Mutual Exclusion - Disposition

Lukas Peter Jørgensen, 201206057, DA4

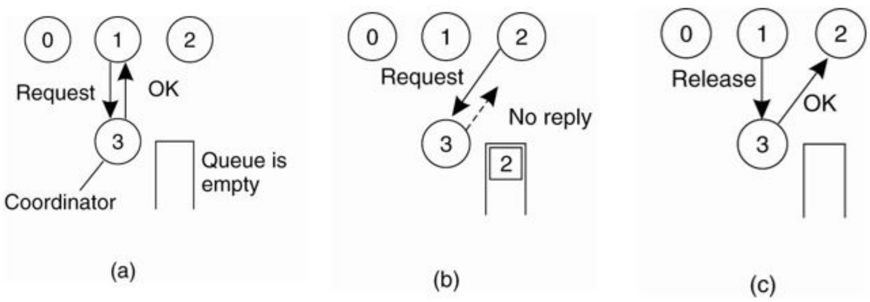
26. marts 2014

1 What is Mutex

Requirements:

1. Mutually exclusive access
2. Starvation (Every process will get a chance to access the resource)
3. Deadlock freedom
4. Fairness (Every process has a right to access the resource)

2 Centralized



Mutual Exclusion?
Trivial. **Starvation?**
Yes, trivial. **Deadlock freedom?**
Yes, trivial. **Fairness?**
Yes, since requests are granted in the order which they are recieved.

Pros

- Simple
- Easy to implement
- Low amount of messages to enter resource.

Cons

- Single point of failure.
- Coordinator becomes bottleneck.
- Depending on implementation, it can be hard to distinquish dead coordinator from permission denied.

3 Decentralized

Replicate resource n times.
When trying to access the resource, get a majority vote from $m > n/2$ coordinators.

If a coordinator crashes, it's assumed that it forgets its vote but will recover quickly. It can handle up to f failures where: $f < 2m - n$

Pros

- Not as prone to failures as the centralized approach.
- Still relatively simple.

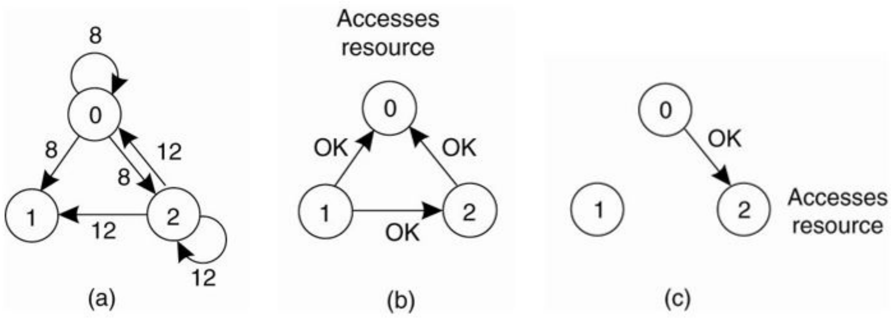
Cons

- If a lot of processes want to access the resource, utilization drops. (Starvation)

4 Distributed

Send out messages to all processes in the system, with a timestamp.
3 cases to handle upon recieving such message:

1. Not accessing resource, and no interest: Send back "OK".
2. Already has access, queue request and delay reply.
3. Has an interest in accessing resource as well, compare timestamp. If incoming has the lowest timestamp, reply "OK". Else queue the request.



Pros

- No single point of failure exists.

Cons

- Number of messages required per entry is $2(n-1)$.
- n points of failure.
- Dependent on information about the group of processes.
- All processes becomes possible bottlenecks.

5 Token ring

Just send an "access token"through a ring of processes.
Not much explanation needed.

Pros

- Fair, starvation free, deadlock free and ensures mutual exclusion.
- Efficient.

Cons

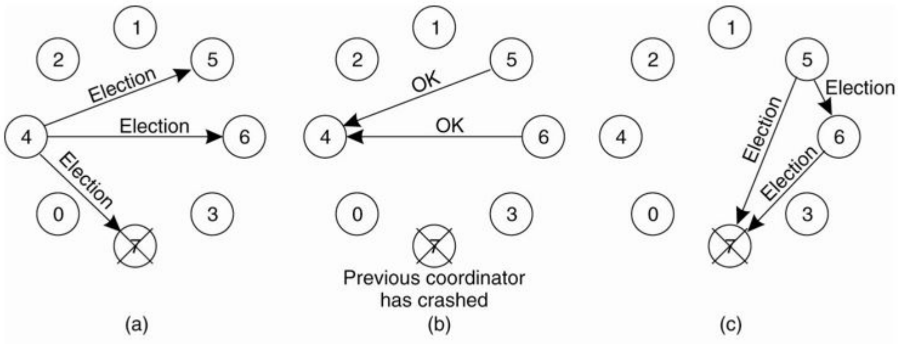
What if the token is lost?

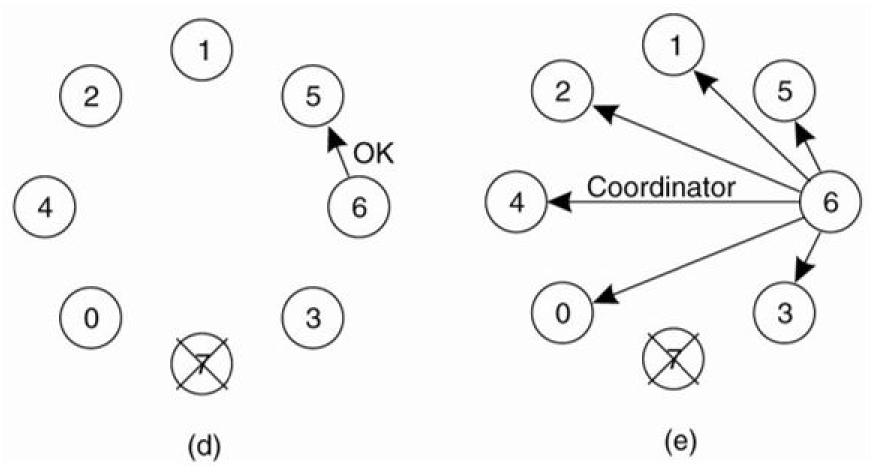
6 Election

7 Bully

1. P sends an *ELECTION* message to all processes with higher numbers.
2. If no one responds, P wins the election and becomes coordinator.

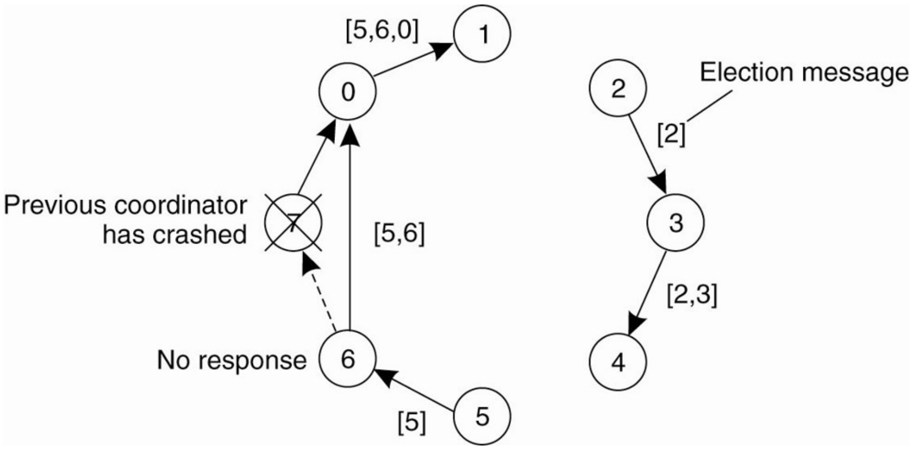
3. If one of the higher-ups answers, it takes over, P's job is done.





8 Ring

Send out an "election"message and send it to the successor, each time reciever adds it's own number to the list, when initiater recieves its own election message, it will choose the highest number in the list and send out a "coordinator"message.

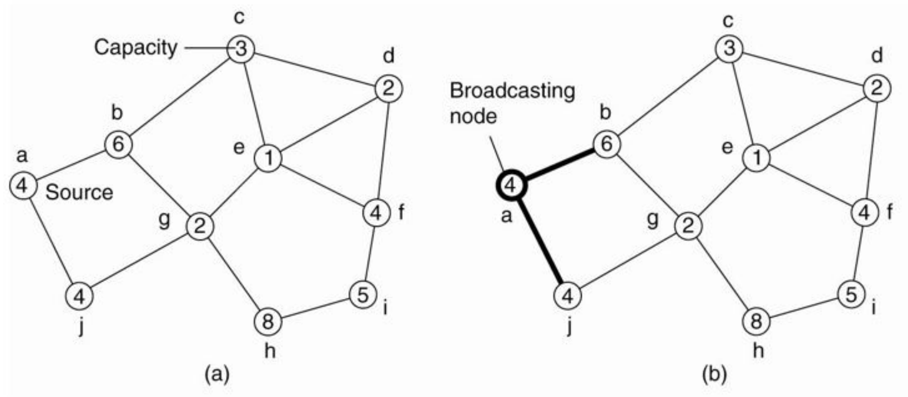


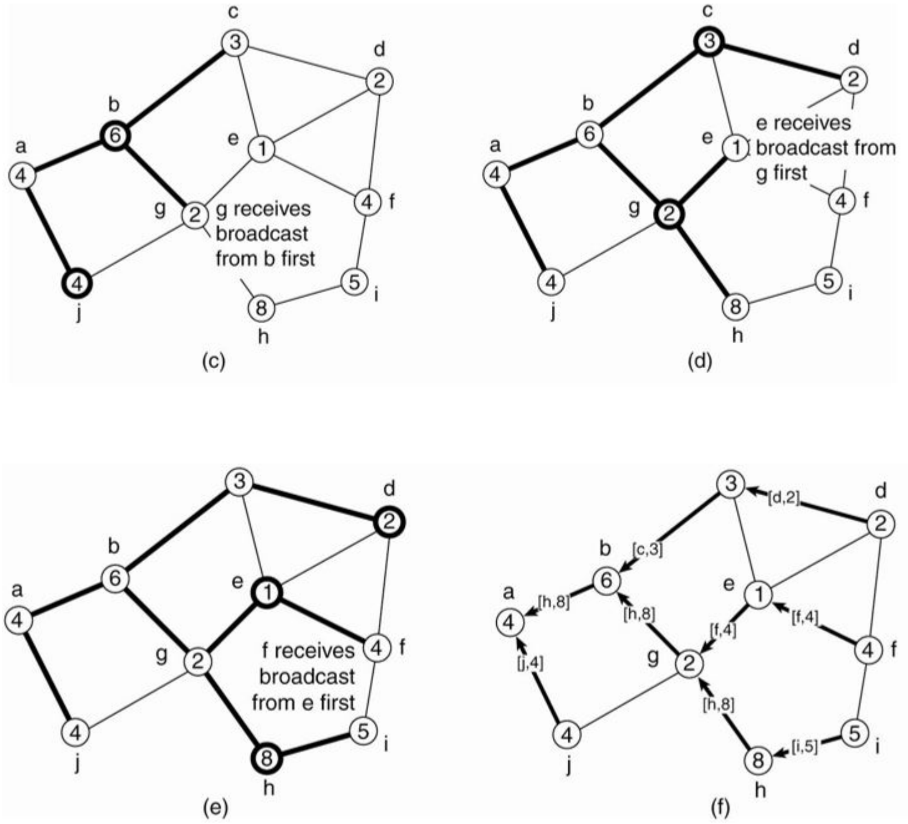
Note: Ours was a combination of ring and bully.

9 Election in wireless networks

When recieving an election message:

1. Assign sender as parent, send out election message to all neighbours except parent.
2. If already participating in the election, respond immediately.
3. When all neighbours have responded, return best candidate to parent.
4. Use unique ID to determine most valid election.





10 Large scale systems

Requirements for superpeer

1. Normal nodes should have low-latency access to superpeers.
2. Superpeers should be evenly distributed across the overlay network.
3. Predefined portion of superpeers relative to the number of nodes.
4. Each superpeer should not need to serve more than a fixed number of normal nodes.

DHT

Assume we want L leaders in m-bit Chord DHT.
Use the $k = \lceil log_2(L) \rceil$ most significant bits.
The leader for p is $lookup(p\&1^k000)$
Each super peer is then responsible for an expected $\frac{2^k}{2^m} Nnodes$.