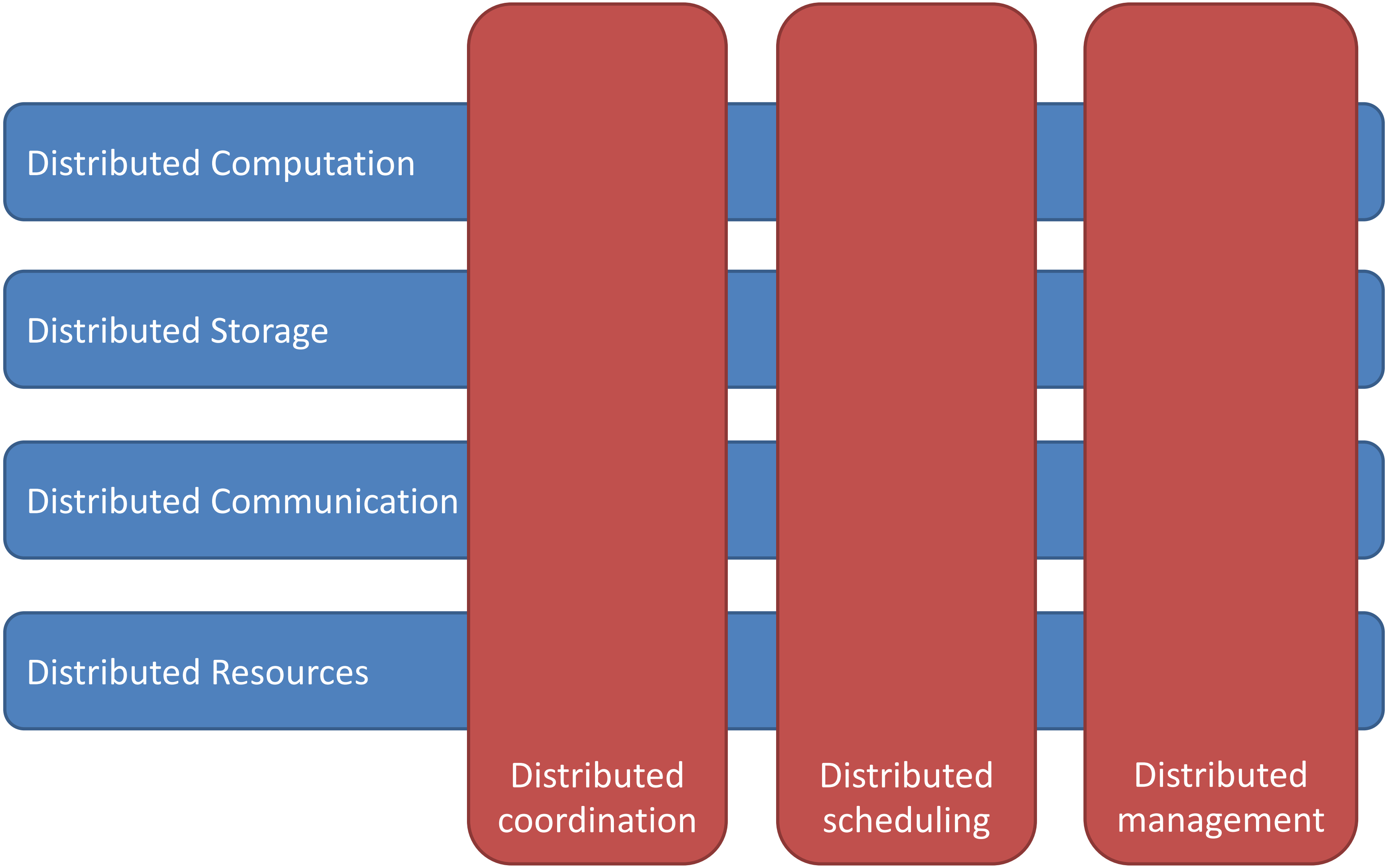
Overview



Distributed Computation

Distributed Storage

Distributed Communication

Distributed Resources

Distributed coordination

Distributed scheduling

Distributed management

- Distributed coordination and synchronization:
  - Distributed mutex, distributed election, distributed consensus, distributed transaction, distributed locks
- Distributed management and resources
  - Centralized structure, decentralized structure, scheduling
- Distributed computation
  - MapReduce, Spark
- Distributed communication
  - RPC, publish and subscribe, message queue
- Distributed storage
  - CAP, distributed storage, distributed cache

CS 7172
Parallel and Distributed Computation

Distributed Mutex

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Outline

- Computer networks, primarily from an application perspective
- Protocol layering
- Client-server architecture
- End-to-end principle
- TCP
- Socket programming

What is Distributed Mutex?

- Suppose you are making coffee at Starbucks, and someone takes away your cup, some other takes away the coffee machine

- Ideal: you want to keep using the machine and cup without interference

What is Distributed Mutex?

- Like the coffee machine, in distributed system, for the same shared resource, one program does not want to be disturbed by other programs while it is being used.

- This requires that only one program can access this resource at a time

What is Distributed Mutex?

- In a distributed system, the method to achieve access to exclusive resource is called ***Distributed Mutual Exclusion***

- The shared resource that is accessed by mutual exclusion is called ***Critical Resource***

Method 1: Centralized algorithm

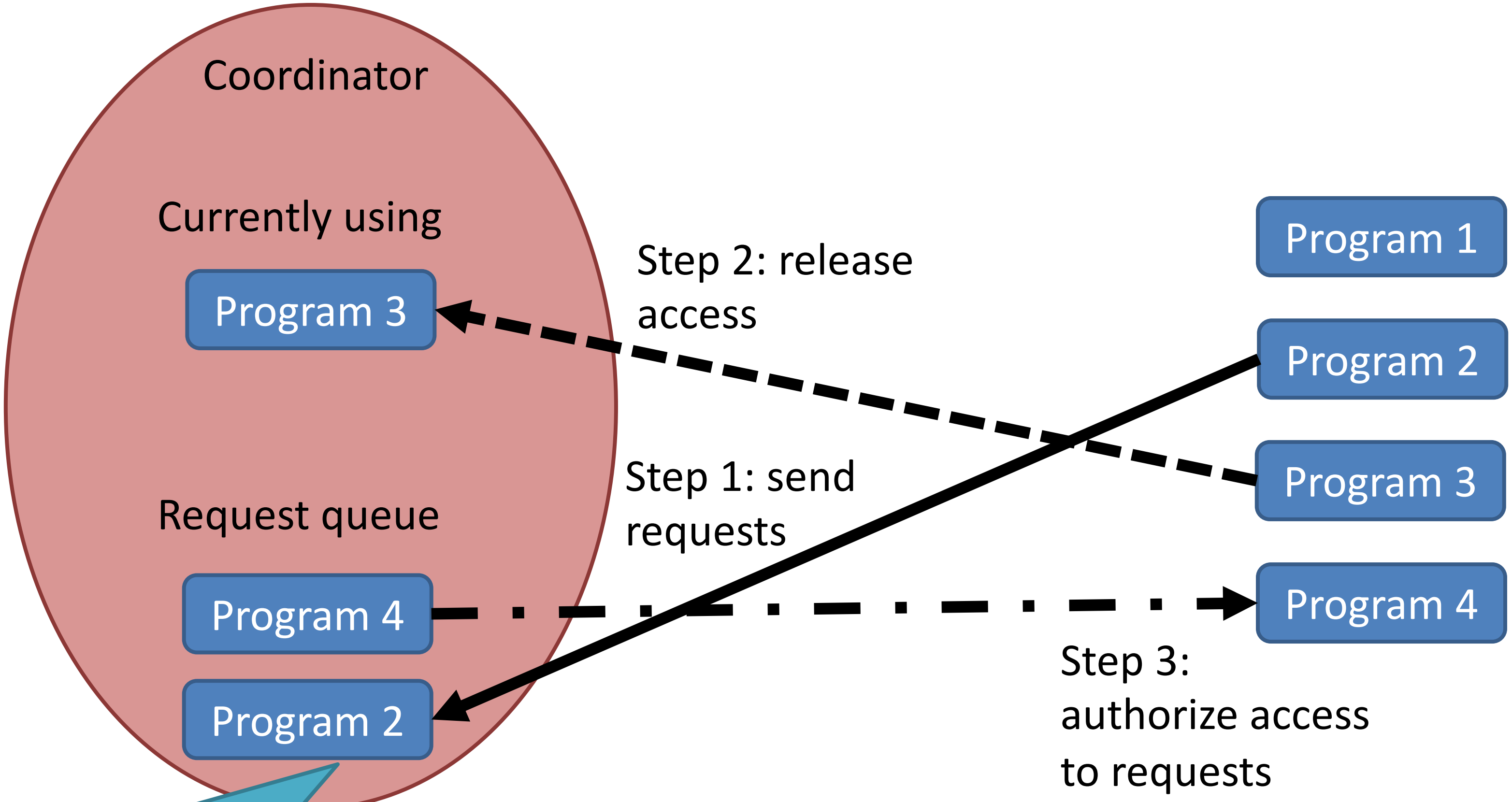
Add a "Coordinator" to restrict everyone to use self-service coffee machines in order to solve the problem of forcibly interrupting others

Method 1: Centralized algorithm

- How centralized algorithm works?
  - Introduce a coordinator program for distributed mutex.
  - Every time a program needs to access critical resources, it first sends a request to the coordinator. If no program currently uses this resource, the coordinator authorizes the requesting program to access it; otherwise, the requesting program is served in a first-come-first-served order.
  - If a program finishes accessing the resource, the coordinator is notified, and pick the first request from the queue and authorizes the program to access critical resources

Method 1: Centralized algorithm

- Centralized algorithm is also named as Central Server algorithm in distributed system



The coordinator puts program 2 and 4 into the waiting queue based on their request time

Method 1: Centralized algorithm

- A program to complete a critical resource access requires the following processes:
  1. sending request to the coordinator;
  2. The coordinator issues authorization to the program;
  3. After the program uses the critical resource, send release notification to the coordinator.

- One request requires *three* interaction between the program and the coordinator

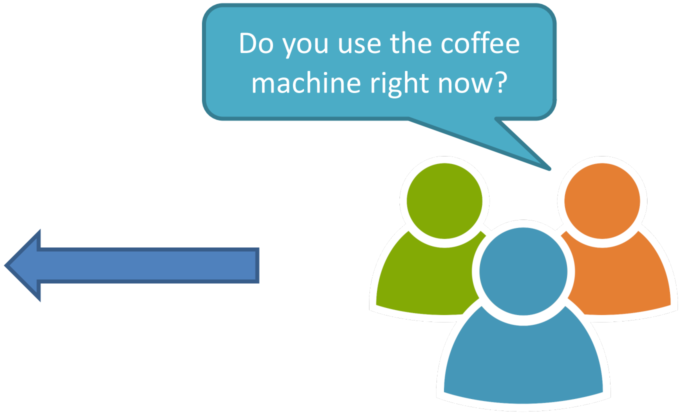
Method 1: Centralized algorithm

- Advantages:
  - Intuitive and simple
  - Less information interaction
  - Easy to implement
  - All programs need only communicate with the coordinator, no communication is required between programs

Method 1: Centralized algorithm

- Disadvantages:
  - The coordinator will become the performance bottleneck of the system
    - If there are 100 programs accessing critical resources, the coordinator has to process 100 * 3 = 300 messages. The number of messages processed by the coordinator increases linearly with the number of programs that need to access critical resources
  - It is easy to cause a single point failure. Poor reliability.
    - The failure of the coordinator will make all programs lose access to critical resources and the entire system unavailable.

Method 2: Distributed algorithm



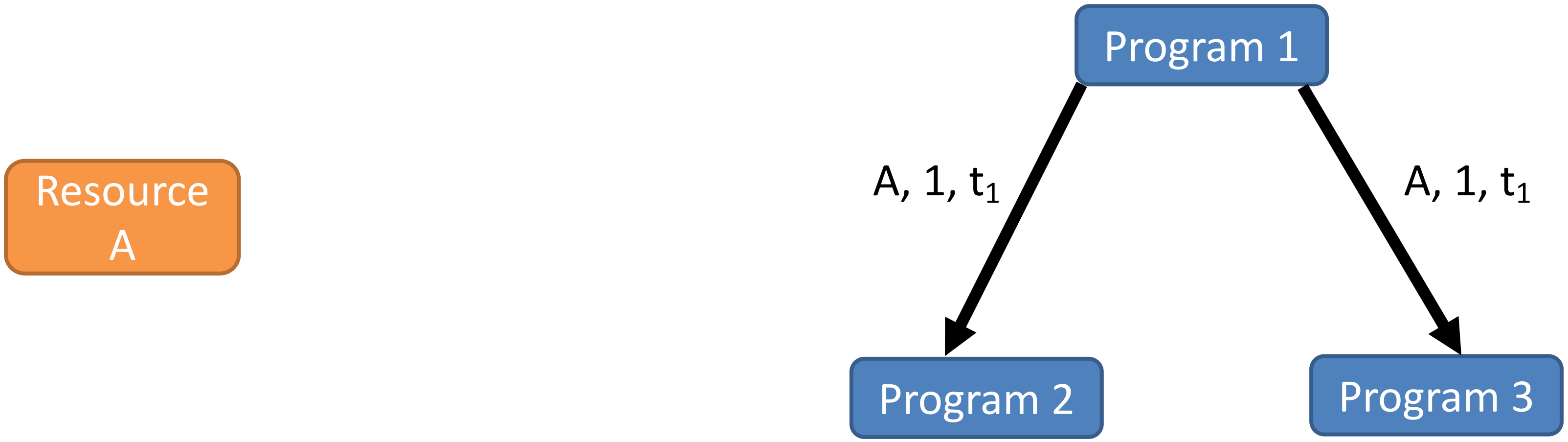
Do you use the coffee machine right now?

When you need to use a self-service coffee machine, you can ask other people first. When confirming that no other people are using, you can make your coffee.

Method 2: Distributed algorithm

- How distributed algorithm works?
  - When a program wants to access a critical resource, it first sends a request message to other programs in the system.
  - After receiving the messages returned by all programs that no programs are using the resource, it can access the critical resource.
  - The request message includes the requested resource (which), the requester's ID (who), and the time the request was made (when).

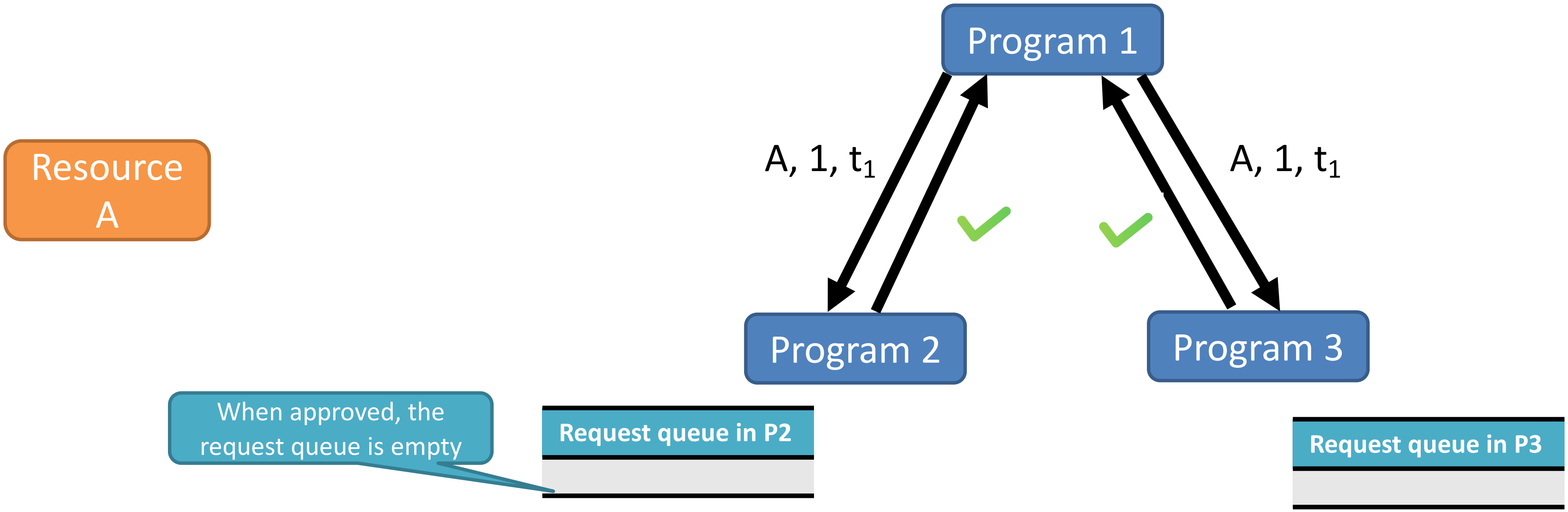
Method 2: Distributed algorithm



Request queue in P2
A, 1, t_1

Request queue in P3
A, 1, t_1

Method 2: Distributed algorithm



Program 1

A, 1, t_1

✓

✓

A, 1, t_1

Resource A

Program 2

Program 3

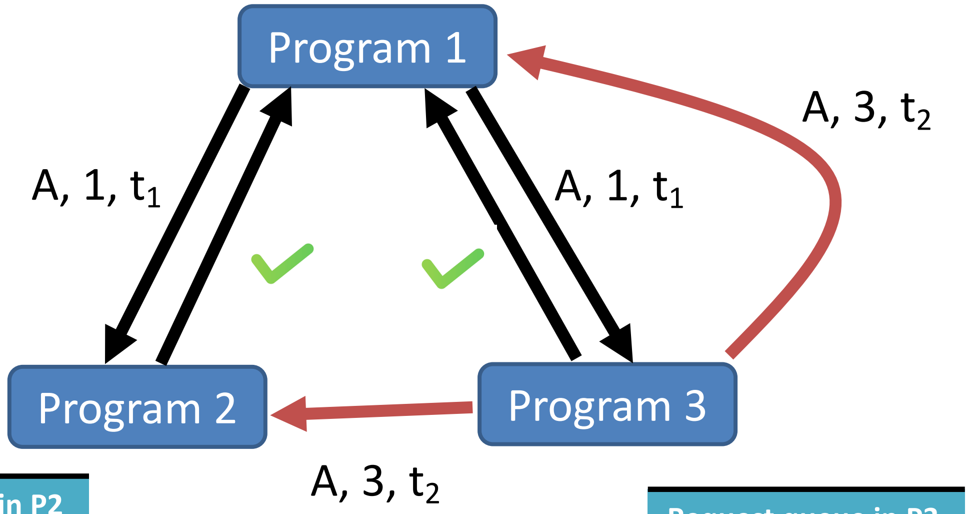
When approved, the request queue is empty

Request queue in P2

Request queue in P3

Method 2: Distributed algorithm

Request queue in P1
A, 3, t_2

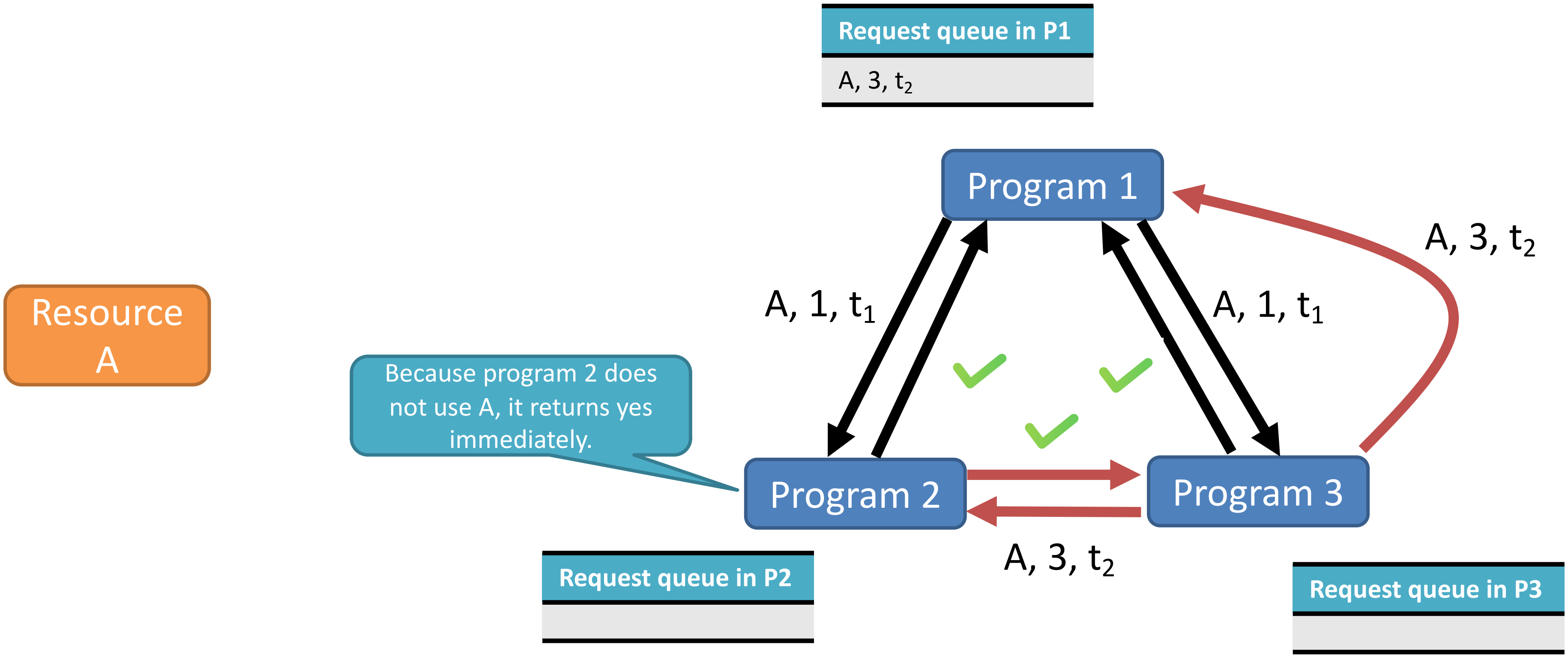


Resource A

Request queue in P2
A, 3, t_2

Request queue in P3

Method 2: Distributed algorithm



Request queue in P1
A, 3, t_2

Program 1

Program 2

Program 3

Resource A

Because program 2 does not use A, it returns yes immediately.

A, 1, t_1

A, 1, t_1

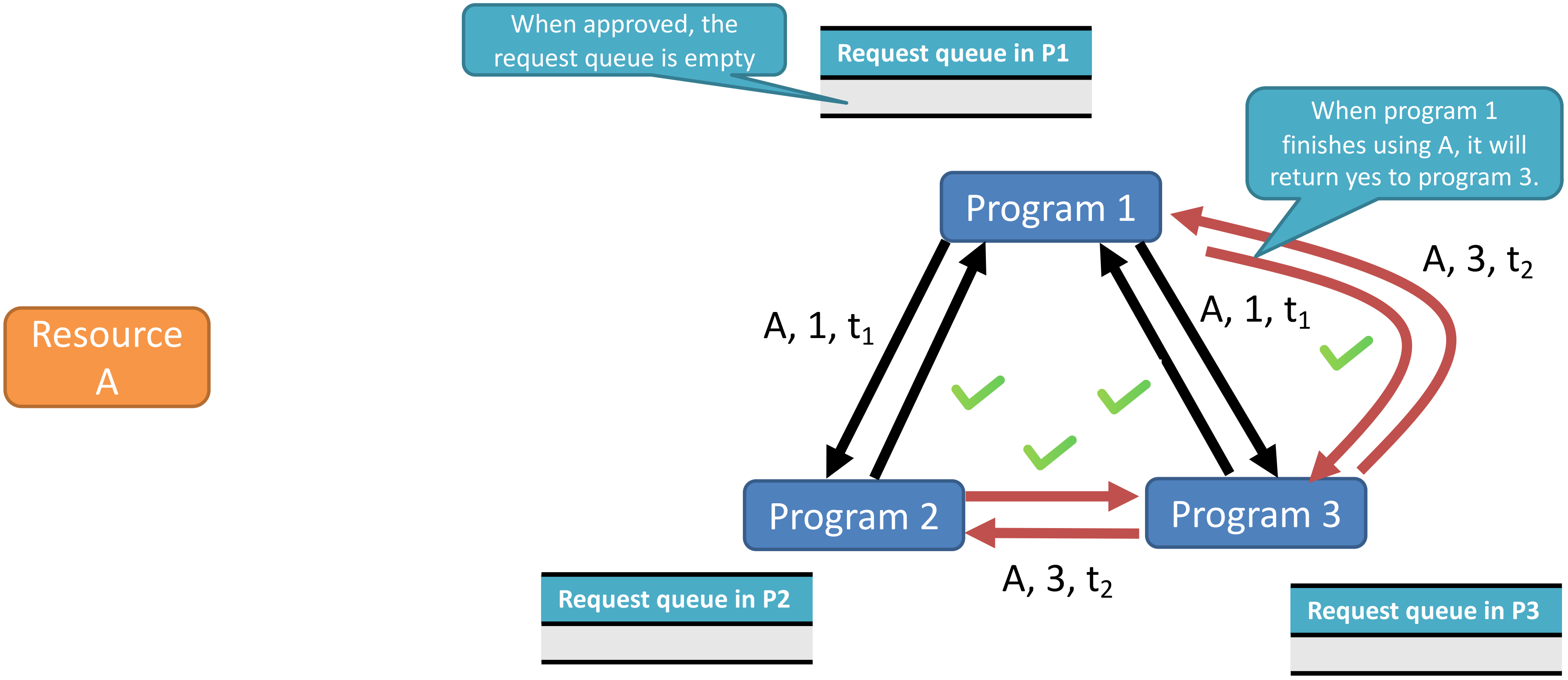
A, 3, t_2

A, 3, t_2

Request queue in P2

Request queue in P3

Method 2: Distributed algorithm



When approved, the request queue is empty

When program 1 finishes using A, it will return yes to program 3.

Method 2: Distributed algorithm

- A program to complete a critical resource access requires the following processes:
  1. Send request to N-1 programs in the system;
  2. After receiving permissions from N-1 programs, it can access the critical resources;

- One request requires *2*(n-1)* interaction between the program and the coordinator

Method 2: Distributed algorithm

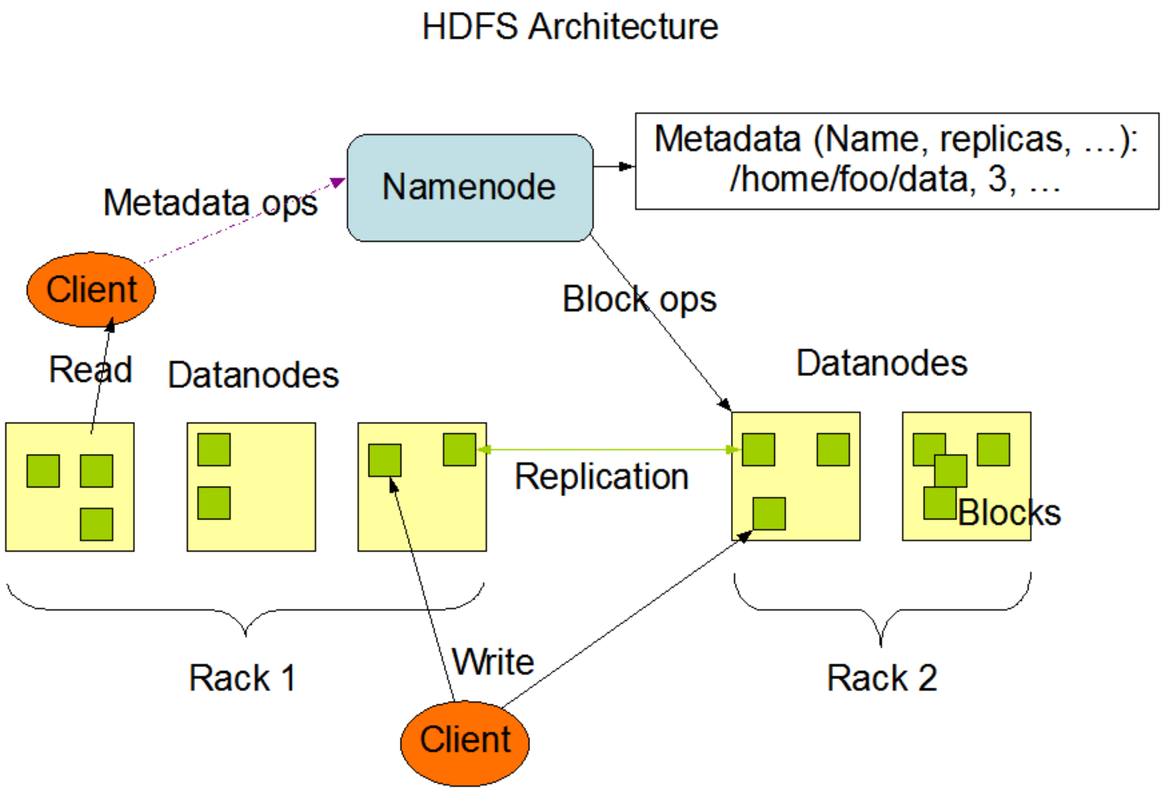
- Advantages:
  - Simple
  - Easy to implement

Method 2: Distributed algorithm

- Disadvantages:
  - The number of messages will increase exponentially with the number of programs that need to access critical resources, leading to high "communication costs"
    - n programs accessing to critical resources will produce 2n(n-1) messages
  - Once a program fails and the confirmation message cannot be sent, other programs are in a state of waiting for a reply, making the entire system unusable

Scenario using Distributed Algorithm

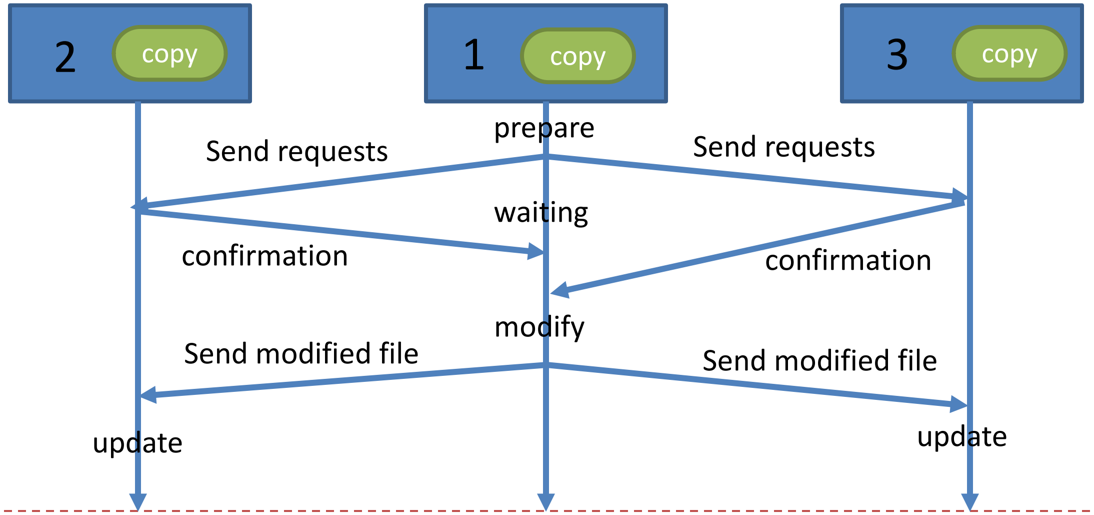
- Hadoop HDFS: a distributed file system across multiple nodes
- To achieve high reliability, one file has multiple copies on different nodes



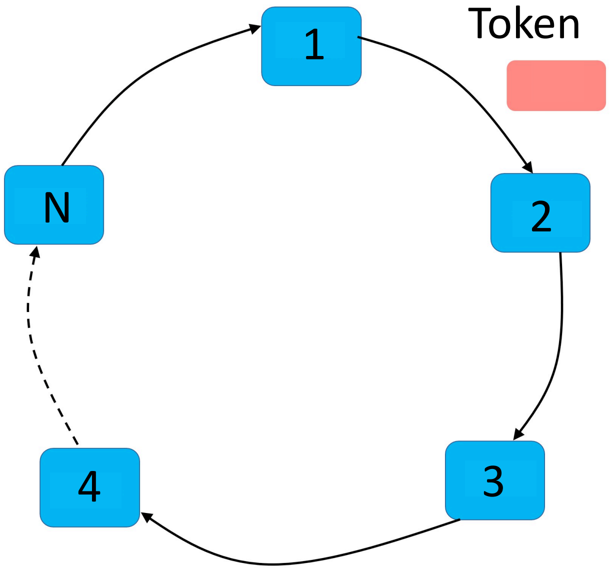
Scenario using Distributed Algorithm

- Suppose node 1, 2, 3 have the copies of the same file. When the node 1 needs to modify the file:

1. Node 1 sends modification requests to node 2 and 3
2. If node 2 and 3 do not use the file, approve the requests;
3. If node 1 receives confirmation messages, modify the file;
4. After modification, node 2 and 3 send confirmation messages to node 2 and 3, and modified file data;
5. When node 2 and 3 receive the modified data, update the local copies

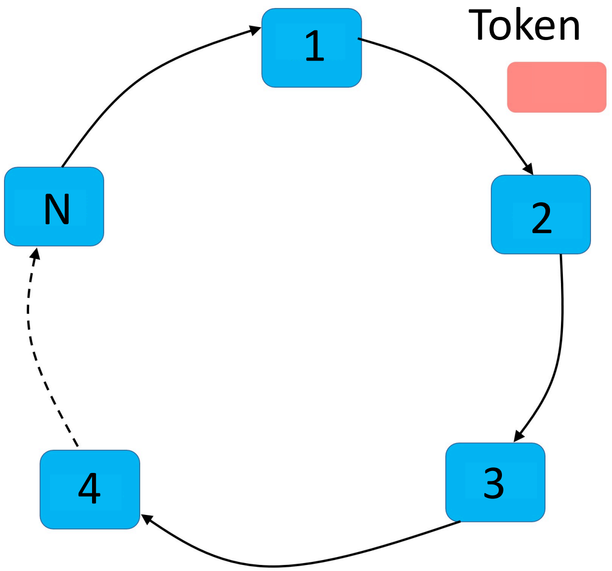


Method 3: Token Ring Algorithm



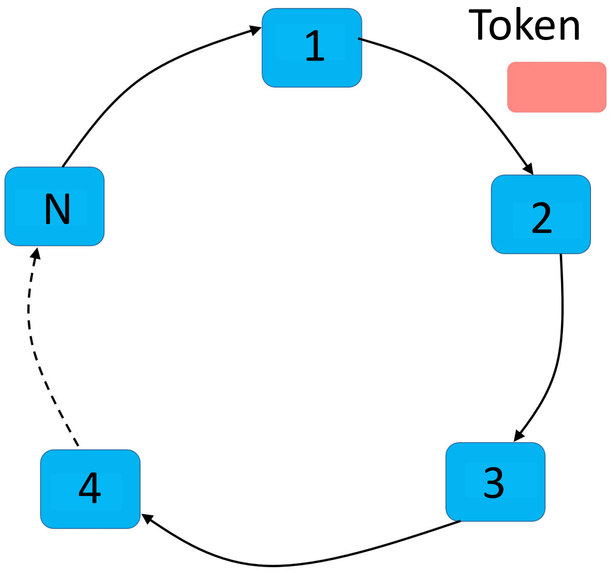
- How token ring algorithm works?
  - All programs form a ring structure. Tokens are passed between programs in a clockwise (or counterclockwise) direction.
  - The program that receives the token has the right to access critical resources. After the access is completed, the token is transferred to the next program.
  - If the program does not need to access critical resources, just passes the token to the next program

Method 3: Token Ring Algorithm



- Advantages:
  - Before using critical resources, it is not necessary to ask the opinions of other programs one by one like a distributed algorithm, so it has higher efficiency
  - Within a period, each program can access critical resources, so the fairness is good

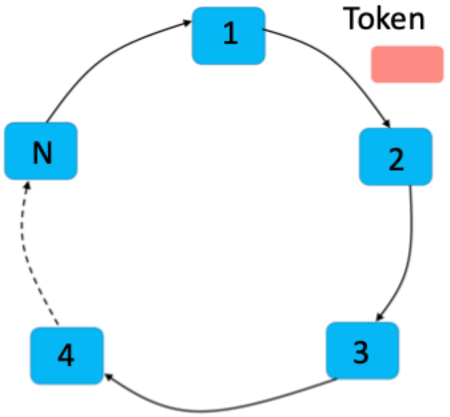
Method 3: Token Ring Algorithm



- Disadvantages:
  - Regardless of whether the program in the ring needs to access the resource, it has to receive and pass the token, so it will bring some invalid communication.
    - Assume that there are 100 programs in the system. After program 1 accesses the resources, even if the other 99 programs do not need access, the tokens must be re-accessed after the 99 other programs are passed, which increases the latency of the system.

Scenario using Token Ring Algorithm

- Walkie-talkie:
  - Can send or receive messages
  - Every time only one walkie-talkie can send
  - The walkie-talkie that holds the token can send and others just receive


1
Token
2
3
4
N