Caltech, CMS. CS/IDS 142: Lecture 6.1

Distributed Dining Philosophers

Mani Chandy

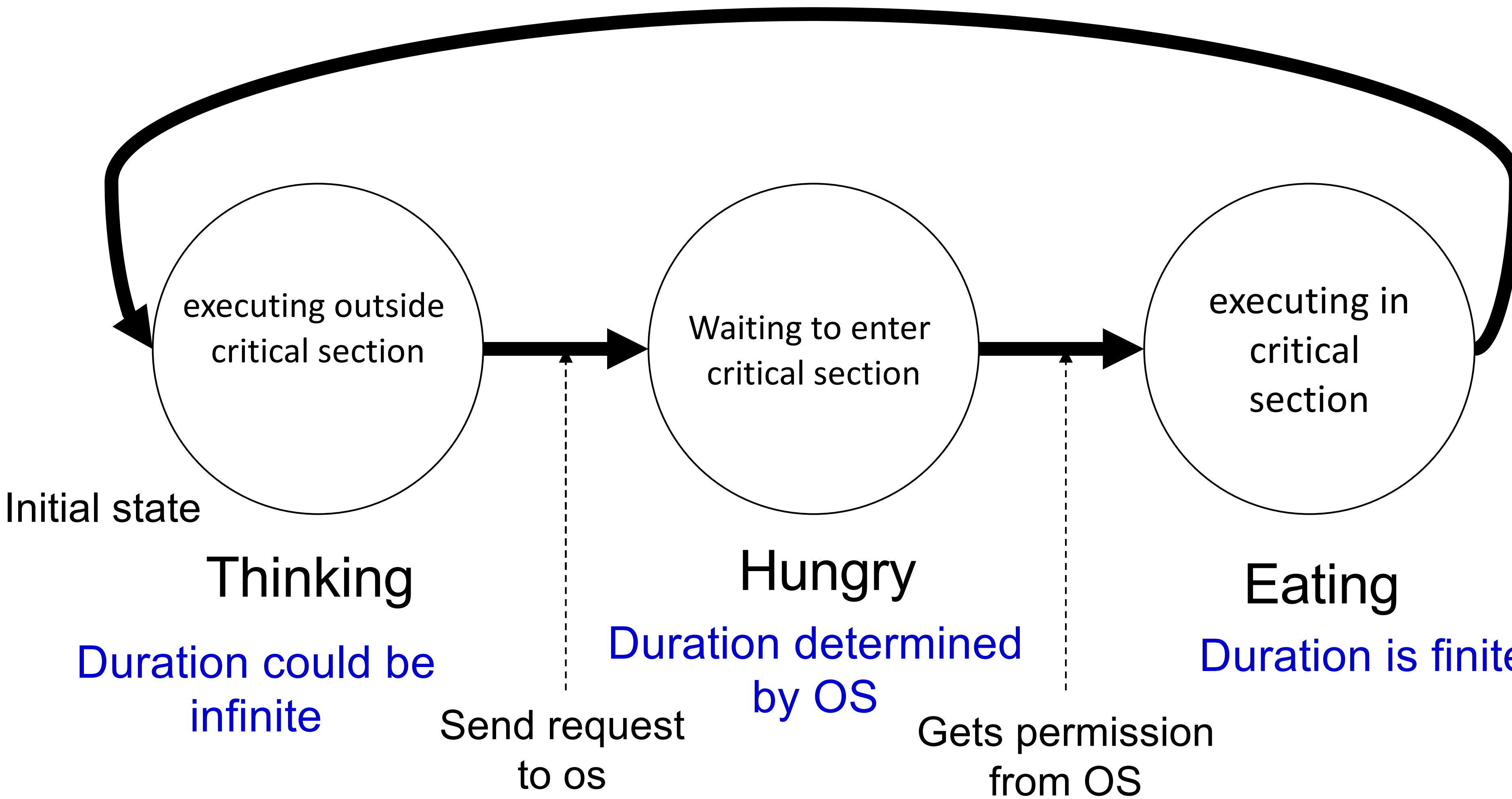
4 November 2019

Goals:

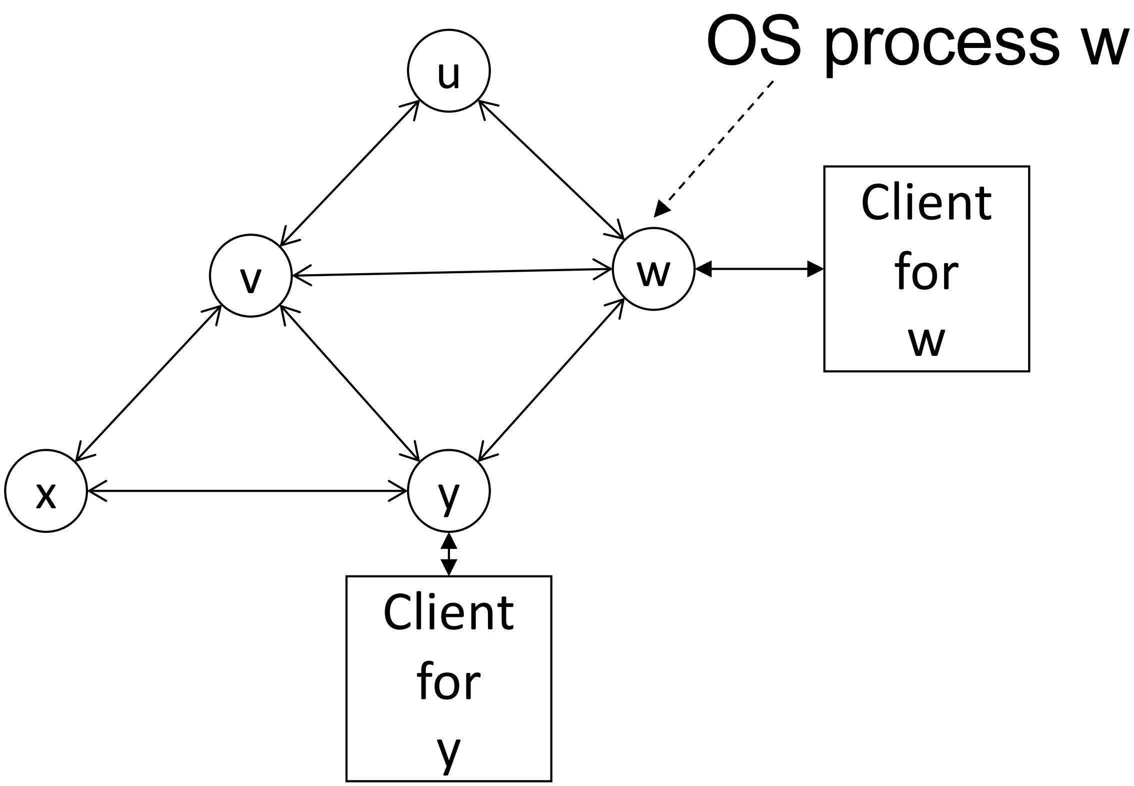
1. Conflict resolution in distributed systems: Previously: mutual exclusion. Today Distributed Dining Philosophers.
2. Dynamically changing graphs to design distributed algorithms. This lecture: Priority Graphs. Earlier lectures: Tree for Diffusing Computation. Snapshot: timeline graph.
3. Use of tokens and other invariant structures.

Reading: Sivilotti, Chapter 8

Client Life Cycle: Same as for mutex



Given: Undirected graph. Each node consists of an OS process and a client process



Specification:

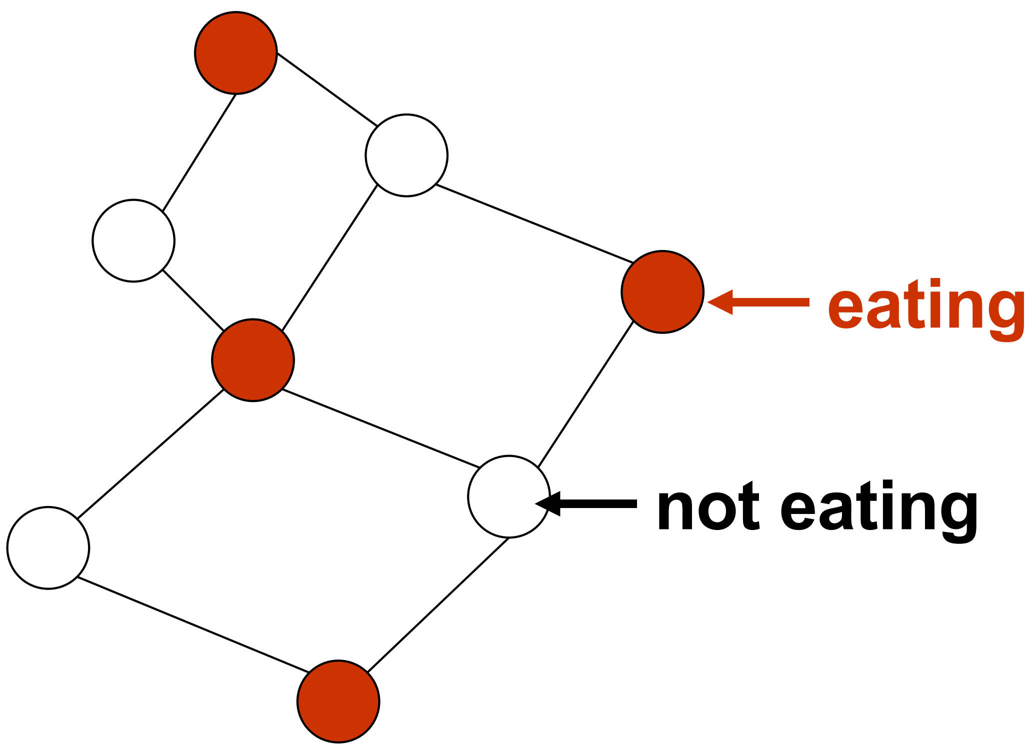
Always: Neighboring clients are not in critical sections.

Eventually: Every client waiting to enter its critical section does so.

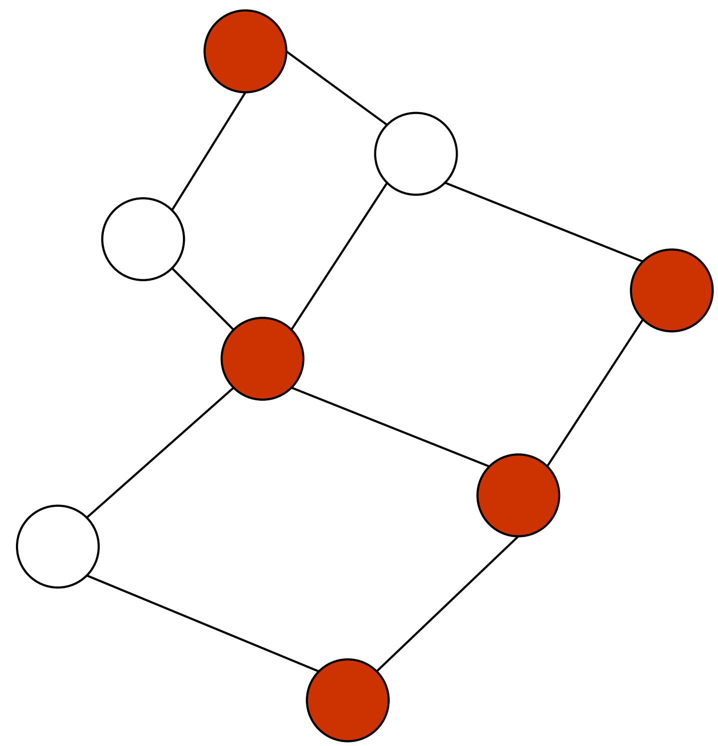
Given: Clients remain in critical sections for finite time.

Channels between neighbors.

Focus on OS: Client code is straightforward
Safety property: Always neighbors aren't eating



Safety property:
Neighbors aren't eating

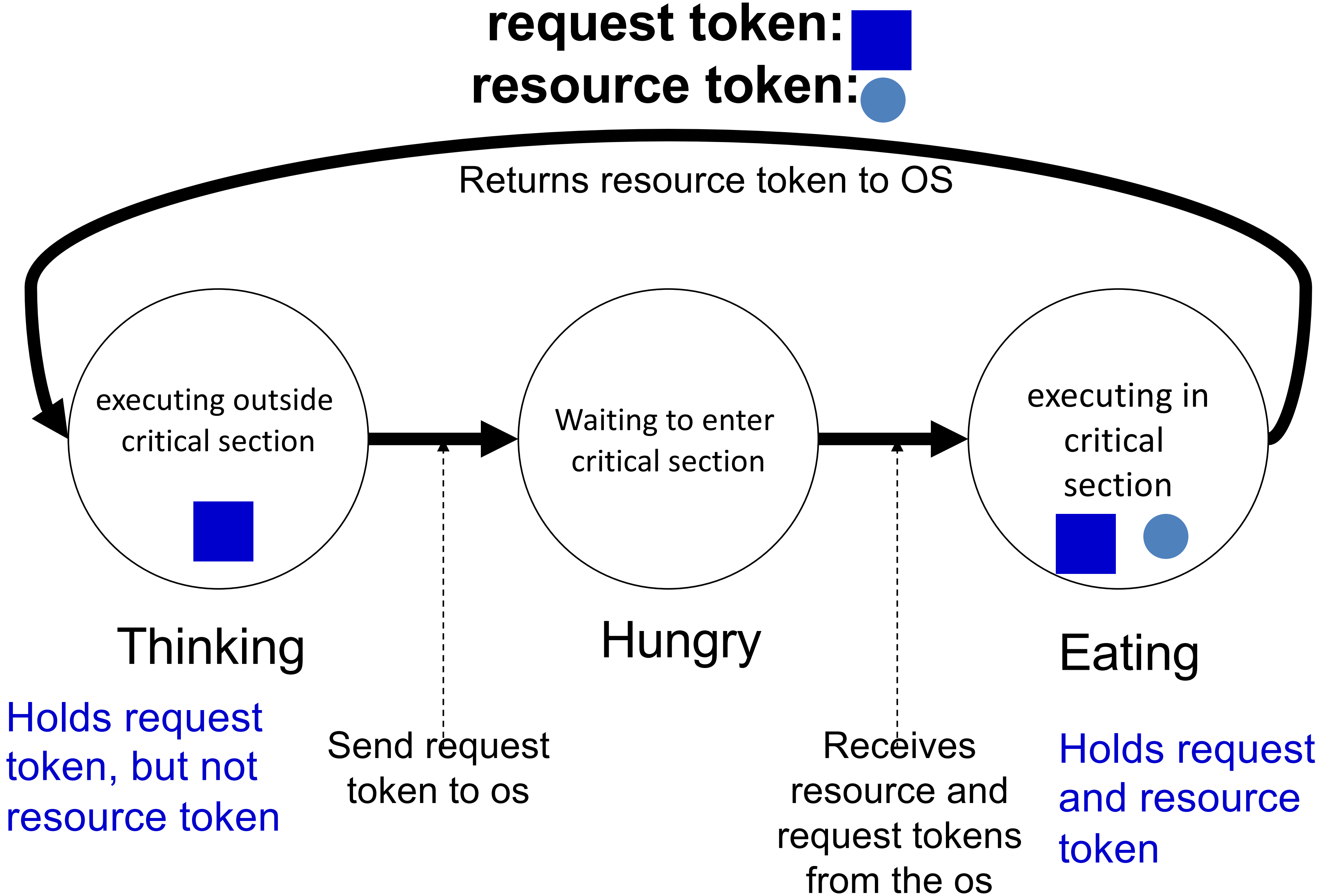


Safety property violated:
Neighbors are eating

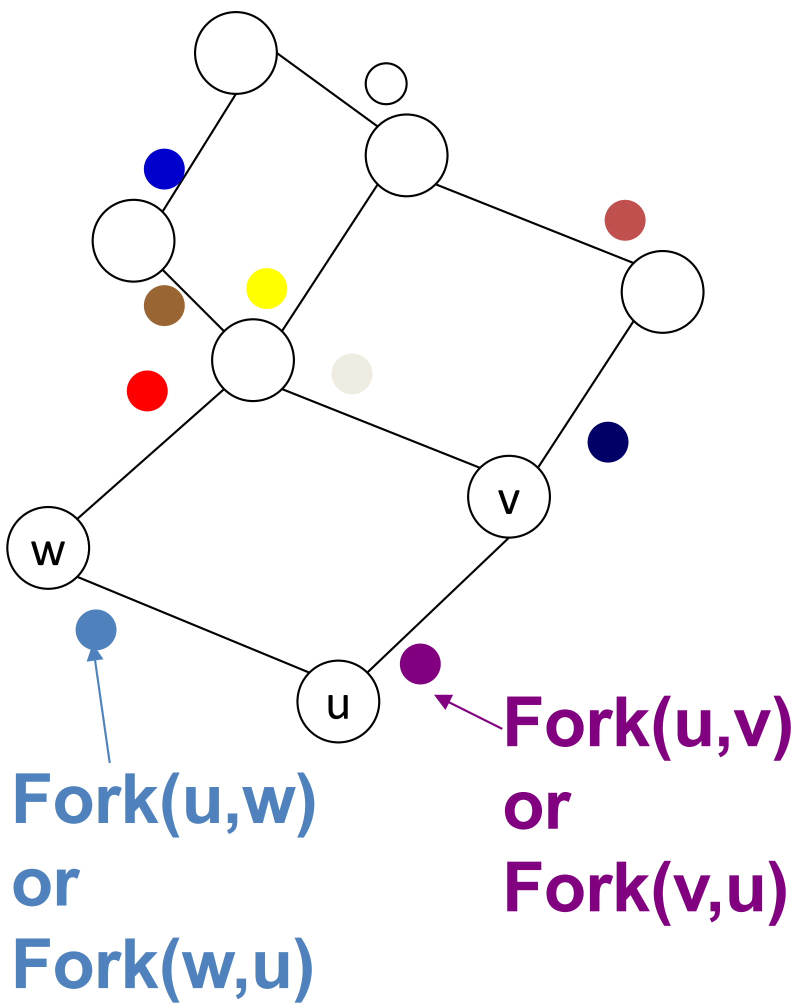
Client to OS messages:
Tokens: example of an invariant structure

request token:

resource token:



Another example of tokens: Introduction of forks

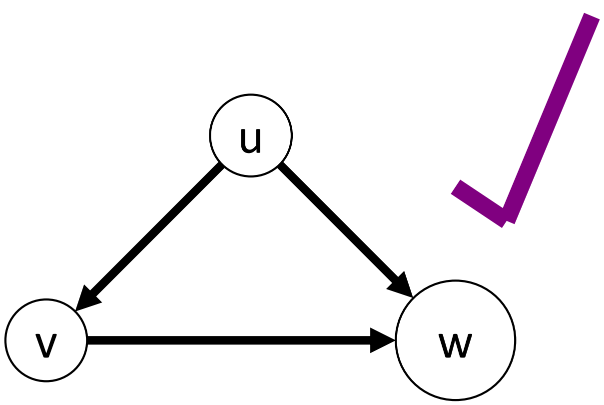


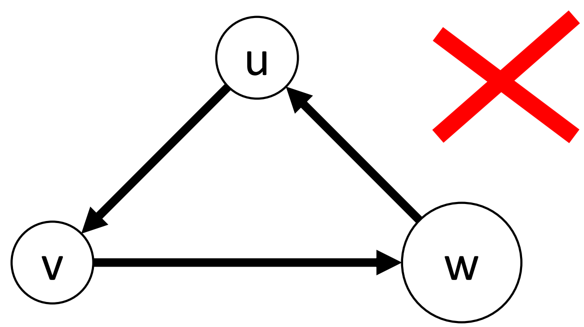
- There is exactly one fork on each edge.
- Forks on different edges have different colors: Color (u,v) is different from color (u,w).
- A fork on an edge (u, v) is at u or at v or in the channel from u to v or in the channel from v to u.
- Philosopher eats only if it holds all its forks.
- Safety property satisfied

Conflict resolution What to do when u and v want fork(u,v) at the same time?

Priority: Give the fork to the agent with higher priority.

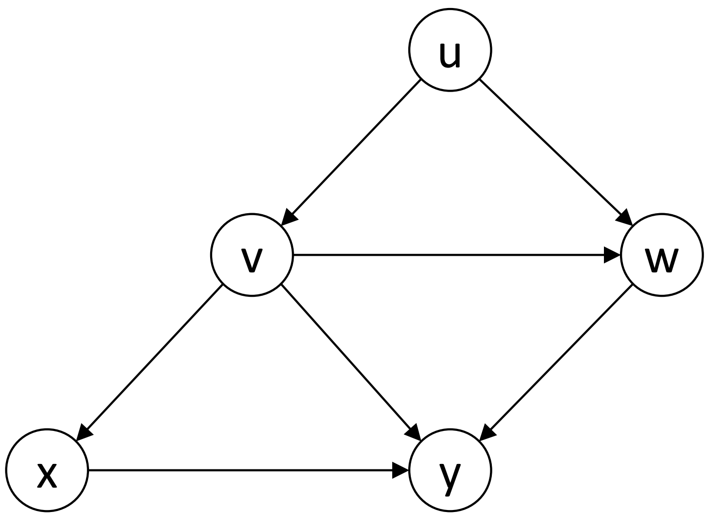
- The vertices of a priority graph represent agents.
- The directed edges represent priority. There is an edge (u, v) exactly when agent u has priority over agent v.
- Maintain the invariant that the priority graph is acyclic. Why? Because a symmetric state can persist forever.

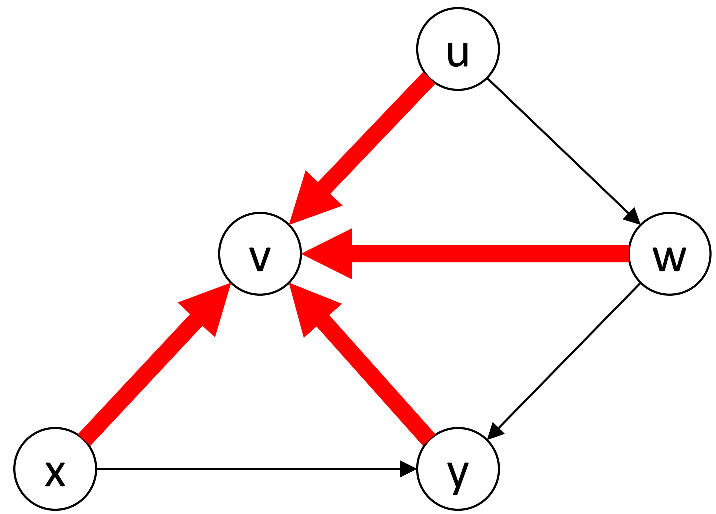




How should priorities change when a process eats?

v holds all its forks and eats

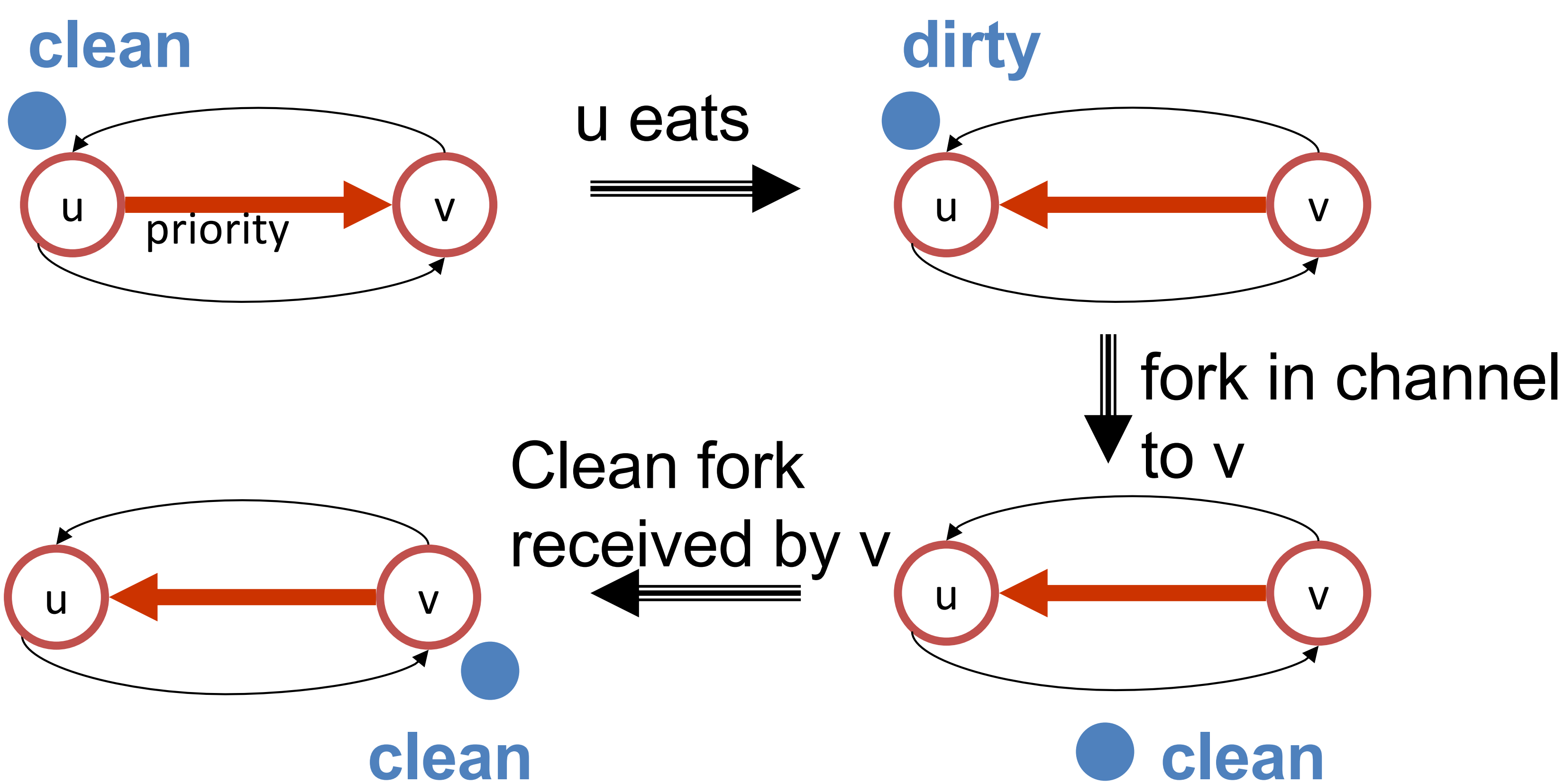




What should happen to edge directions after v eats?

- Flip edges incident on v? No. may cycle.
- Make all edges directed towards v? Yes. Prove that the graph remains acyclic.

How can we represent priorities in terms of forks?
All forks held by an eating agent are dirty.
An agent holding a dirty fork has lower priority.



Priority changes only when a clean fork becomes dirty

- **The key question**: What does a hungry philosopher u, who holds fork (u,v), do when it gets a request from a neighbor v?

- If fork (u,v) is clean then u holds the fork: it has priority; if the fork is dirty then u sends the (cleaned) fork to v.

- What does an eating philosopher u do when it gets a request for fork(u,v)?

- Yields the fork when eating completes.

- What does a thinking philosopher u, who holds fork (u,v), do when it gets a request from a neighbor v?

- Thinking philosophers must yield forks because thinking philosophers may think forever, and must not hold forks forever.

- So, we must establish the following property: **Always: thinking philosophers do not hold clean forks. (They may hold dirty forks.)**

Next class: Write program and prove correctness

1. Conflict resolution in distributed systems: Previously: mutual exclusion. Today Distributed Dining Philosophers.
2. Dynamically changing graphs to design distributed algorithms. This lecture: Priority Graphs. Earlier lectures: Tree for Diffusing Computation. Snapshot: timeline graph.
3. Use of tokens and other invariant structures. Previous lectures also emphasized always properties.