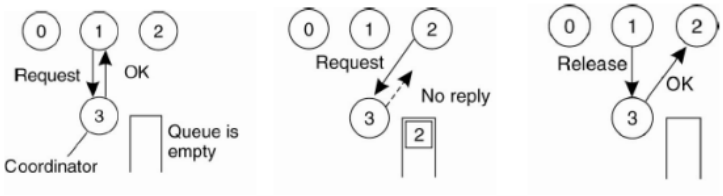
6. Synchronization

Mutual exclusion

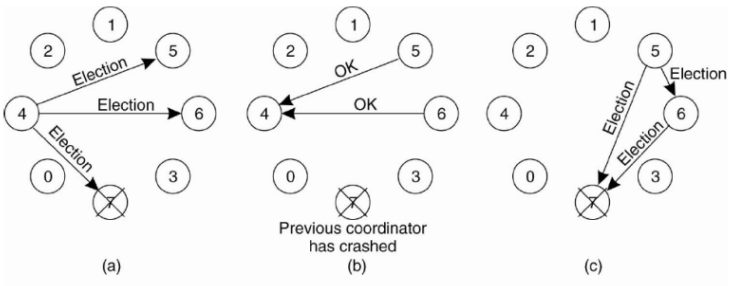
- Samme problem som ved enkelte maskiner → delte resurser. Starvation ,deadlock, fairness.



Algoritmer + sammenligning

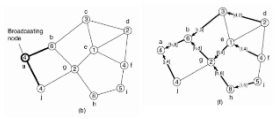
- **Centraliseret:** En central koordinator vælges, som bestemmer, hvem der har adgang.
  - One-point-of-failure (og flaskehals)
- **Decentraliseret:** Skal have tilladelse af mindst halvdelen af alle processer.
  - Traffik på netværket. Minimal risiko for, at flertallet mistes pga processer skifter mening. Risiko for starvation.
- **Distribueret:** Skal have et OK fra _alle_ processer. Sender beskeder ud til alle og afventer svar.
  - Multiple-points-of-failure (og flaskehalse)
  - Ubrugelig.
- **Token-ring:** Token sendes rundt → den der har den, må bruge den delte resurse.
  - Hvis en af processerne fejler, vil der være hul i ringen + hvis proces med token fejler, skal der en ny token ind i systemet.
  - Svær at implementere.

Election algoritmer: En koordinator skal udskiftes. Det antages at alle processer har et id, som kendes af alle andre processer.



- **Bully:** En process sender valg-besked til alle andre med større id. Dem der svarer skal gøre det samme. Først process der ikke får svar, dvs den process med størst id, bliver leader.
- **Ring:** Process sender valg-besked rundt i ringen. Alle processer skriver id på en liste. Slutteligt bliver den proces med størst id, der har skrevet sig på, valgt.
- **Wireless:** Ustruktureret ad-hoc netværk. Bedst egnede knude (batteri, forbindelse, etc) skal vælges. Start knude sender valg-besked til naboer. Rekursivt indtil alle knuder er kontaktet. Returnere egnetheds parameter. **[TRANSPARENT]**

Ordered multicasting



- Opdateringer skal sendes ud i samme rækkefølge til alle. Synkroniser beskeder vha Lamport clocks der sendes med multicast beskeder.
- JGroups: Vælger første proces i viewet til at være koordinator. Multicast beskeder sendes til den og multicastes ud til alle.