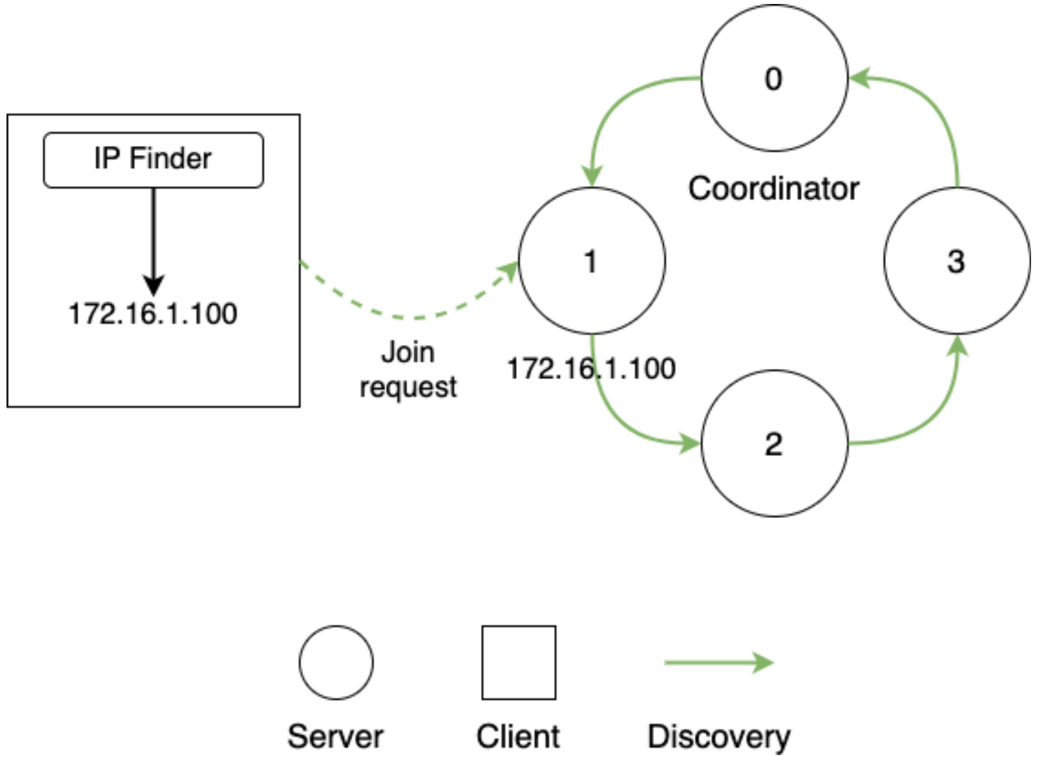
Discovery: Cluster Join

1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to



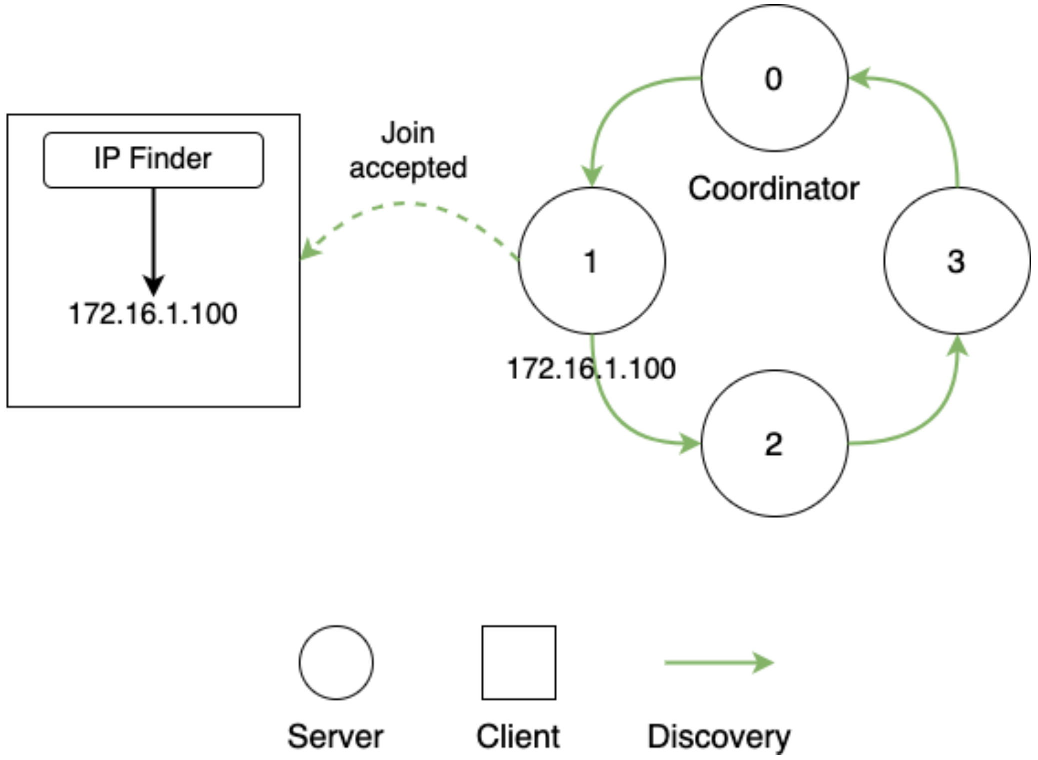
Discovery: Cluster Join

1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to
3. Node sends a join request



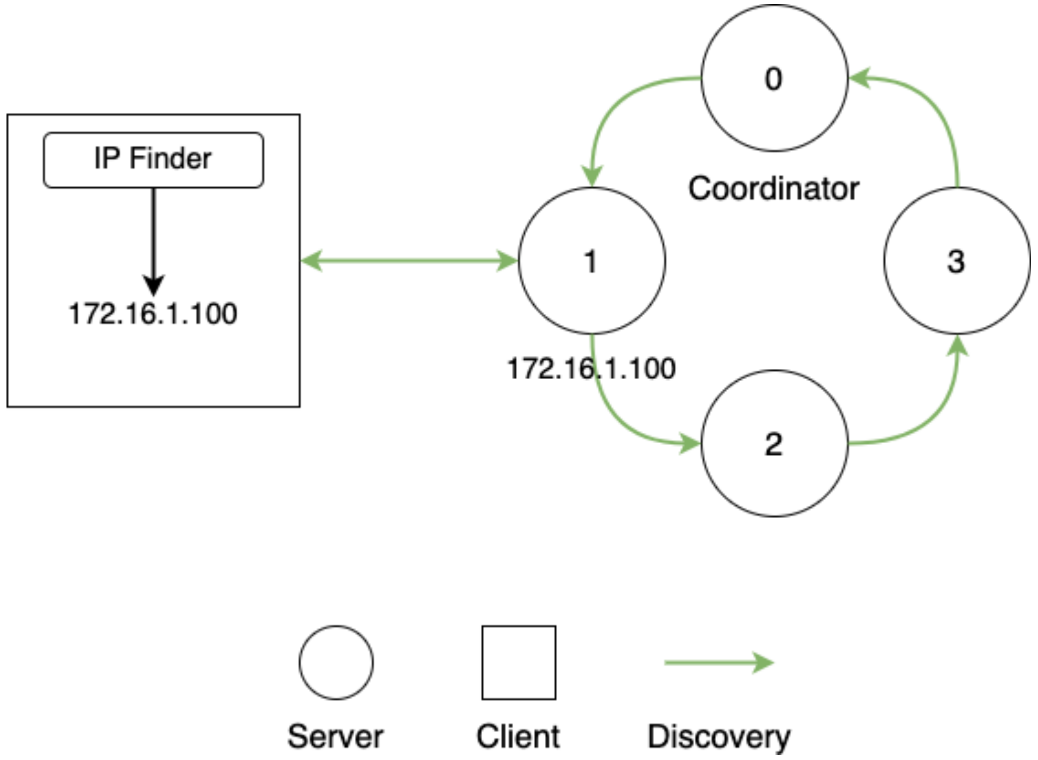
Discovery: Cluster Join

1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to
3. Node sends a join request
4. The server accepts the join



Discovery: Cluster Join

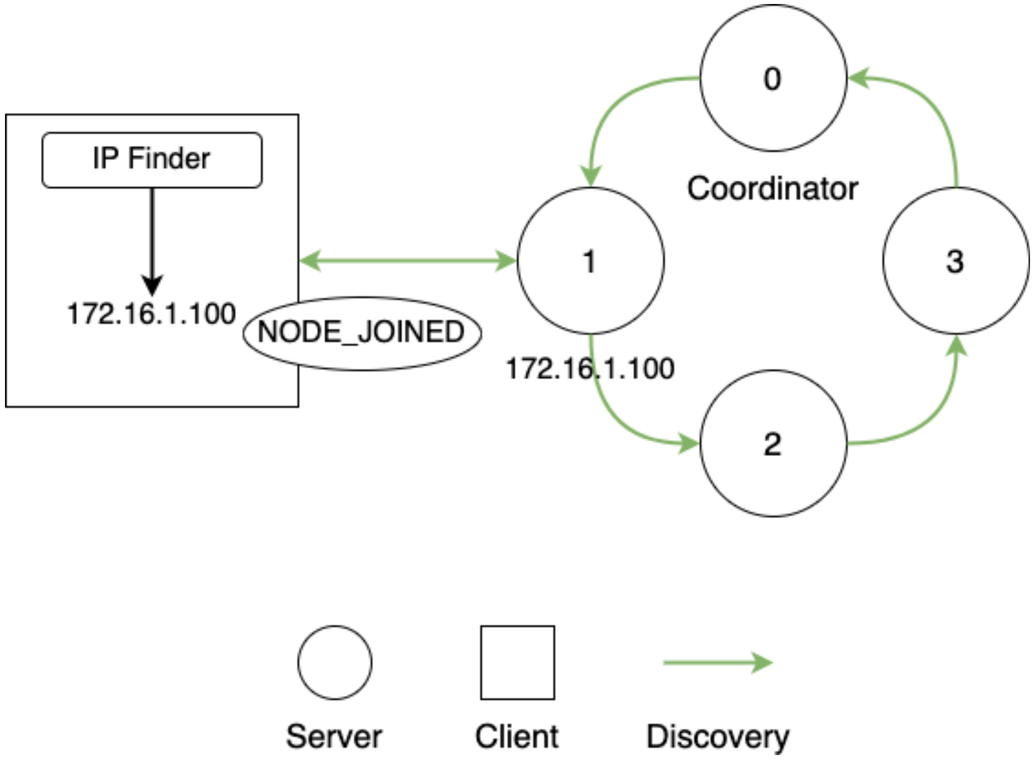
1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to
3. Node sends a join request
4. The server accepts the join
5. Discovery link is established



GridGain

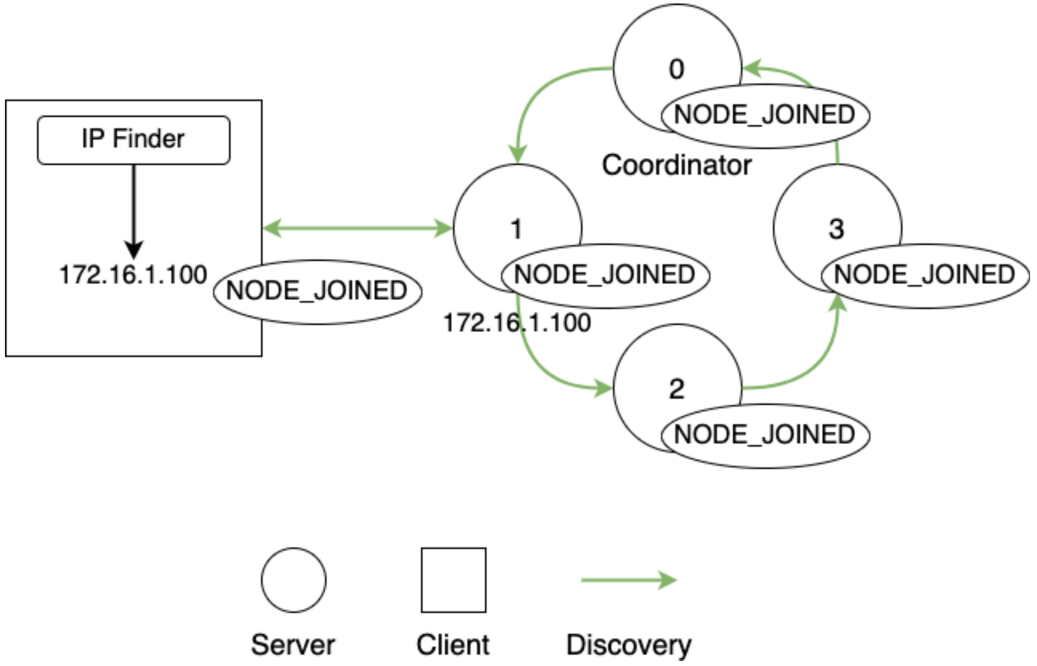
Discovery: Cluster Join

1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to
3. Node sends a join request
4. The server accepts the join
5. Discovery link is established
6. Discovery event NODE_JOINED with the node's details is sent
    - E.g. its addresses for Communication connections



Discovery: Cluster Join

1. A node starts
2. The node asks its IP Finder for an IP of a server to connect to
3. Node sends a join request
4. The server accepts the join
5. Discovery link is established
6. Discovery event NODE_JOINED with the node's details is sent
   - E.g. its addresses for Communication connections



Discovery: Cluster Join

Key points

- A new node finds a running server and establishes a Discovery link with it
- The Discovery link is used to send information about the new node to the entire cluster
  - Information includes address for Communication
  - The new node also receives information about the cluster
- After that, the new node can establish Communication links to do some actual work

Failure Detection

Failure Detection

What can we say about this function?

- Defines High Availability of the system
- What “failure” means?
  - “Idle” must not be treated as “failed”
  - Can be attributed to network reachability
- Must be detecting failures all the time

Discovery: Node Failure Detection

Discovery connectivity defines whether a node has failed

- Unreachable via Discovery = failed
- Has not been heard from via Discovery for X seconds = failed
- All nodes constantly send special heartbeat messages via Discovery
  - This way failure detection can work even if all nodes are idle and don't send other messages

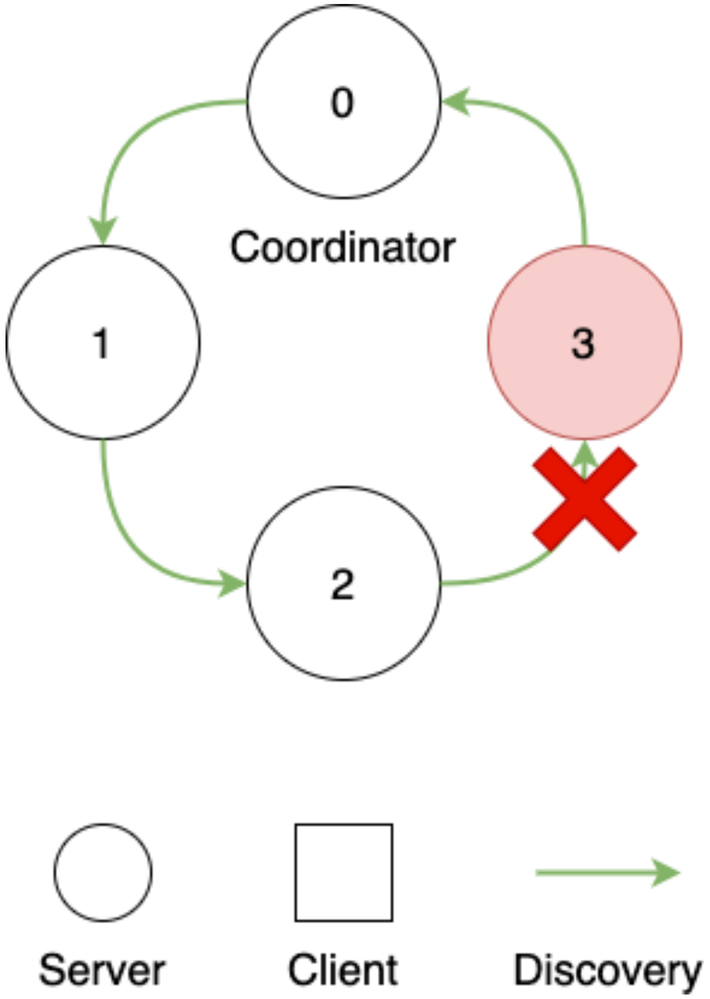
GridGain

Discovery: Node Failure Detection

Discovery: Node Failure Detection

Example 1

- Node 2 can't send a message to 3

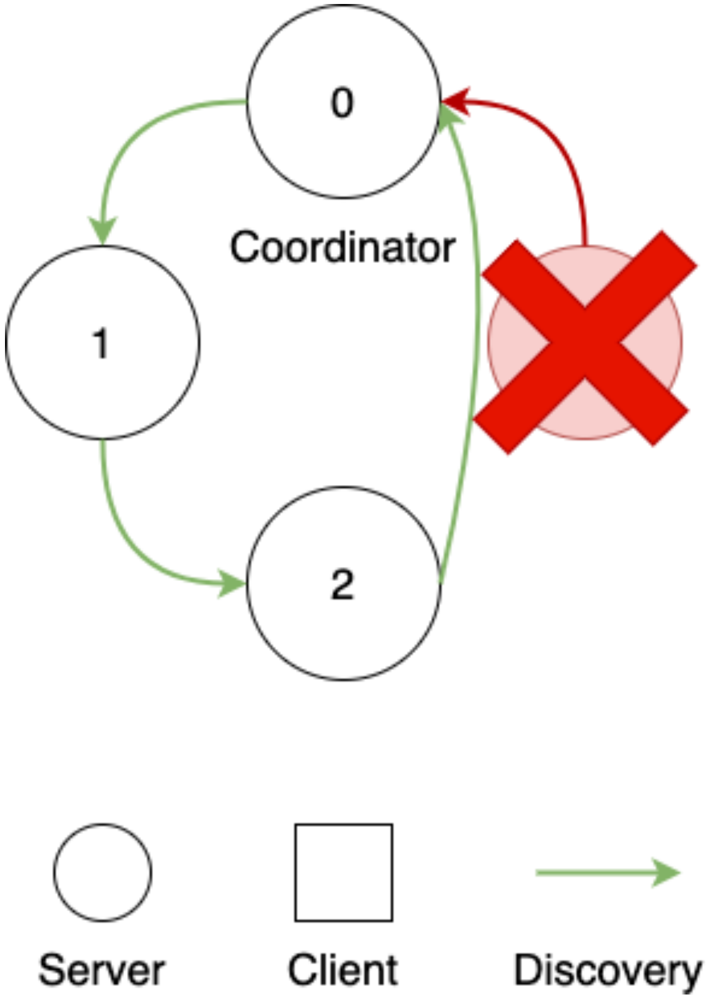


GridGain

Discovery: Node Failure Detection

Example 1

- Node 2 can’t send a message to 3
- Node 2 will bypass 3 and connect to 0, 3 will be removed

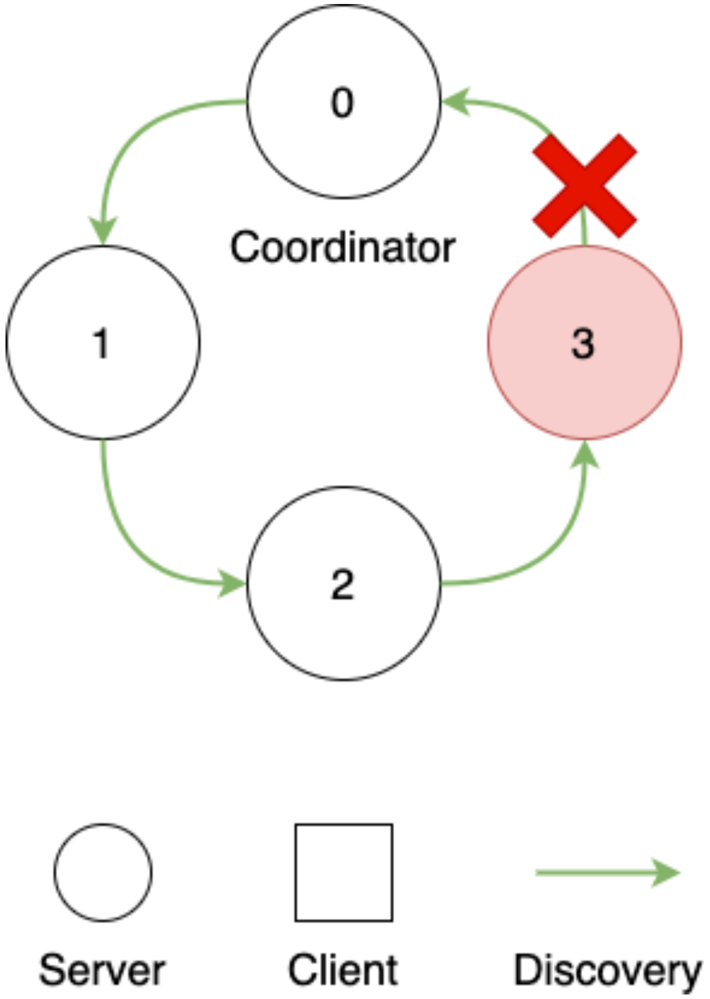


GridGain

Discovery: Node Failure Detection

Example 2

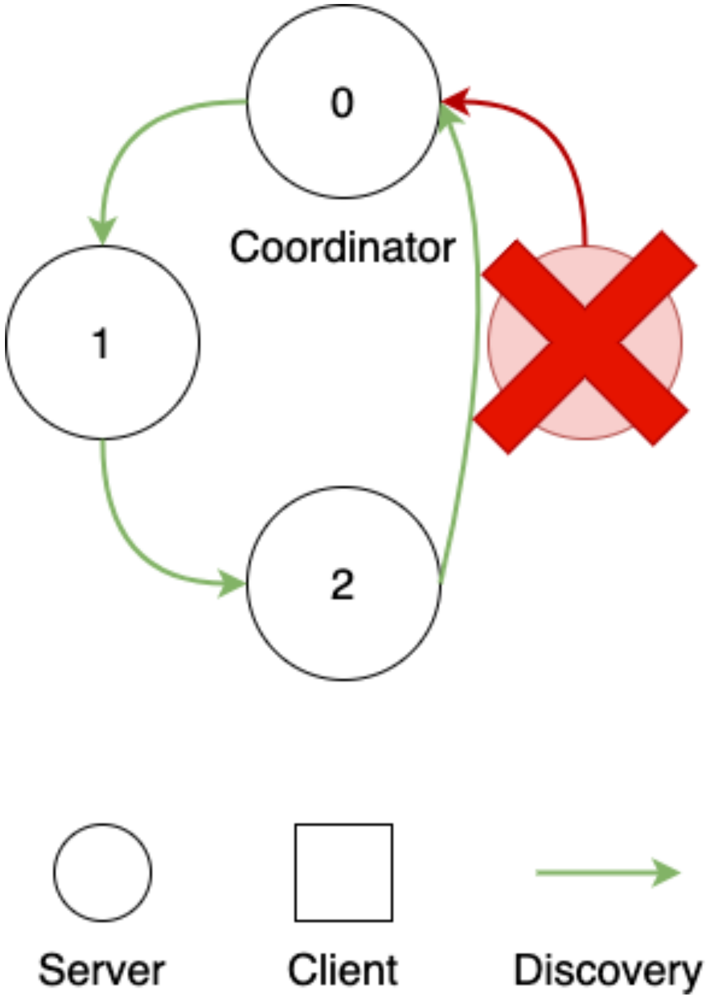
- Node 3 doesn't send messages to 0



Discovery: Node Failure Detection

Example 2

- Node 3 doesn’t send messages to 0
- 0 will eventually kick out 3 due to not receiving heartbeats



Discovery vs Communication

Discovery vs Communication

Discovery	Communication
Both are Ignite networking subsystems	
Delivers system cluster-wide **events** Allows nodes to **join** Detects **failures**	Fast **data transfer** between two nodes
Key – global **order** of events	Key – **speed**
Single-threaded	Multi-threaded
Complex topology	**Fully connected** peer-to-peer topology
Servers and clients are different **Coordinator** is a special server	Servers and clients are equal

Advanced Topics

Large Clusters and The Ring

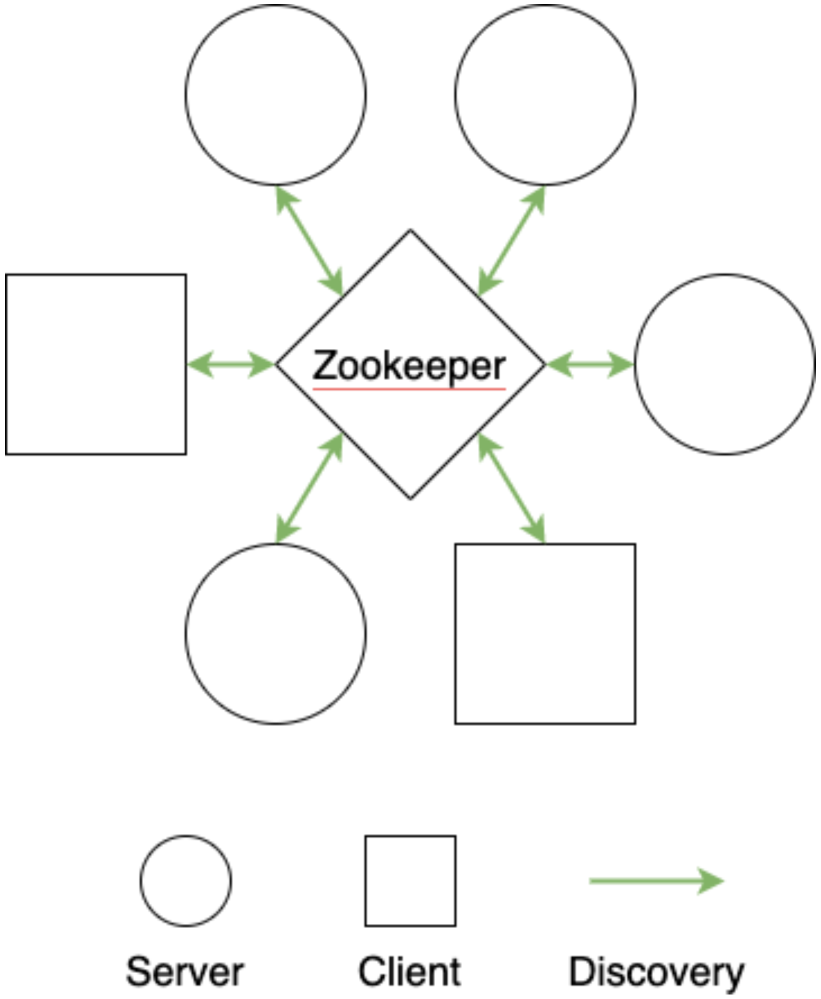
What is the downside of the ring topology?

- Large cluster means large Discovery rings
- Discovery messages need up to two full circles to be delivered to all nodes
- Result: huge latencies in Discovery on topologies with hundreds and thousands of nodes

Large Clusters and The Ring

Alternative Discovery implementation for large clusters

- Zookeeper-based Discovery uses star-shaped topology
- Can handle much larger clusters + some bonuses
- Downside: requires a Zookeeper installation



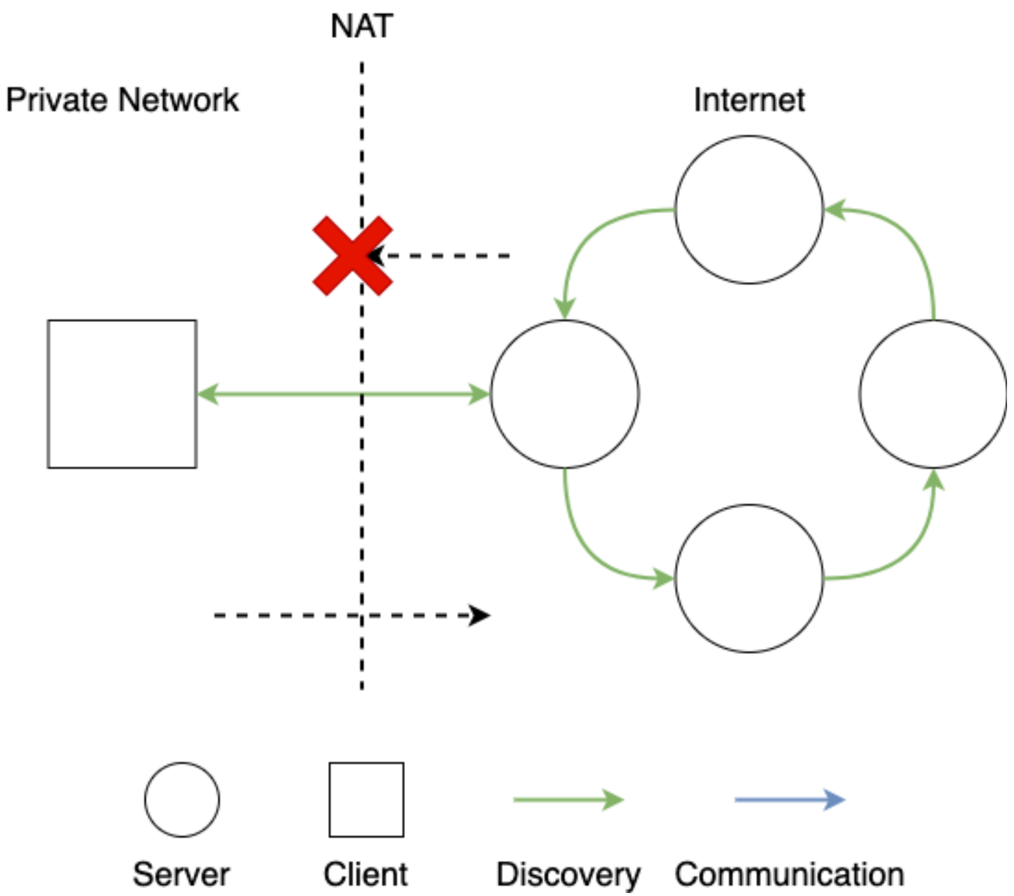
One-Way Connectivity

Example

One-Way Connectivity

Example

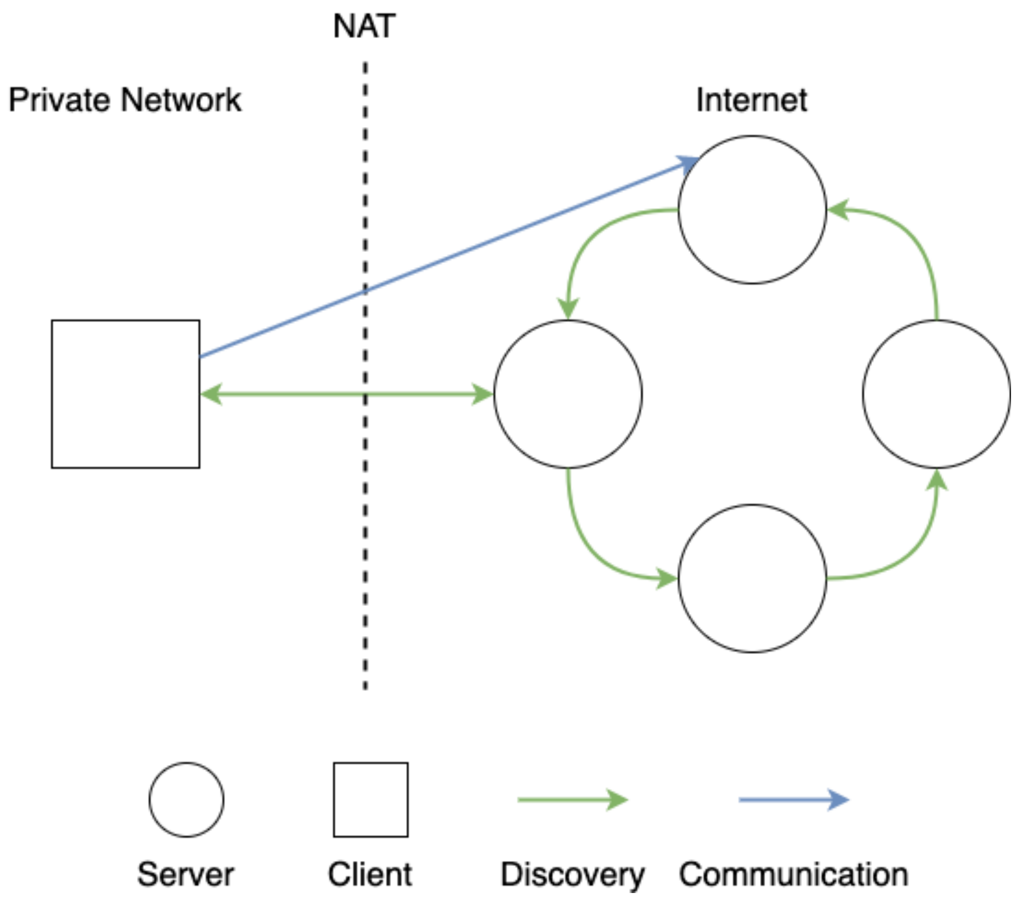
1. Client is behind a NAT



One-Way Connectivity

Example

1. Client is behind a NAT
2. Client connects to servers – all good

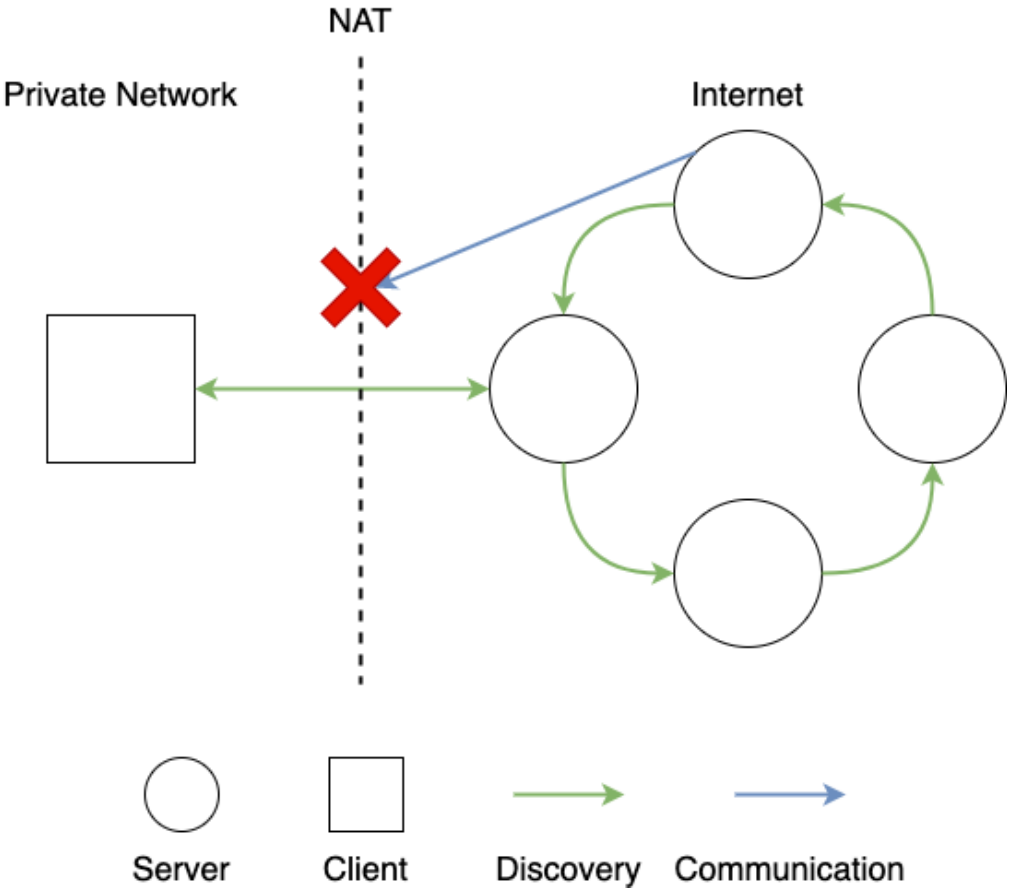


One-Way Connectivity

Example

1. Client is behind a NAT
2. Client connects to servers – all good
3. Server tries to connect to the client – connection fails

This can lead to various issues up to complete cluster freeze.

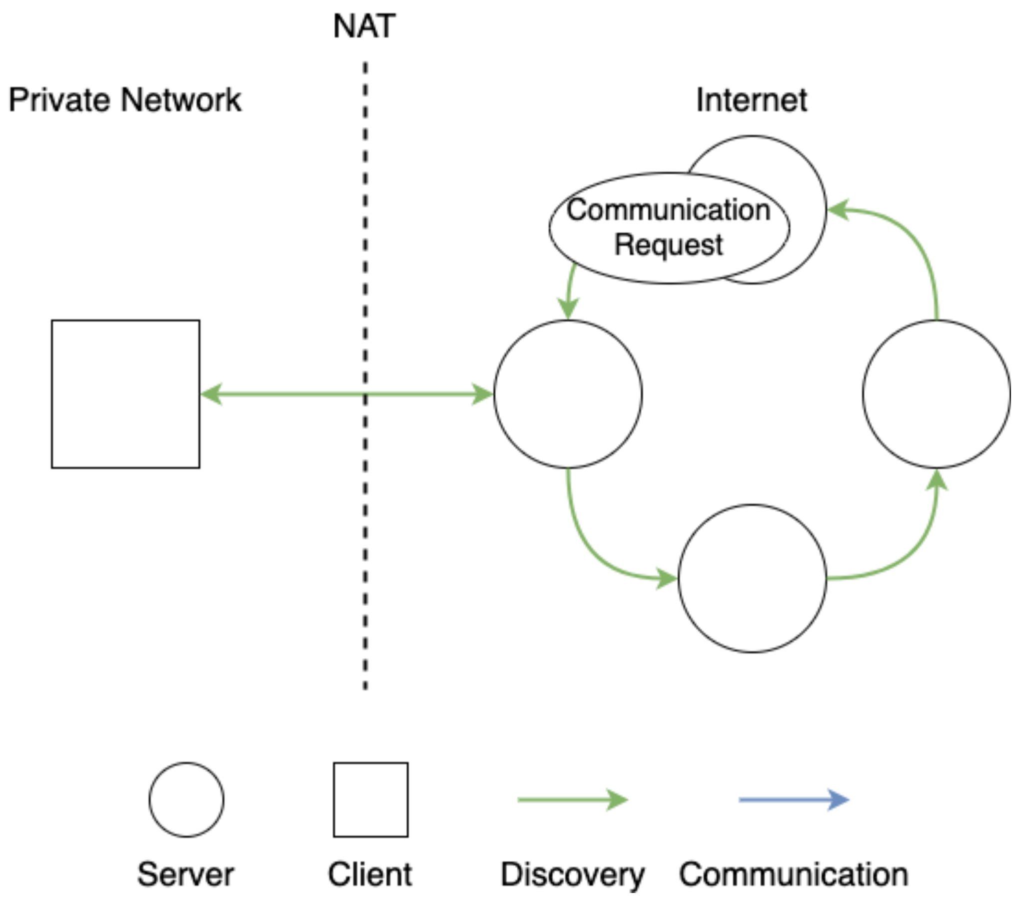


GridGain

One-Way Connectivity

Solution – leverage Discovery

1. Server sends a special Discovery message to the client – a request to open a Communication link

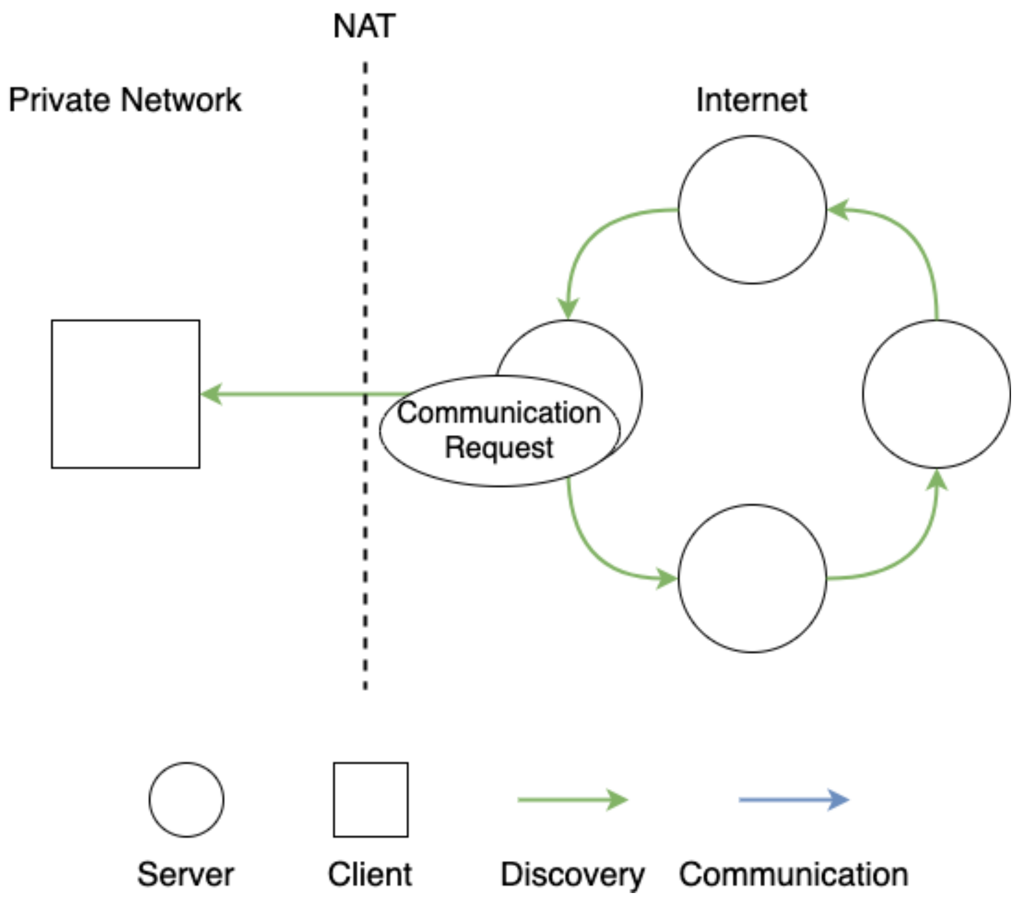


GridGain

One-Way Connectivity

Solution – leverage Discovery

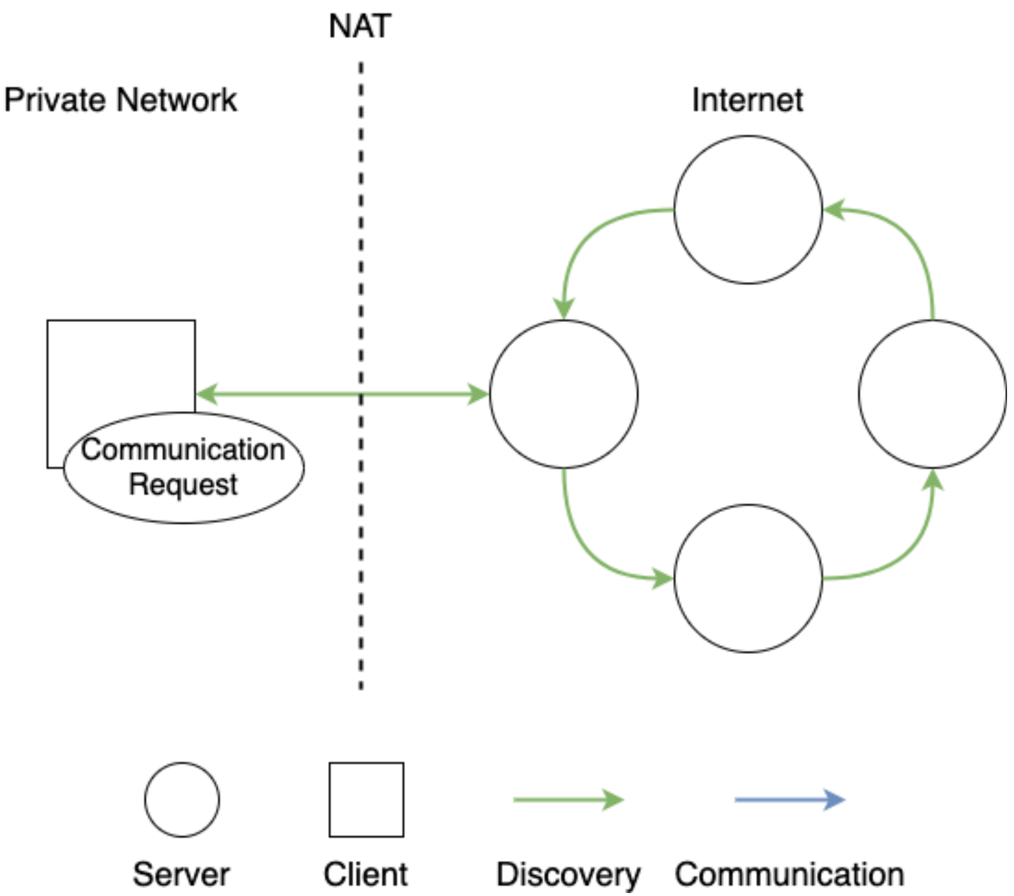
1. Server sends a special Discovery message to the client – a request to open a Communication link



One-Way Connectivity

Solution – leverage Discovery

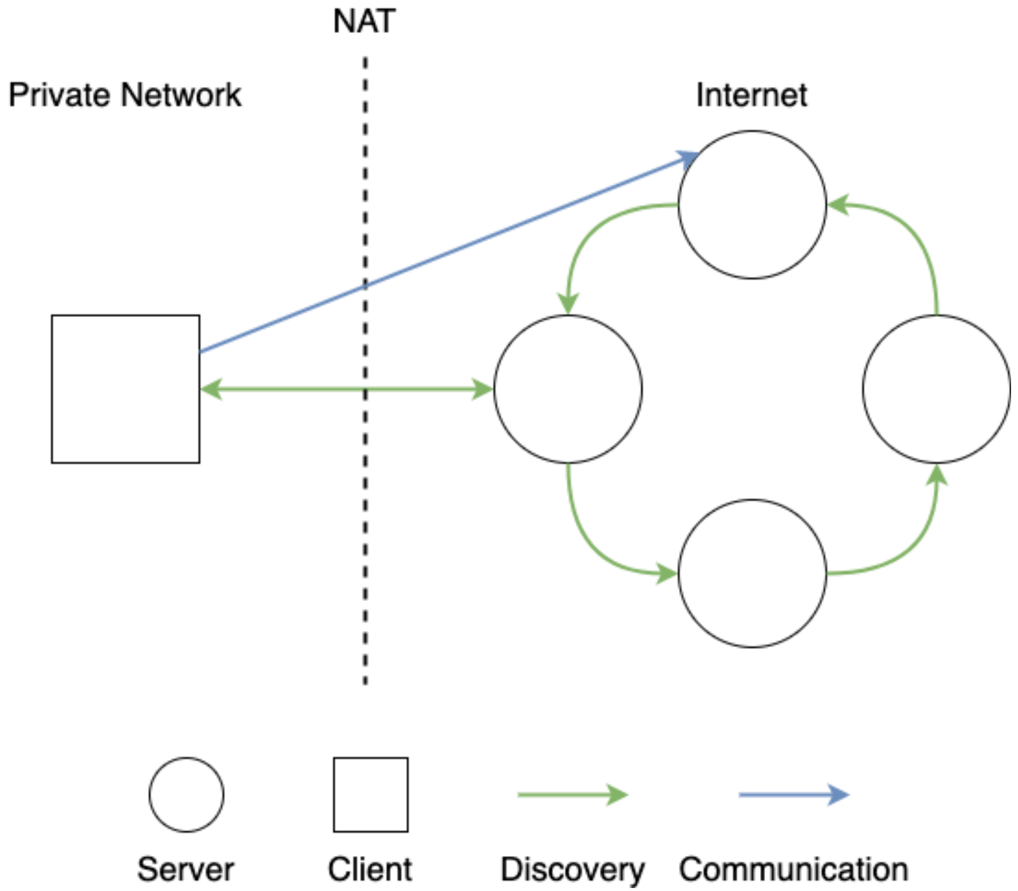
1. Server sends a special Discovery message to the client – a request to open a Communication link



One-Way Connectivity

Solution – leverage Discovery

1. Server sends a special Discovery message to the client – a request to open a Communication link
2. Client successfully opens the link from its side



Q&A

GridGain

Q&A

- Join Apache Ignite Community!
- Great Way to Learn Distributed Storages, Computing, SQL, ML, Transactions
- How To Contribute:
  - https://ignite.apache.org/community/contribute.html
- Join Ignite Meetups:
  - https://ignite.apache.org/meetup-groups.html