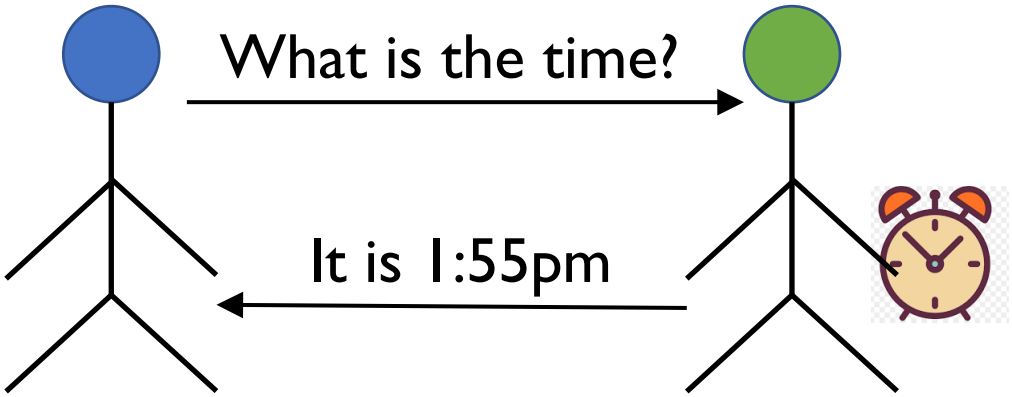
Distributed Systems

CS425/ECE428

Instructor: Radhika Mittal

While we wait…



Bluey does not own a clock, and wants to know the time.
He sends a message to Greeny asking the time, and Greeny sends a response as soon as he receives the request.
Bluey records that it took 6 minutes for him to receive Greeny's response after sending his request.

Given this information, what time should Bluey assume it actually is when he receives Greeny's message? Can he be totally accurate?

Logistics Related

- Make sure you are on CampusWire.
  - Email Emerson (sie2) to get access if you are not already on it.
- Please fill up VM cluster form by tonight.
- MP0 released today
  - Will discuss in more details at the end of the class.

Today's agenda

- Failure Detection
  - Chapter 15.1
- Time and Clocks
  - Chapter 14.1-14.3
- Logical Clocks and Timestamps (if time)
  - Chapter 14.4

Key aspects of a *distributed* system

- Processes must communicate with one another to coordinate actions. Communication time is variable.

- Different processes (on different computers) have different clocks!

- Processes and communication channels may fail.

Two ways to model

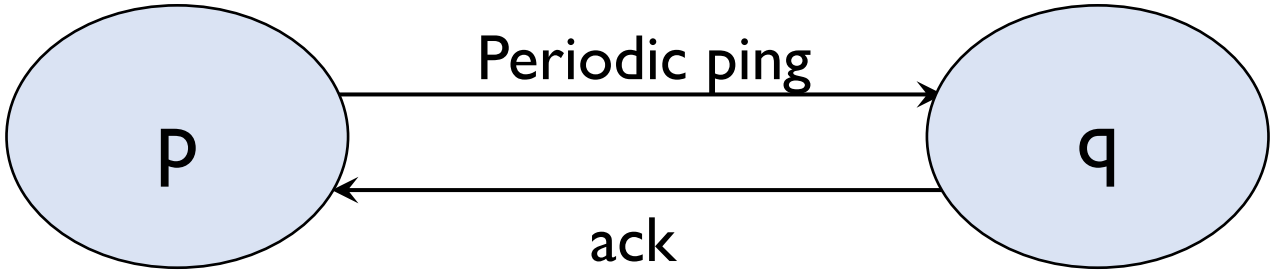
- Synchronous distributed systems:
  - Known upper and lower bounds on time taken by each step in a process.
  - Known bounds on message passing delays.
  - Known bounds on clock drift rates.

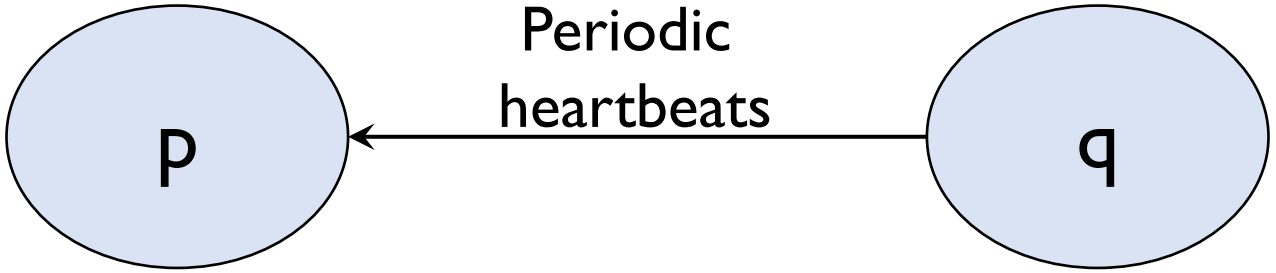
- Asynchronous distributed systems:
  - No bounds on process execution speeds.
  - No bounds on message passing delays.
  - No bounds on clock drift rates.

Types of failure

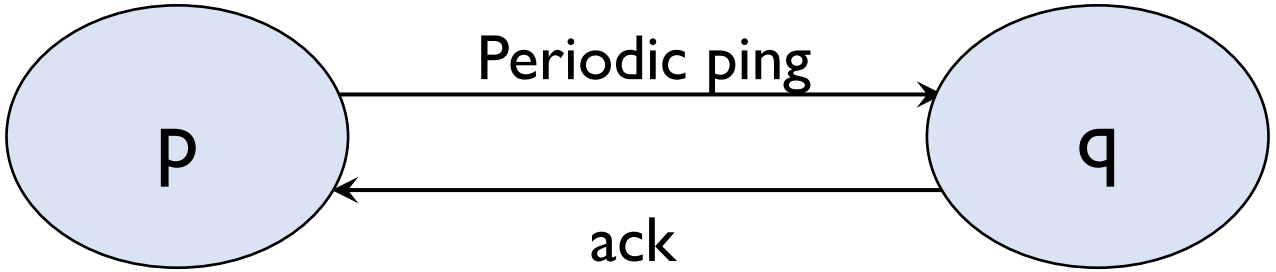
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
  - Process may **crash**.
    - Detected using ping-ack or heartbeat failure detector.
    - Completeness and accuracy in synchronous and asynchronous systems.
    - Worst case failure detection time.

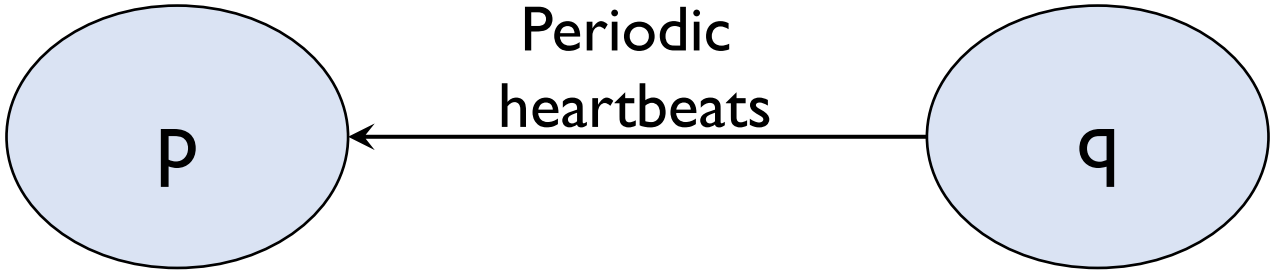
How to detect a crashed process?





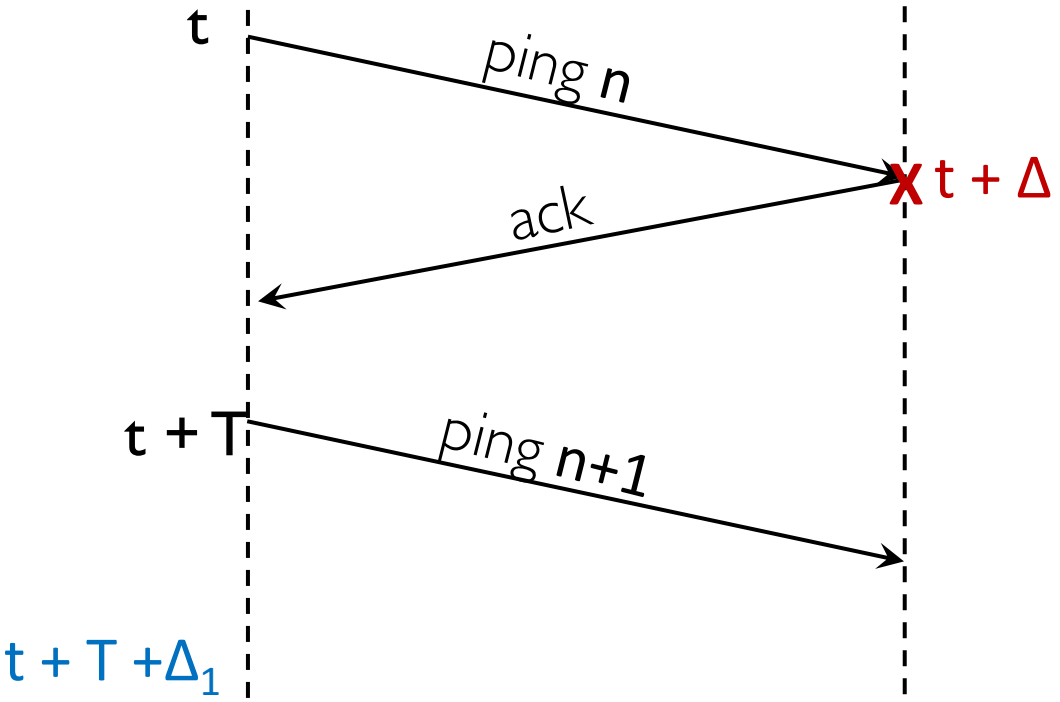
How to detect a crashed process?





Metrics for failure detection

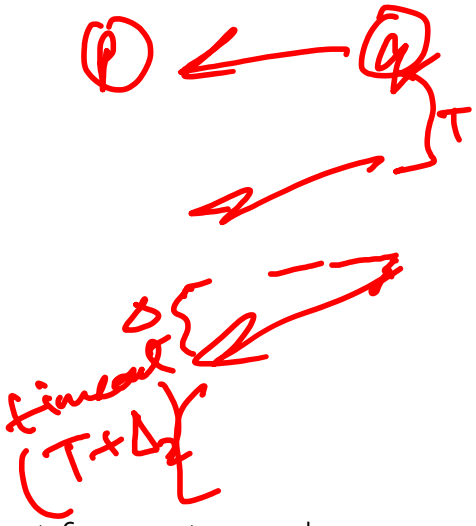
- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ where Δ is time taken for the last ping from p to reach q before q crashed. T is the time period for pings, and Δ_1 is timeout value.



Worst case failure detection time:
$t + T + \Delta_1 - (t + \Delta)$
$= T + \Delta_1 - \Delta$

Q: What is worst case value of Δ *for a synchronous system?*
A: min network delay

Metrics for failure detection



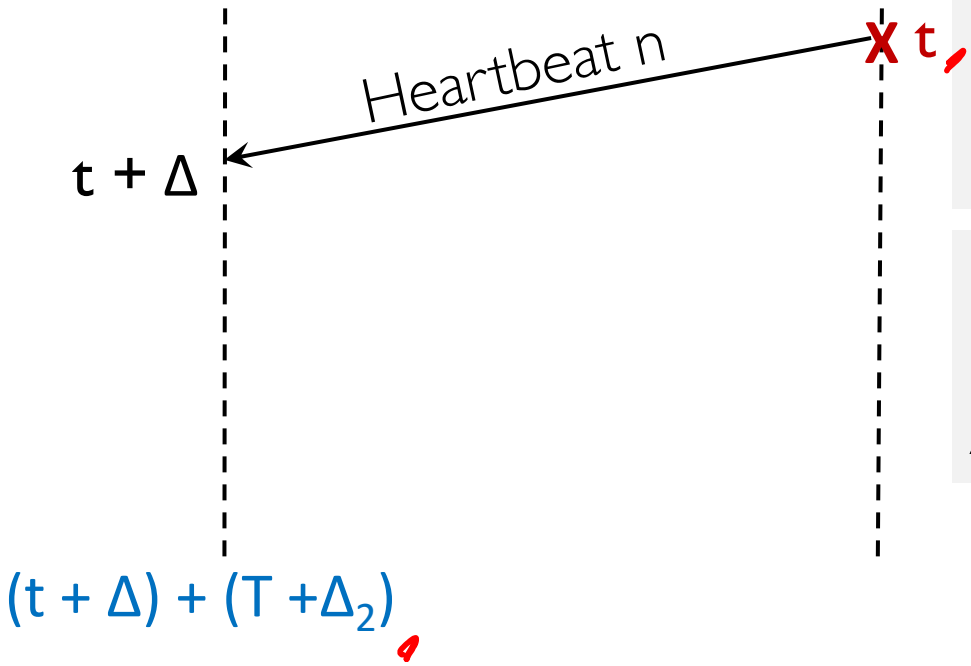
- Worst case failure detection time
  - Heartbeat: $T + \Delta_2 + \Delta$ where Δ is time taken for last heartbeat from q to reach p

    T is the time period for heartbeats, and $T + \Delta_2$ is the timeout.

Try deriving this!

Metrics for failure detection

- Worst case failure detection time
  - Heartbeat: $T + \Delta_2 + \Delta$ where Δ is time taken for last heartbeat from q to reach p
    T is the time period for heartbeats, and $T + \Delta_2$ is the timeout.



Worst case failure detection time:
$(t + \Delta) + (T + \Delta_2) - t$
$= T + \Delta_2 + \Delta$

Q: What is worst case value of Δ in a *synchronous system?*
A: max network delay

Metrics for failure detection

- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for last ping from p to reach q before crash)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)

Metrics for failure detection

- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: 1 message every T units.

Metrics for failure detection

- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: 1 message every T units.

Effect of decreasing T?

Metrics for failure detection

- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: 1 message every T units.

Decreasing T decreases failure detection time, but increases bandwidth usage.

Metrics for failure detection

- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: 1 message every T units.

Effect of increasing Δ_1 or Δ_2?

Metrics for failure detection

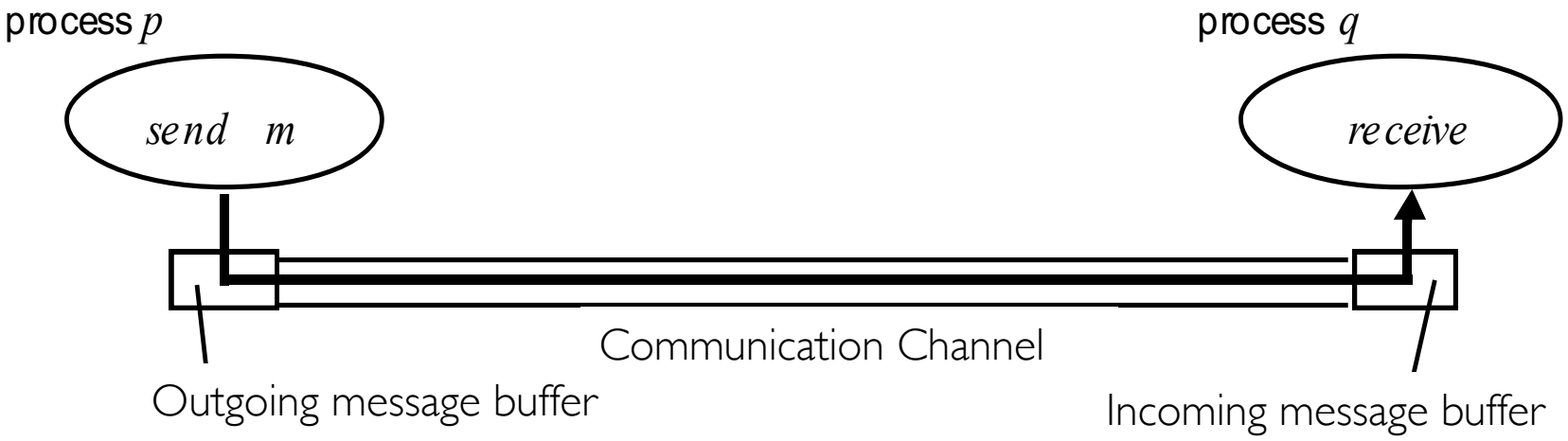
- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: 1 message every T units.

Increasing Δ_1 or Δ_2 increases accuracy (in an asynchronous system)
but also increases failure detection time.

Types of failure

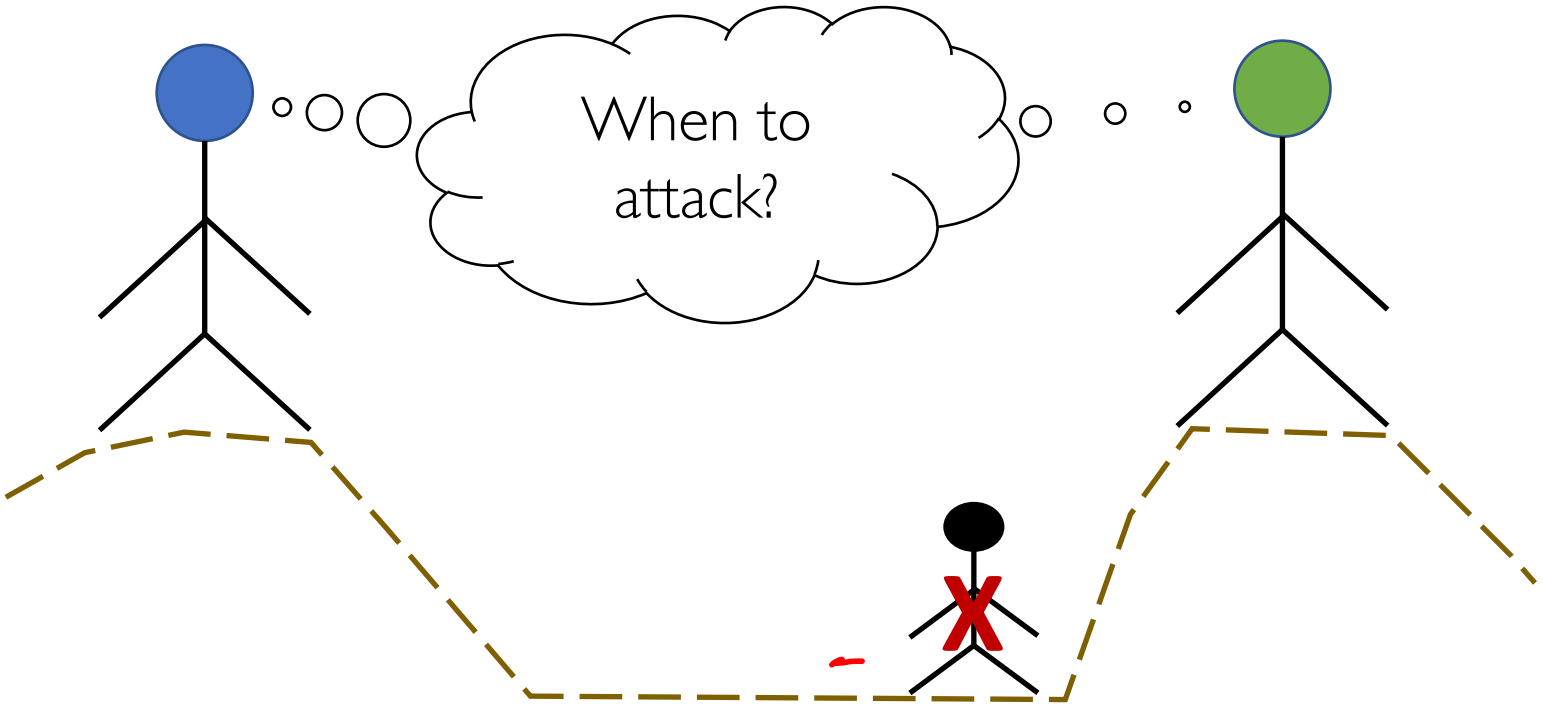
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
  - Process may **crash**.
  - **Fail-stop**: if other processes can certainly detect the crash.
  - **Communication omission**: a message sent by process was not received by another.

Communication Omission

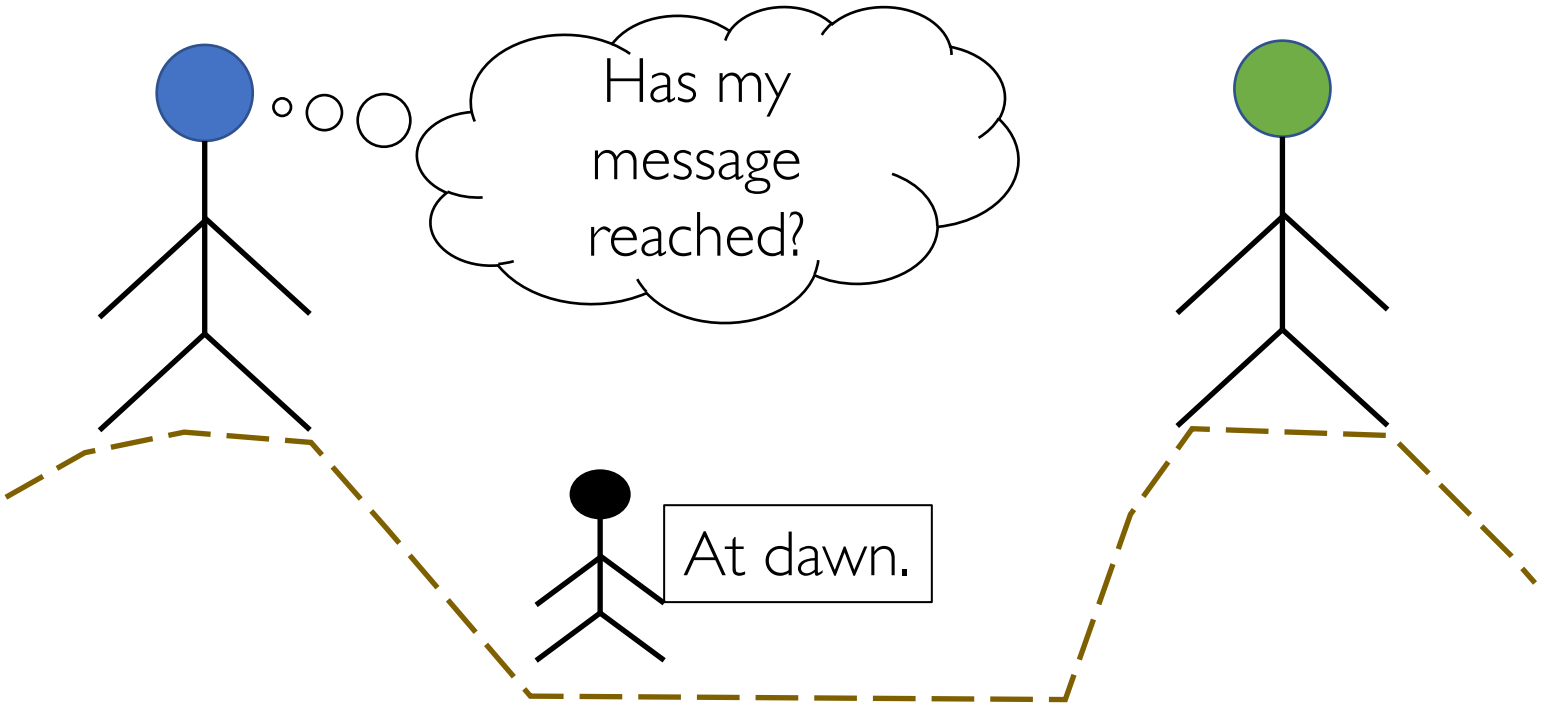


- Channel Omission: omitted by channel
- Send omission: process completes 'send' operation, but message does not reach its outgoing message buffer.
- Receive omission: message reaches the incoming message buffer, but not received by the process.

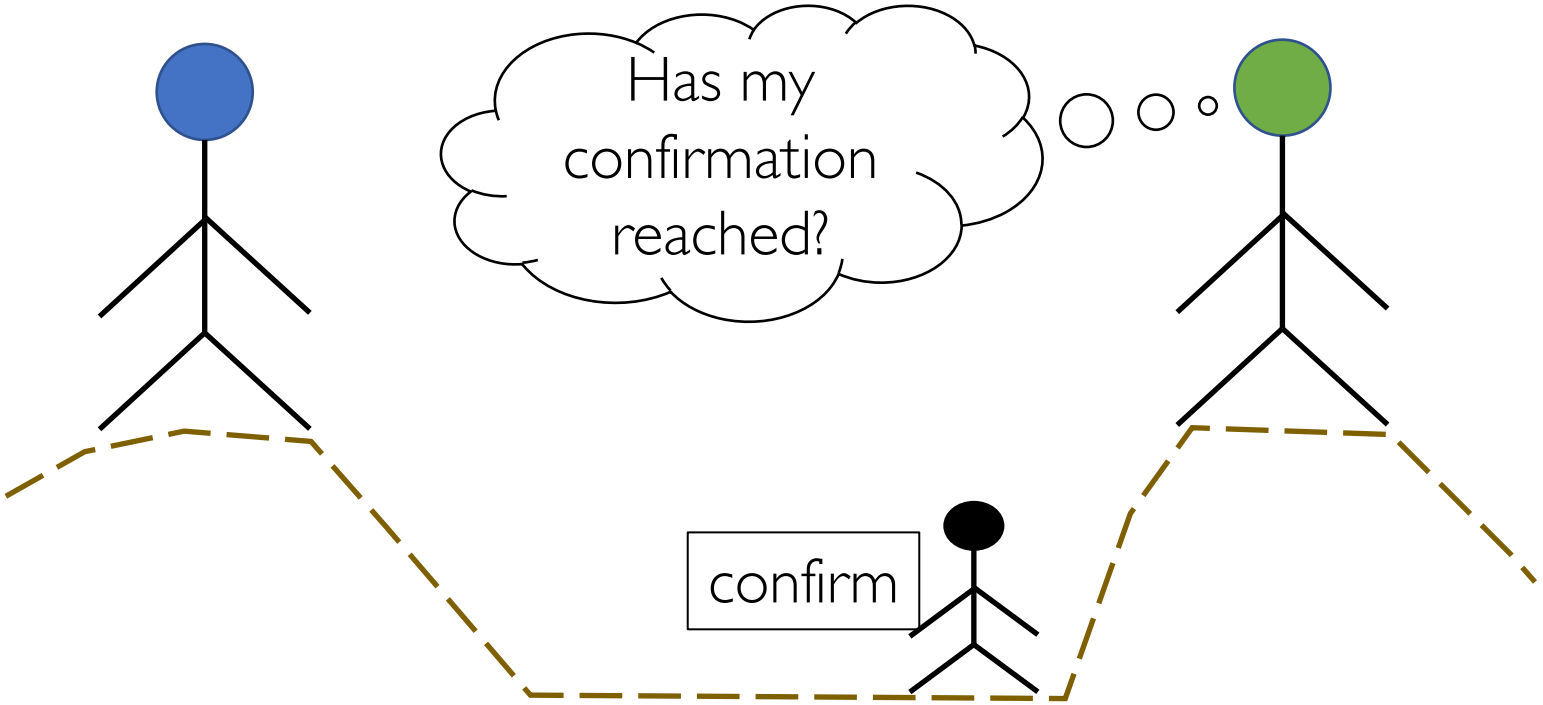
Two Generals Problem



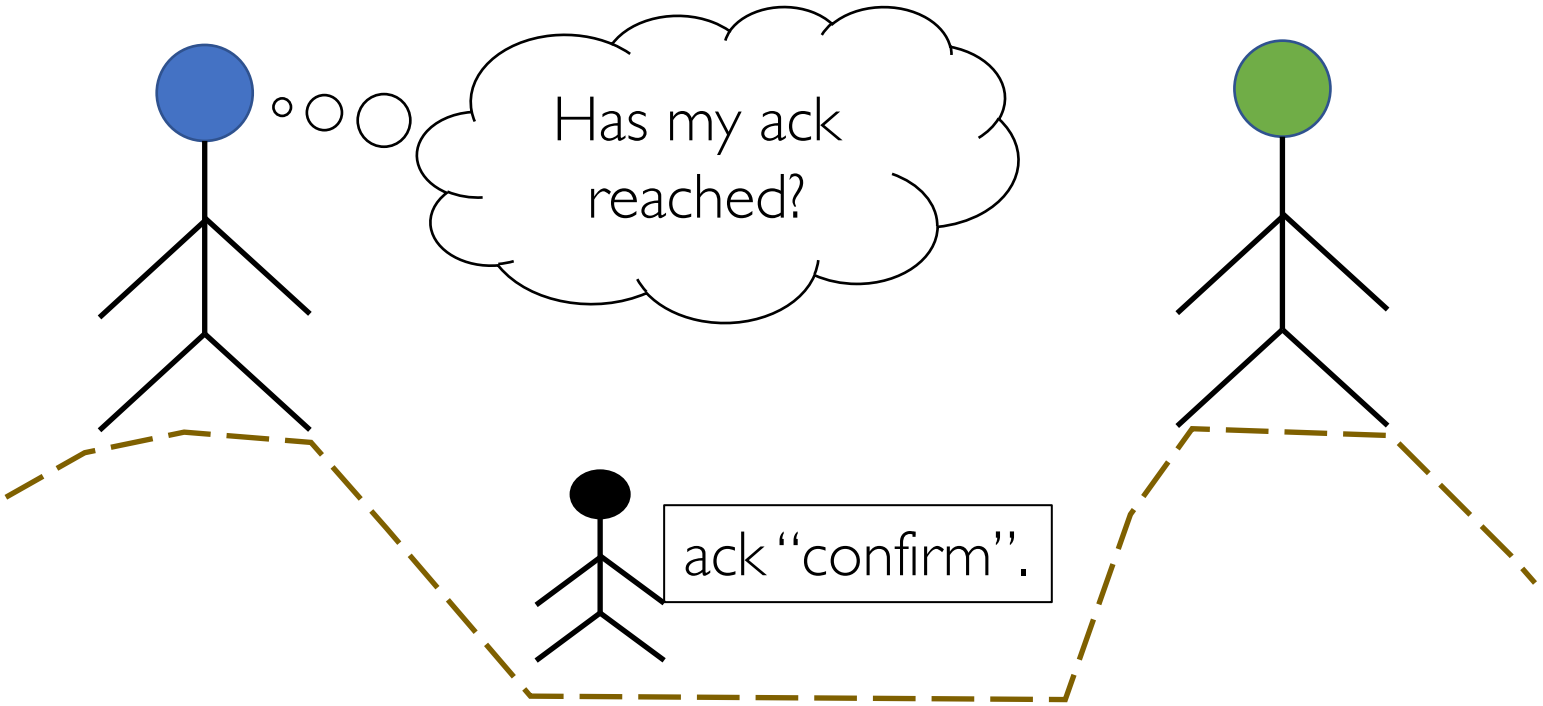
Two Generals Problem



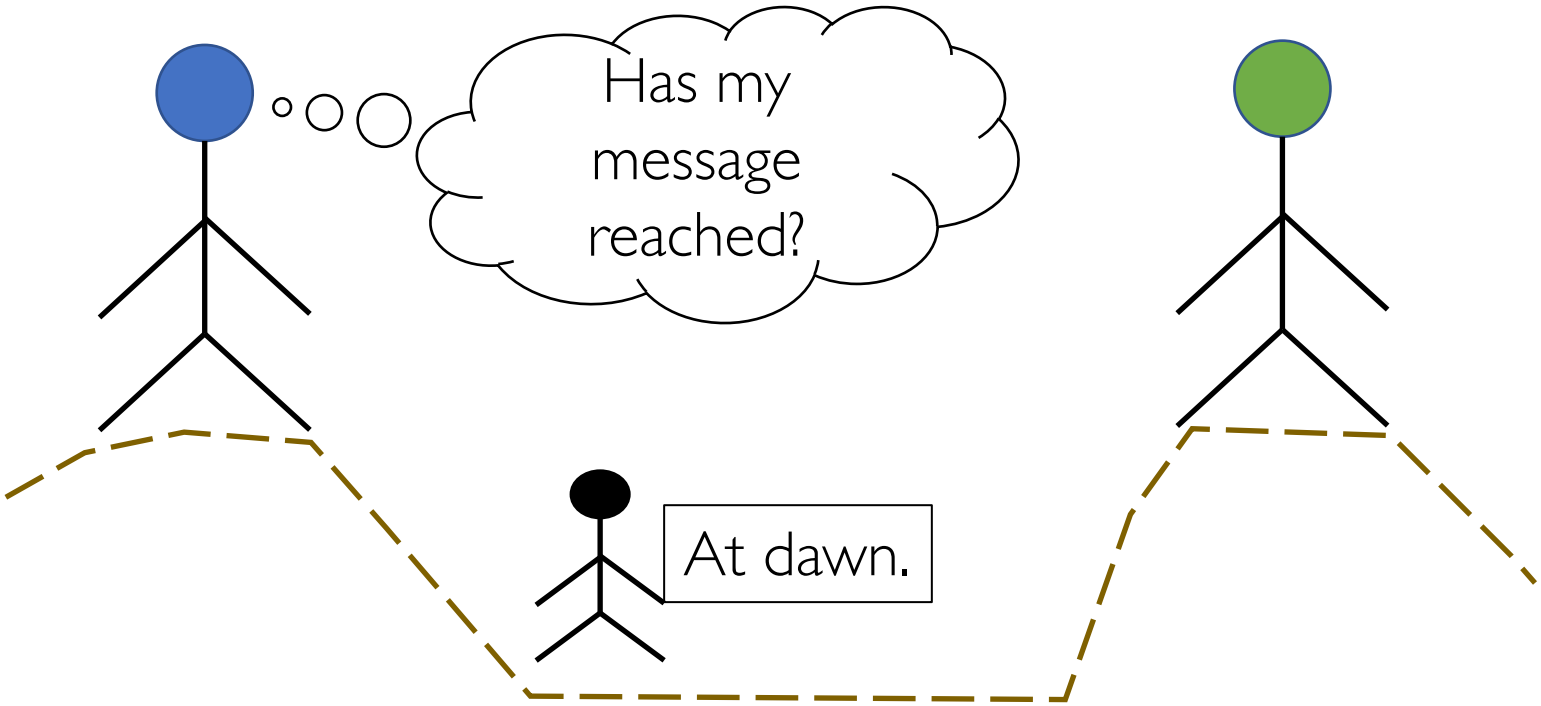
Two Generals Problem



Two Generals Problem

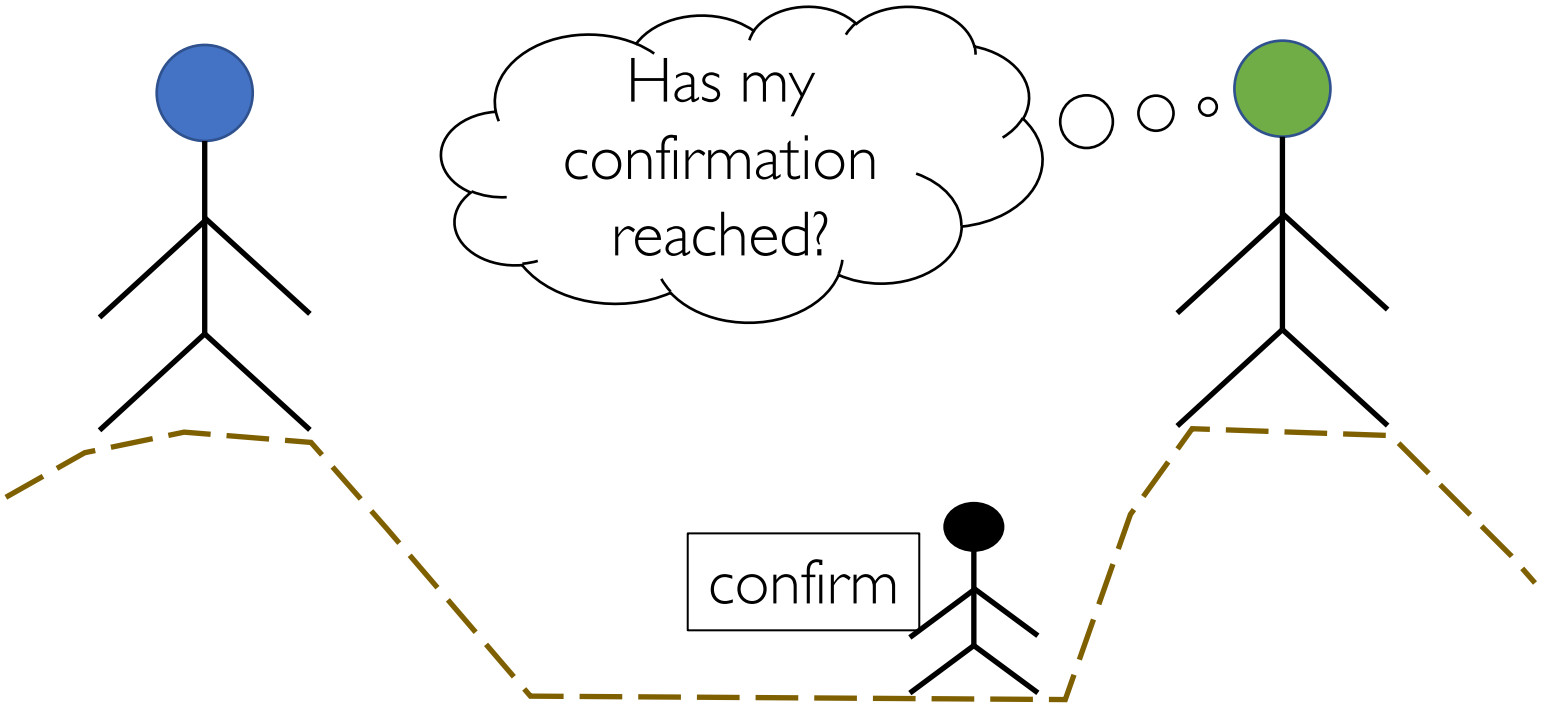


Two Generals Problem



Keep sending the message until confirmation arrives.

Two Generals Problem



Assume confirmation has reached in the absence of a repeated message.

Still no guarantees! But may be good enough in practice.

Types of failure

- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
  - Process may **crash**.
  - **Fail-stop**: if other processes can detect that the process has crashed.
  - **Communication omission**: a message sent by process was not received by another.

Message drops (or omissions) can be mitigated by network protocols.

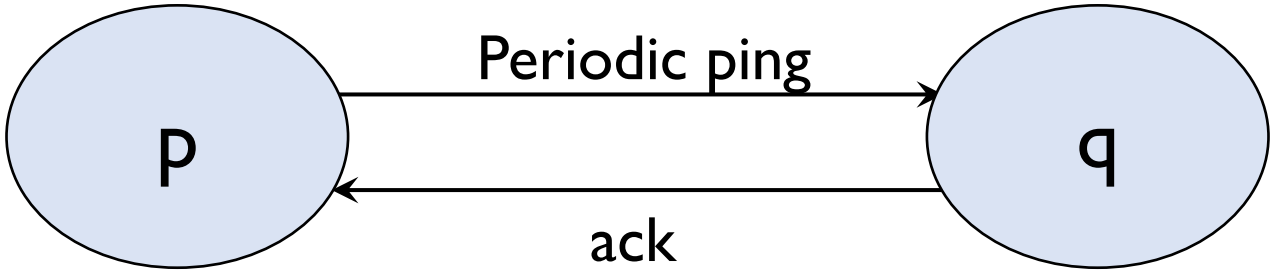
Types of failure

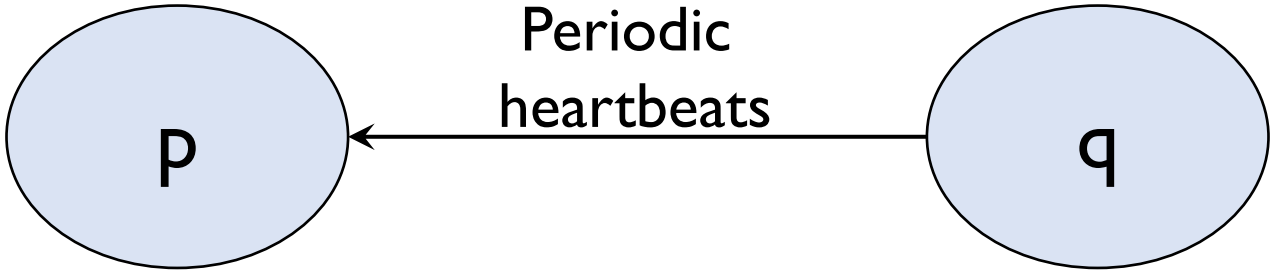
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do, e.g. process crash and message drops.

- **Arbitrary (Byzantine) Failures:** any type of error, e.g. a process executing incorrectly, sending a wrong message, etc.

- **Timing Failures:** Timing guarantees are not met.
  - Applicable only in synchronous systems.

How to detect a crashed process?





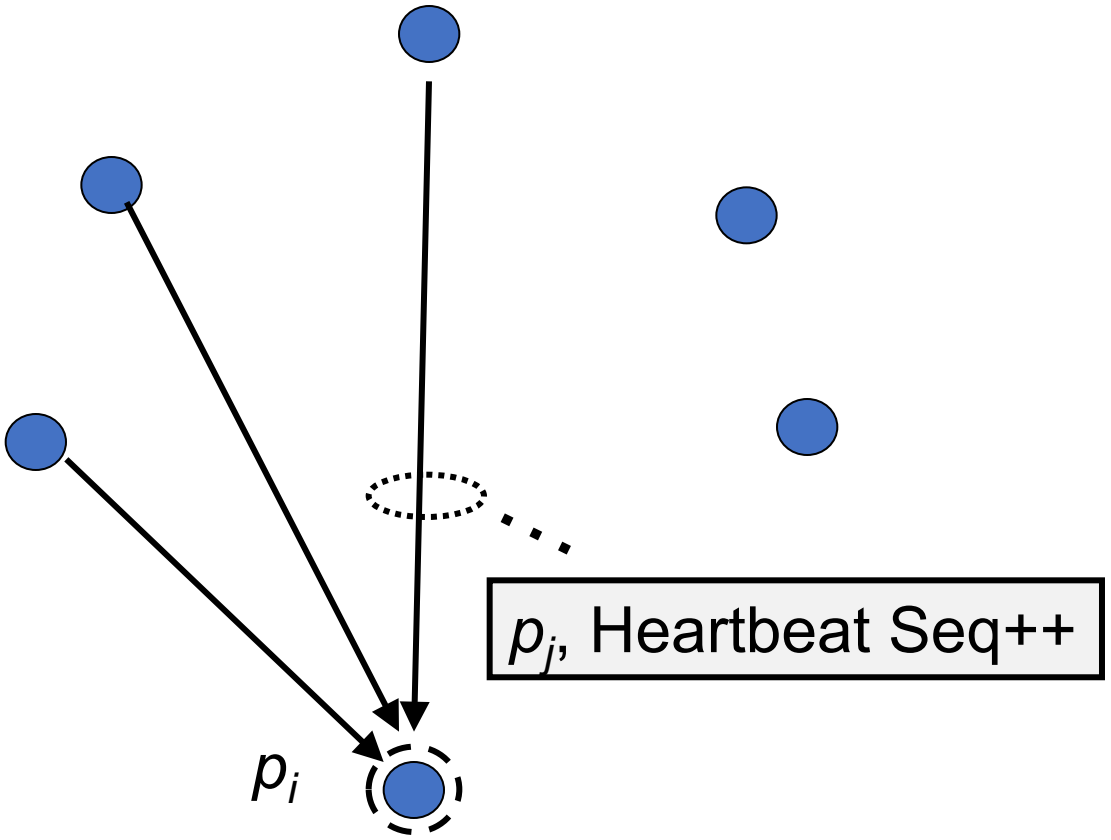
Extending heartbeats

- Looked at detecting failure between two processes.
- How do we extend to a system with multiple processes?

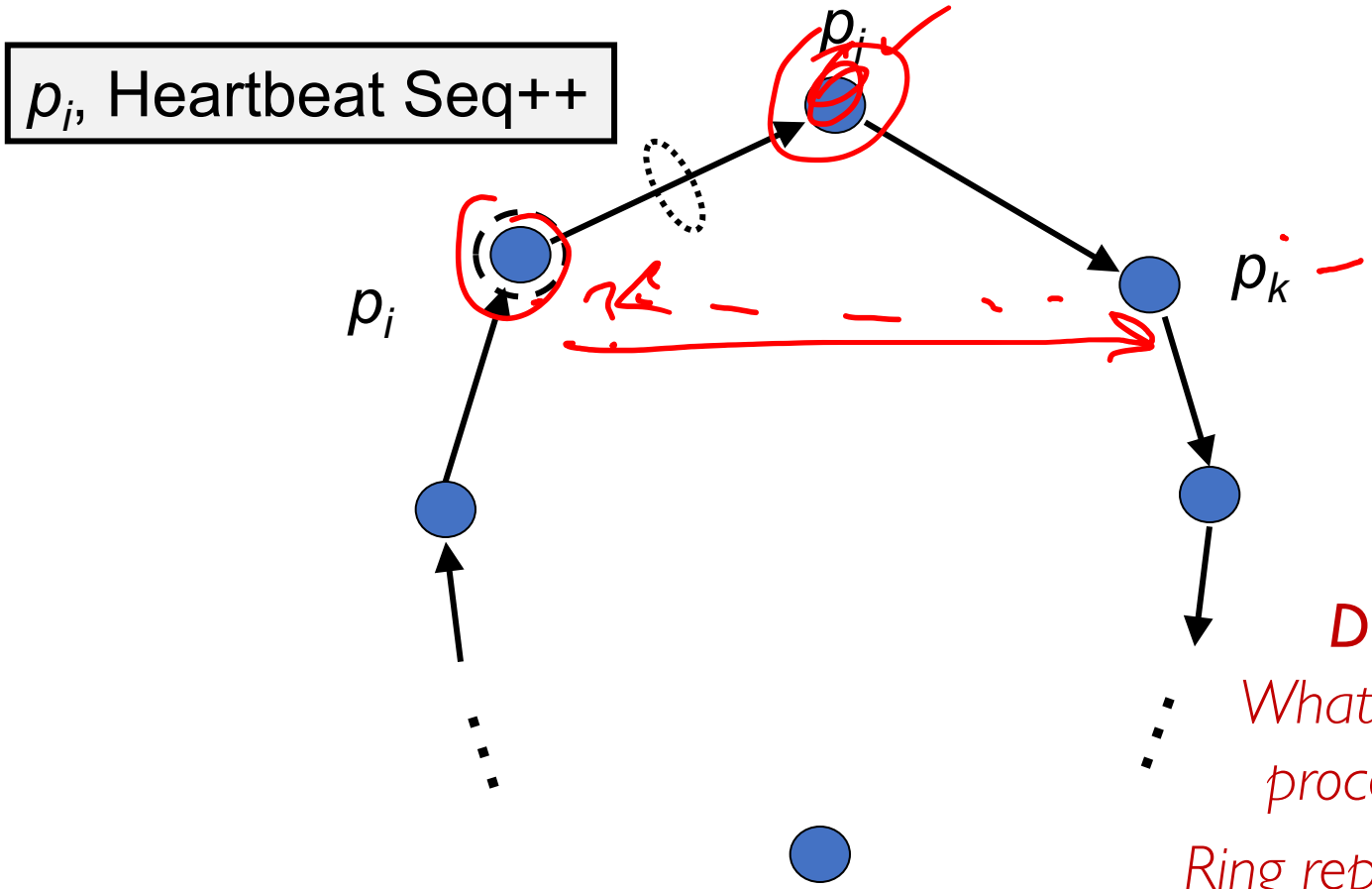
Centralized heartbeating

Downside:

What if p_i fails?



Ring heartbeating

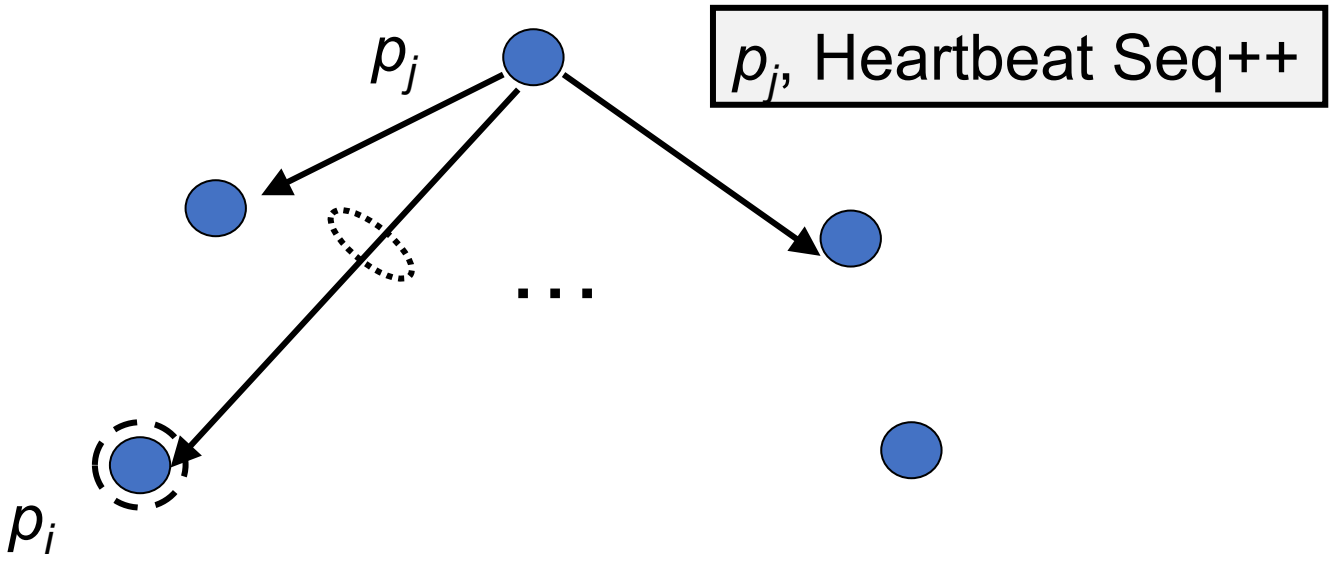


Downside:

What if multiple processes fail?

Ring repair overhead

All-to-all heartbeats



Everyone can keep track of everyone.

Downside:

Extending heartbeats

- Looked at detecting failure between two processes.
- How do we extend to a system with multiple processes?
  - Centralized heartbeating: *not complete.*
  - Ring heartbeating: *not entirely complete, ring repair overhead.*
  - All-to-all: *complete, but more bandwidth usage.*

Types of failure

- **Omission:** when a process or a channel fails to perform actions that it is supposed to do, e.g. process crash and message drops.

- **Arbitrary (Byzantine) Failures:** any type of error, e.g. a process executing incorrectly, sending a wrong message, etc.

- **Timing Failures:** Timing guarantees are not met.
  - Applicable only in synchronous systems.

Failures: Summary

- Three types
  - omission, arbitrary, timing.

- Failure detection (detecting a crashed process):
  - Send periodic ping-acks or heartbeats.
  - Report crash if no response until a timeout.
  - Timeout can be precisely computed for synchronous systems and estimated for asynchronous.
  - Metrics: *completeness, accuracy, failure detection time, bandwidth.*
  - Failure detection for a system with multiple processes:
    - Centralized, ring, all-to-all
    - Trade-off between completeness and bandwidth usage.

Today's agenda

- Failure Detection
  - Chapter 15.1
- Time and Clocks
  - Chapter 14.1-14.3
- Logical Clocks and Timestamps (if time)
  - Chapter 14.4

Why are clocks useful?

- How long did it take my search request to reach Google?
  - Requires my computer's clock to be *synchronized* with Google's server.

- Use timestamps to order events in a distributed system.
  - Requires the system clocks to be *synchronized* with one another.

- At what day and time did Alice transfer money to Bob?
  - Require *accurate* clocks (*synchronized* with a global authority).

Clock Skew and Drift Rates

- Each process has an internal clock.
- Clocks between processes on different computers differ:
  - Clock skew: relative difference between two clock values.
  - Clock drift rate: change in skew from a perfect reference clock per unit time (measured by the reference clock).
    - Depends on change in the frequency of oscillation of a crystal in the hardware clock.
- Synchronous systems have bound on maximum drift rate.

Ordinary and Authoritative Clocks

- Ordinary quartz crystal clocks:
  - Drift rate is about 10^{-6} seconds/second.
  - Drift by 1 second every 11.6 days.
  - Skew of about 30minutes after 60 years.

- High precision atomic clocks:
  - Drift rate is about 10^{-13} seconds/second.
  - Skew of about 0.18ms after 60 years.
  - Used as standard for real time.
  - Universal Coordinated Time (UTC) obtained from such clocks.

Two forms of synchronization

- External synchronization
  - Synchronize time with an authoritative clock.
  - When accurate timestamps are required.
- Internal synchronization
  - Synchronize time internally between all processes in a distributed system.
  - When internally comparable timestamps are required.
- If all clocks in a system are externally synchronized, they are also internally synchronized.

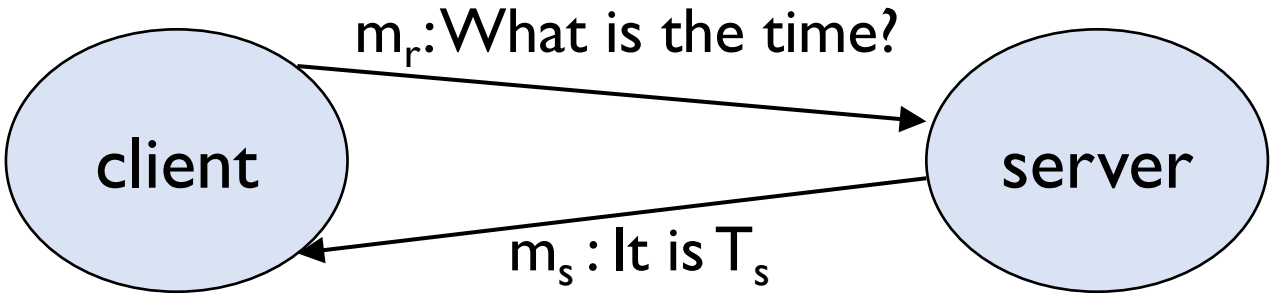
Synchronization Bound

- Synchronization bound (D) between two clocks A and B over a real time interval I.
  - $|A(t) - B(t)| < D$, for all t in the real time interval I.
    - Skew(A, B) < D during the time interval I.
    - A and B *agree* within a bound *D*.
  - If A is authoritative, D can also be called *accuracy bound*.
    - B is *accurate* within a bound of *D*.
- Synchronization/accuracy bound (D) at time 't'
  - worst-case skew between two clocks at time 't'
    - Skew(A, B) < D at time t

Q: *If all clocks in a system are externally synchronized within a bound of D, what is the bound on their skew relative to one another?*

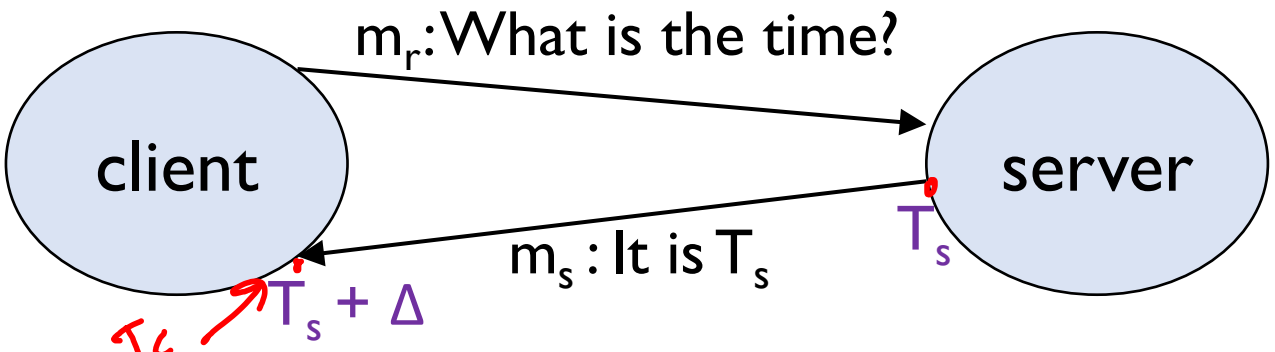
A: 2D. So the clocks are internally synchronized within a bound of 2D.

Synchronization in synchronous systems



What time T_c should client adjust its local clock to after receiving m_s ?

Synchronization in synchronous systems



What time T_c should client adjust its local clock to after receiving m_s ?

Let *max* and *min* be maximum and minimum network delay.

To be continued in next class….