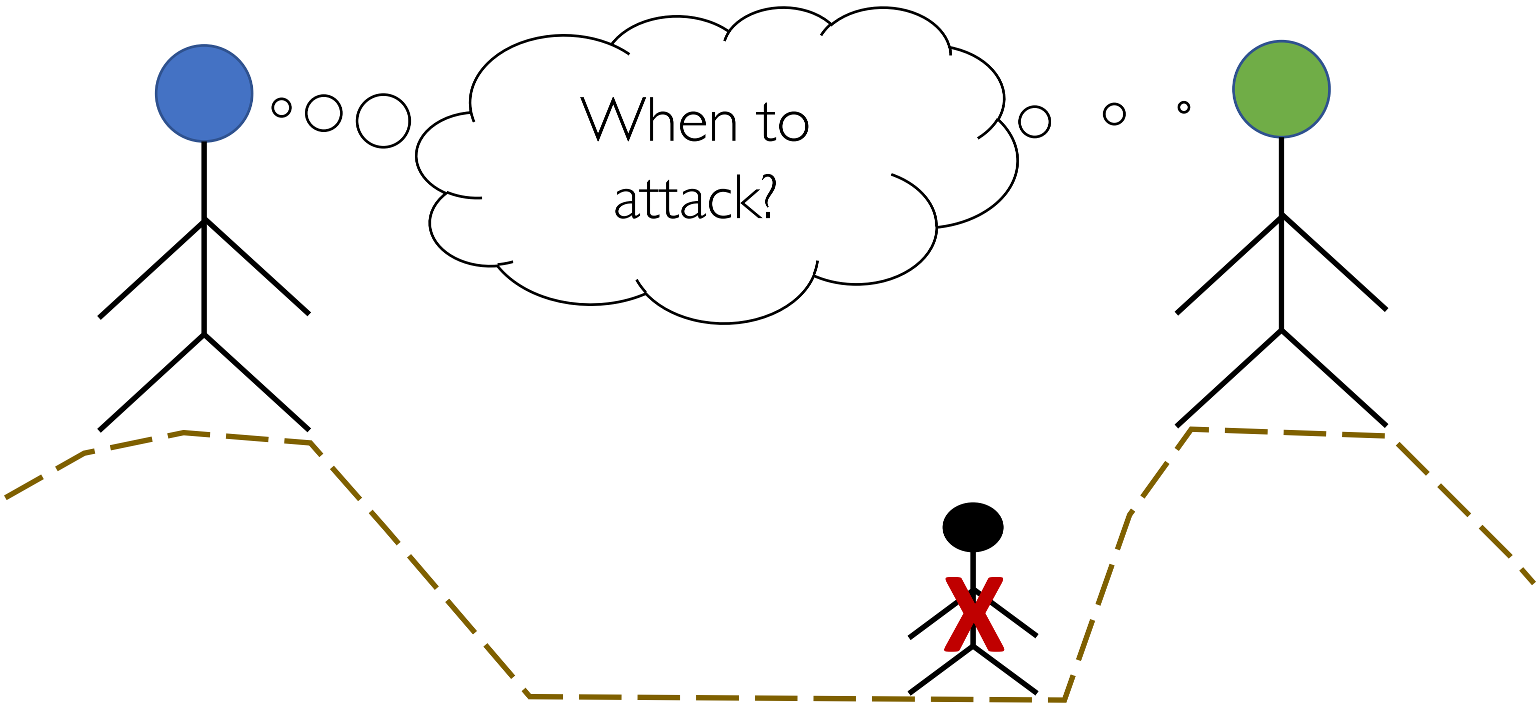
Distributed Systems

CS425/ECE428

Instructor: Radhika Mittal

While we wait…..



Two generals must agree on a time to attack the enemy base.
They can communicate with each-other by sending messengers.
But, a messenger may get killed by the enemy along the way.
Thankfully, they have unlimited no. of messengers at their disposals.
How can the two generals agree on a time to attack?

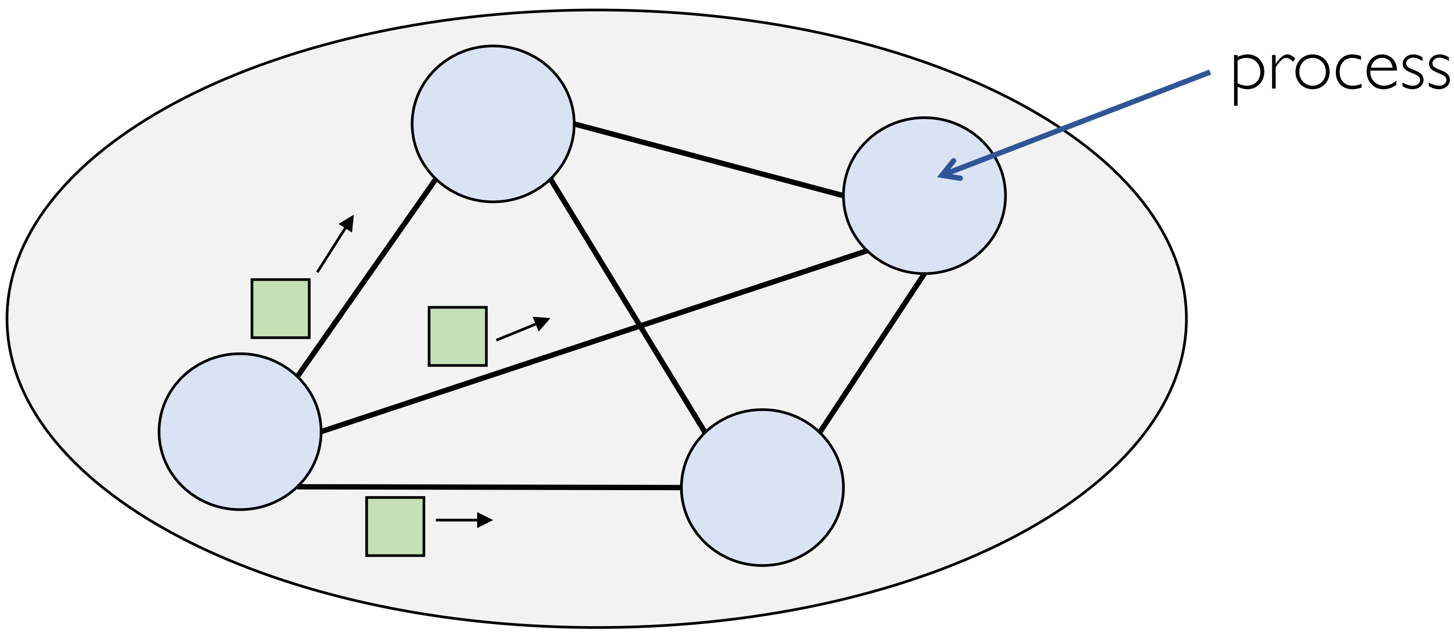
Logistics Related

- OHs information is up on website/course calendar
- Sign-up forms for VM clusters is available on CampusWire.
  - Please fill it up by Thursday, Jan 25th, Thursday, 11:59pm.
- MP0 will be released on Wednesday.
- Lecture recordings on MediaSpace should be accessible to all registered students.

Today's agenda

- System Model
  - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3
- Failure Detection
  - Chapter 15.1

What is a distributed system?



Independent components that are **connected by a network** and communicate by **passing messages** to achieve a common goal, appearing as **a single coherent system**.

Relationship between processes

- Two main categories:
  - Client-server
  - Peer-to-peer

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.

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.

Synchronous and Asynchronous

- Most real-world systems are asynchronous.
  - Bounds can be estimated, but hard to guarantee.
  - Assuming system is synchronous can still be useful.

- Possible to build a synchronous system.

Key aspects of a *distributed* system

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

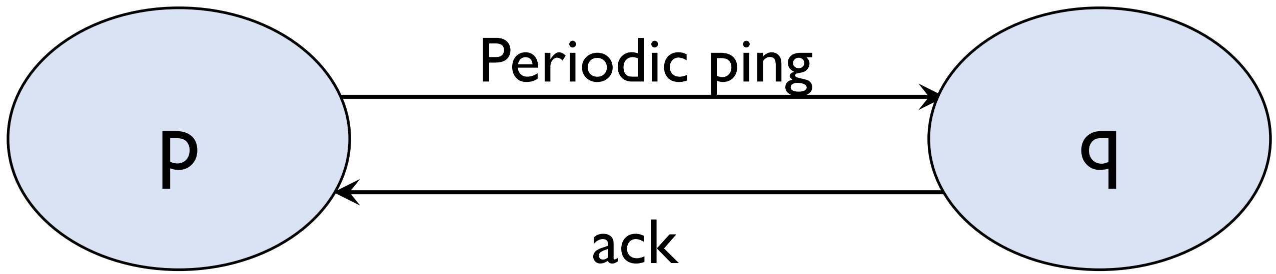
- Different processes (on different computers) have different clocks!

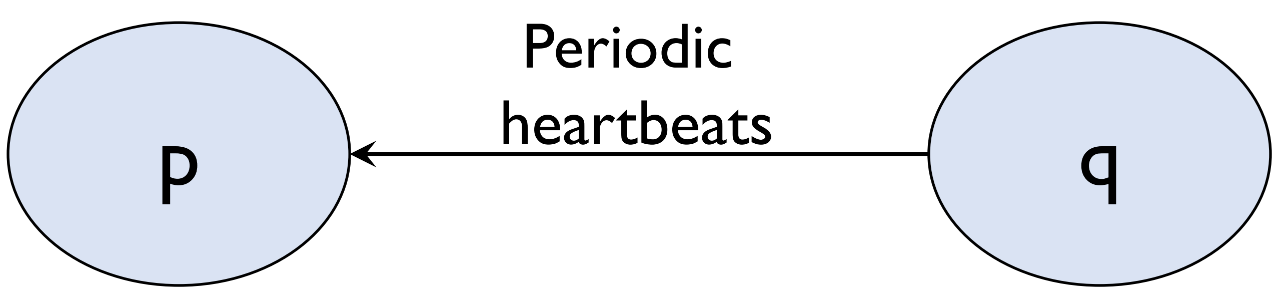
- Processes and communication channels may fail.

Types of failure

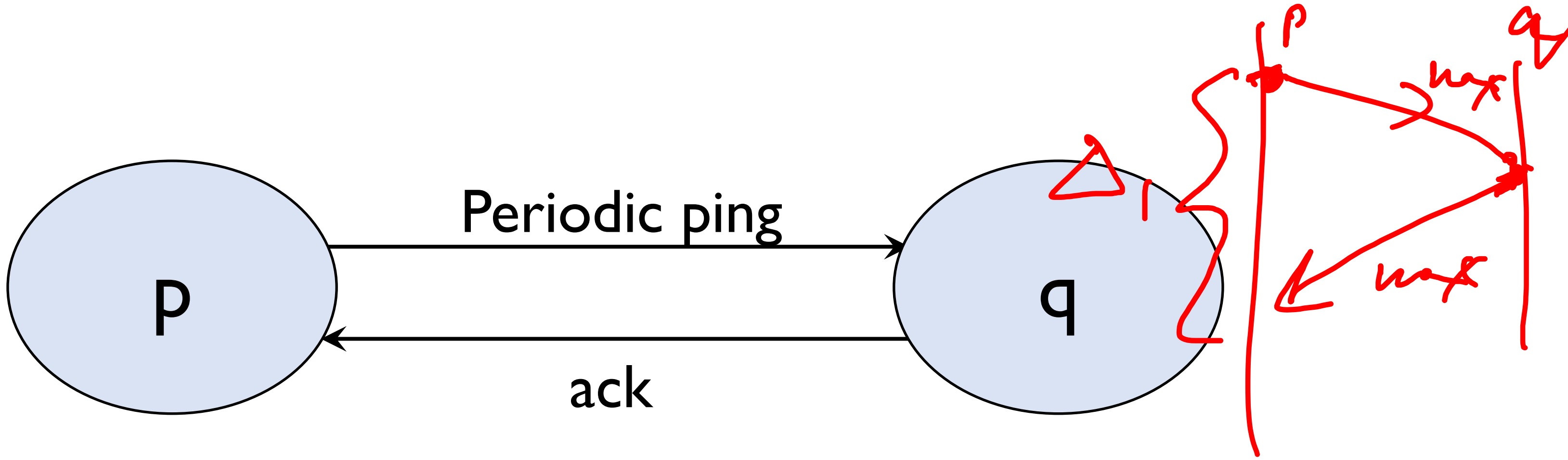
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
    - Process may **crash**.

How to detect a crashed process?





How to detect a crashed process?



p sends pings to q every T seconds.

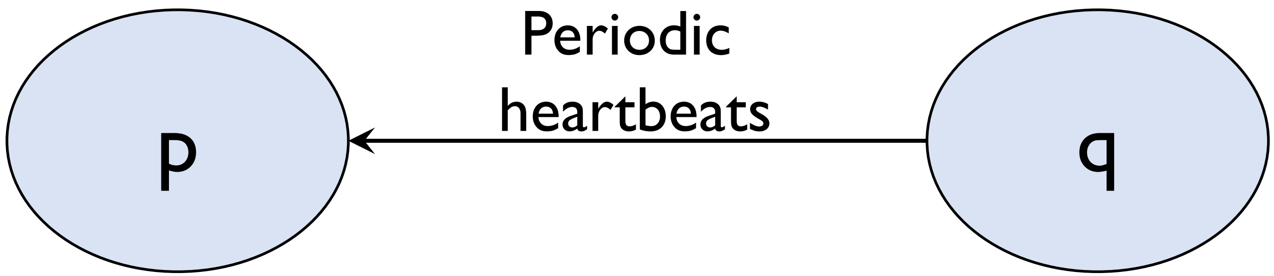
Δ_1 is the *timeout* value at p.

If Δ_1 time elapsed after sending ping, and no ack, report q crashed.

If synchronous, Δ_1 = 2(max network delay)

If asynchronous, Δ_1 = k(max observed round trip time)

How to detect a crashed process?



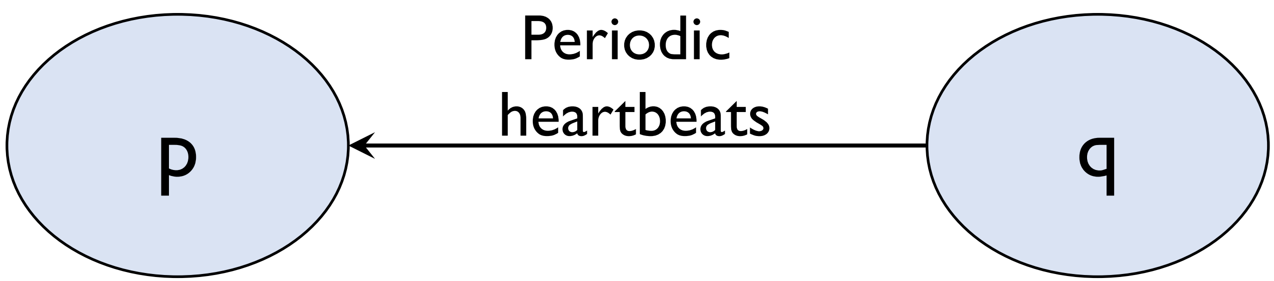
q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ is the *timeout* value at p.

If $(T + \Delta_2)$ time elapsed since last heartbeat, report q crashed.

If synchronous, $\Delta_2 =$ max network delay – min network delay

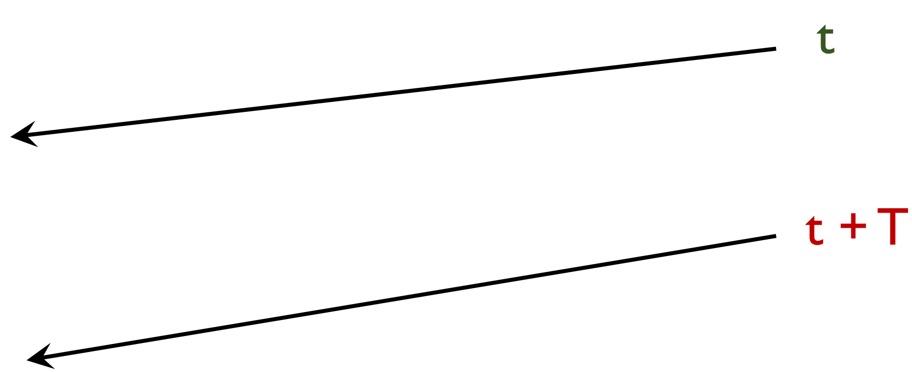
How to detect a crashed process?



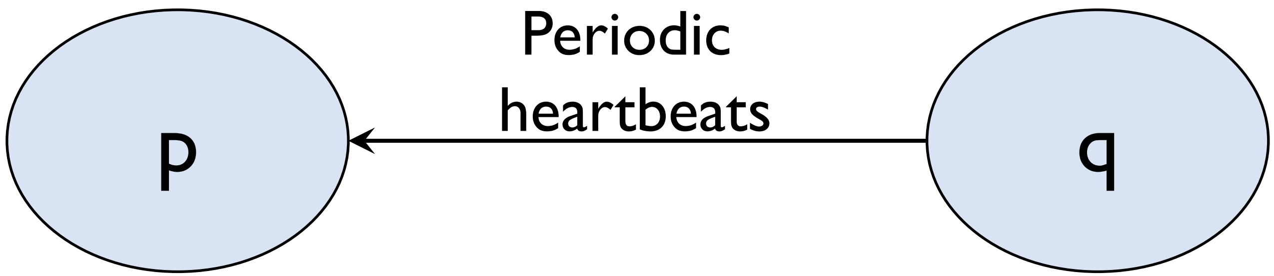
q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ is the *timeout* value at p.

If $(T + \Delta_2)$ time elapsed since last heartbeat, report q crashed.



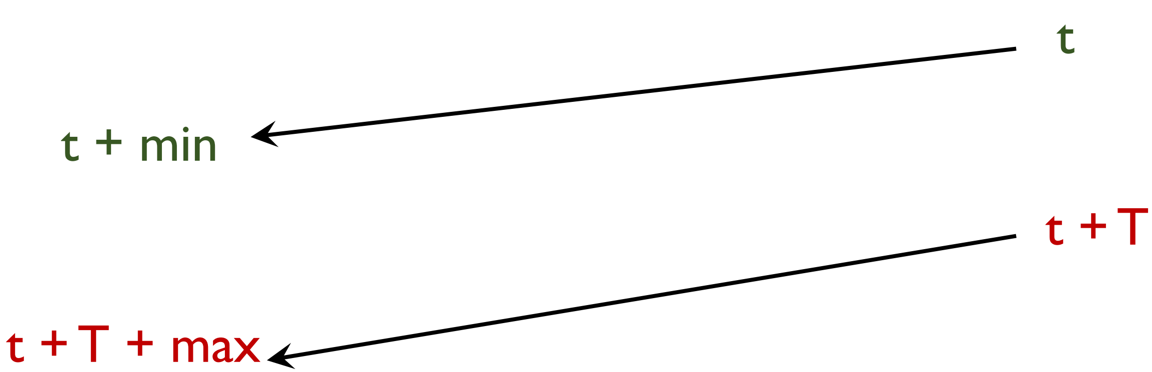
How to detect a crashed process?



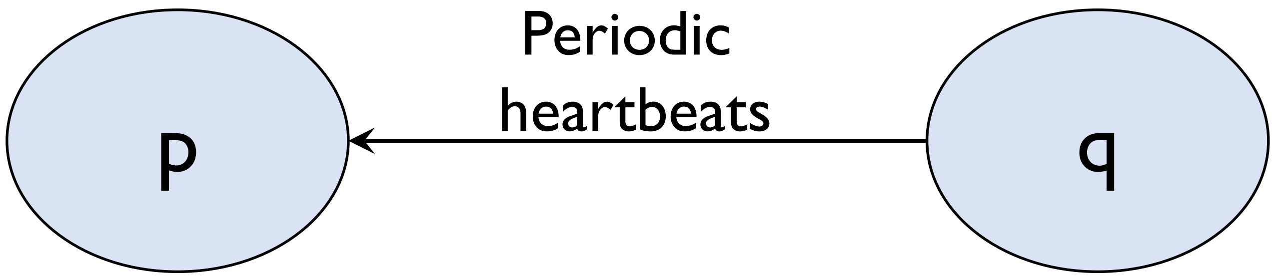
q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ is the *timeout* value at p.

If $(T + \Delta_2)$ time elapsed since last heartbeat, report q crashed.



How to detect a crashed process?



q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ is the *timeout* value at p.

If $(T + \Delta_2)$ time elapsed since last heartbeat, report q crashed.

If synchronous, $\Delta_2 =$ max network delay – min network delay

If asynchronous, $\Delta_2 =$ k(observed delay)

Correctness of failure detection

- **Completeness**
  - Every failed process is *eventually* detected.
- **Accuracy**
  - Every detected failure corresponds to a crashed process (no mistakes).

Correctness of failure detection

- Characterized by **completeness** and **accuracy**.

- Synchronous system
  - Failure detection via ping-ack and heartbeat is both complete and accurate.

- Asynchronous system
  - *Our strategy for ping-ack and heartbeat is*
  - Impossible to achieve both completeness and accuracy.
  - Can we have an accurate but incomplete algorithm?
    - *Never report failure.*

Metrics for failure detection

Try deriving these before next class!

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

Metrics for failure detection

- Worst case failure detection time
  - After a process crashes, how long does it take for the other process to detect the crash in the worst case?

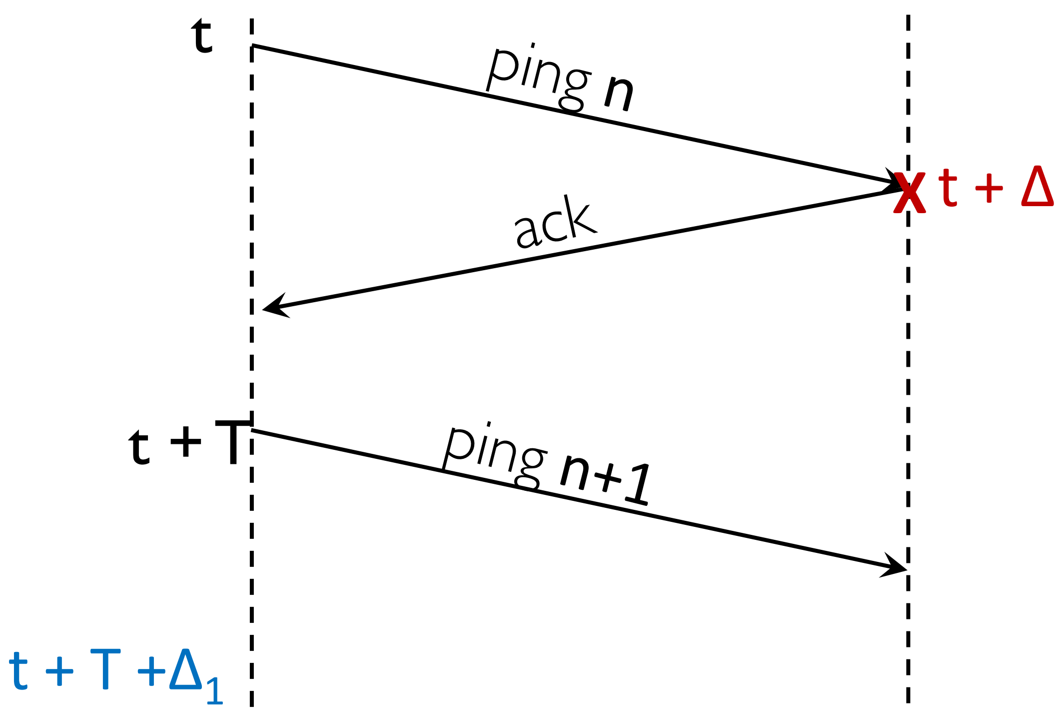
Metrics for failure detection

- 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.

Try deriving this!

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

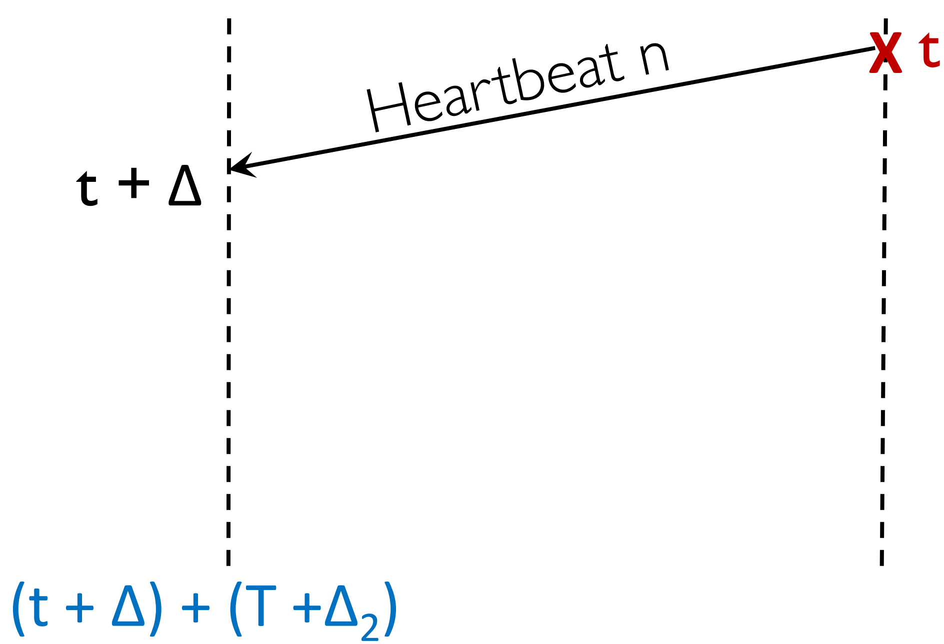
- Worst case failure detection time
  - Heartbeat: $T + \Delta_2 + \Delta$ where Δ is time taken for last heartbeat from q to reach p
    T is the time period for heartbeats, and $T + \Delta_2$ is the timeout.

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

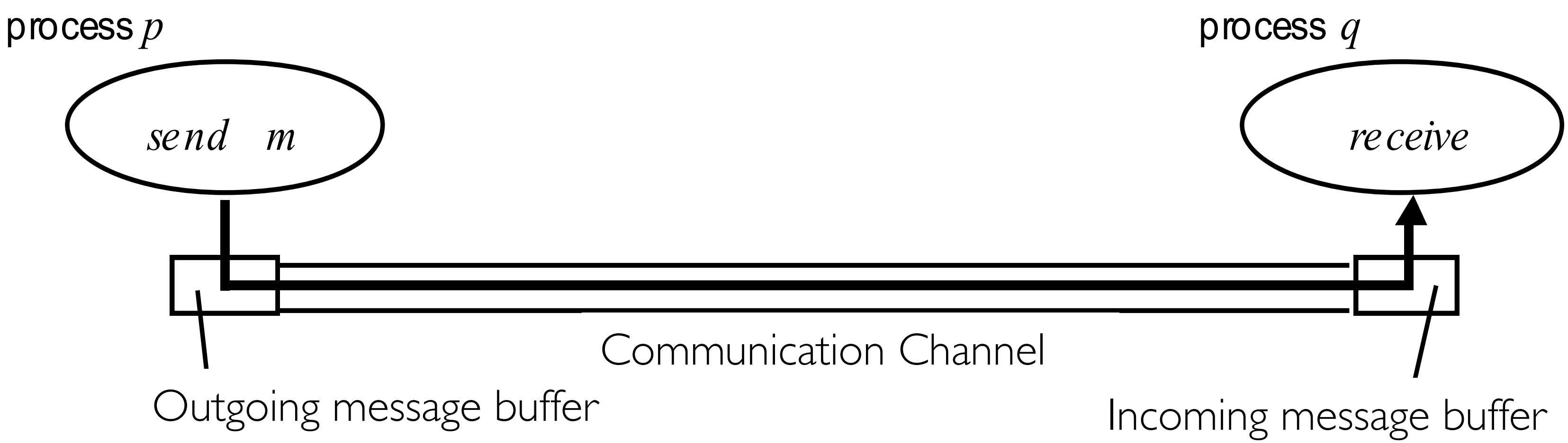
- Worst case failure detection time
  - Ping-ack: $T + \Delta_1 - \Delta$ (where Δ is time taken for previous ping from p to reach q)
  - Heartbeat: $T + \Delta_2 + \Delta$ (where Δ is time taken for last heartbeat from q to reach p)
- Bandwidth usage:
  - Ping-ack: 2 messages every T units
  - Heartbeat: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

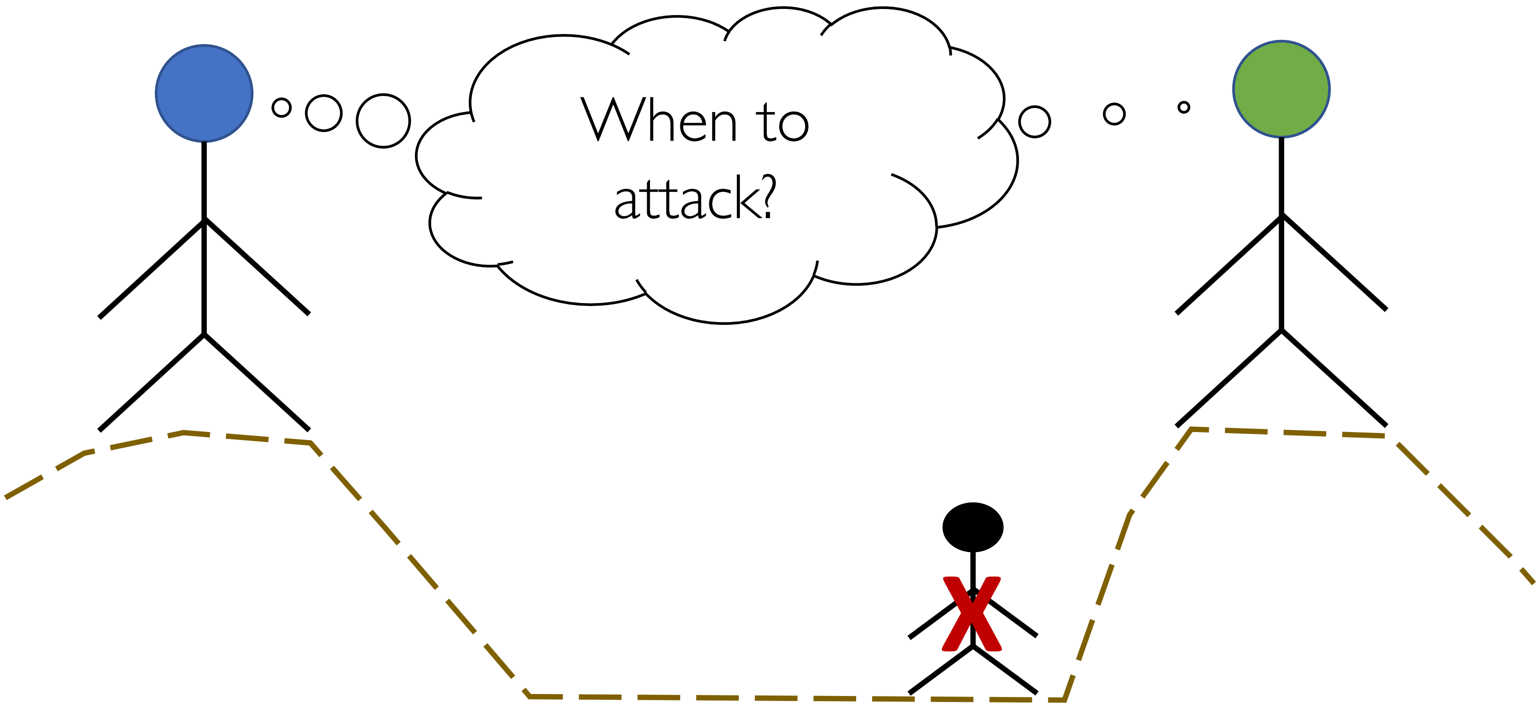
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
  - Process may **crash**.
  - **Fail-stop**: if other processes can certainly detect the crash.
  - **Communication omission**: a message sent by process was not received by another.

Communication Omission

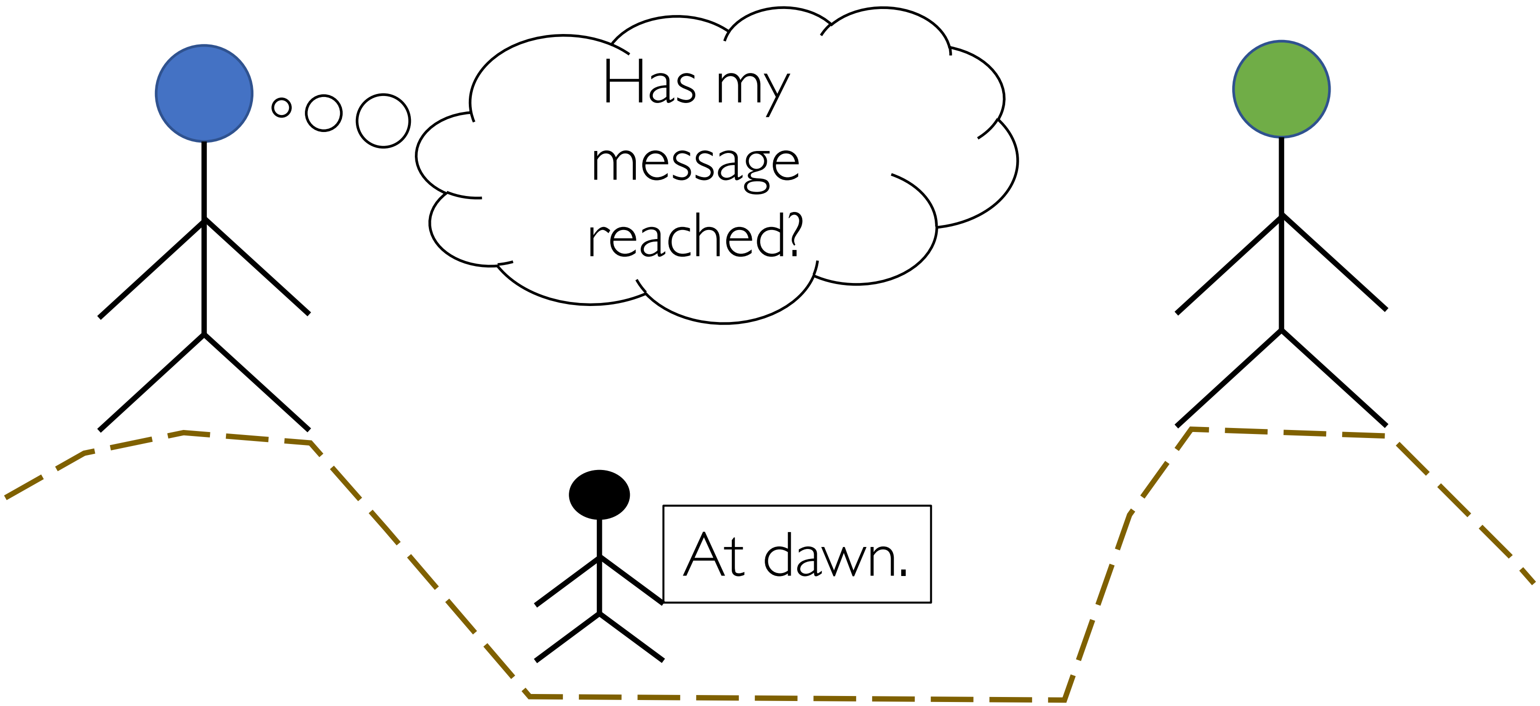


- Channel Omission: omitted by channel
- Send omission: process completes 'send' operation, but message does not reach its outgoing message buffer.
- Receive omission: message reaches the incoming message buffer, but not received by the process.

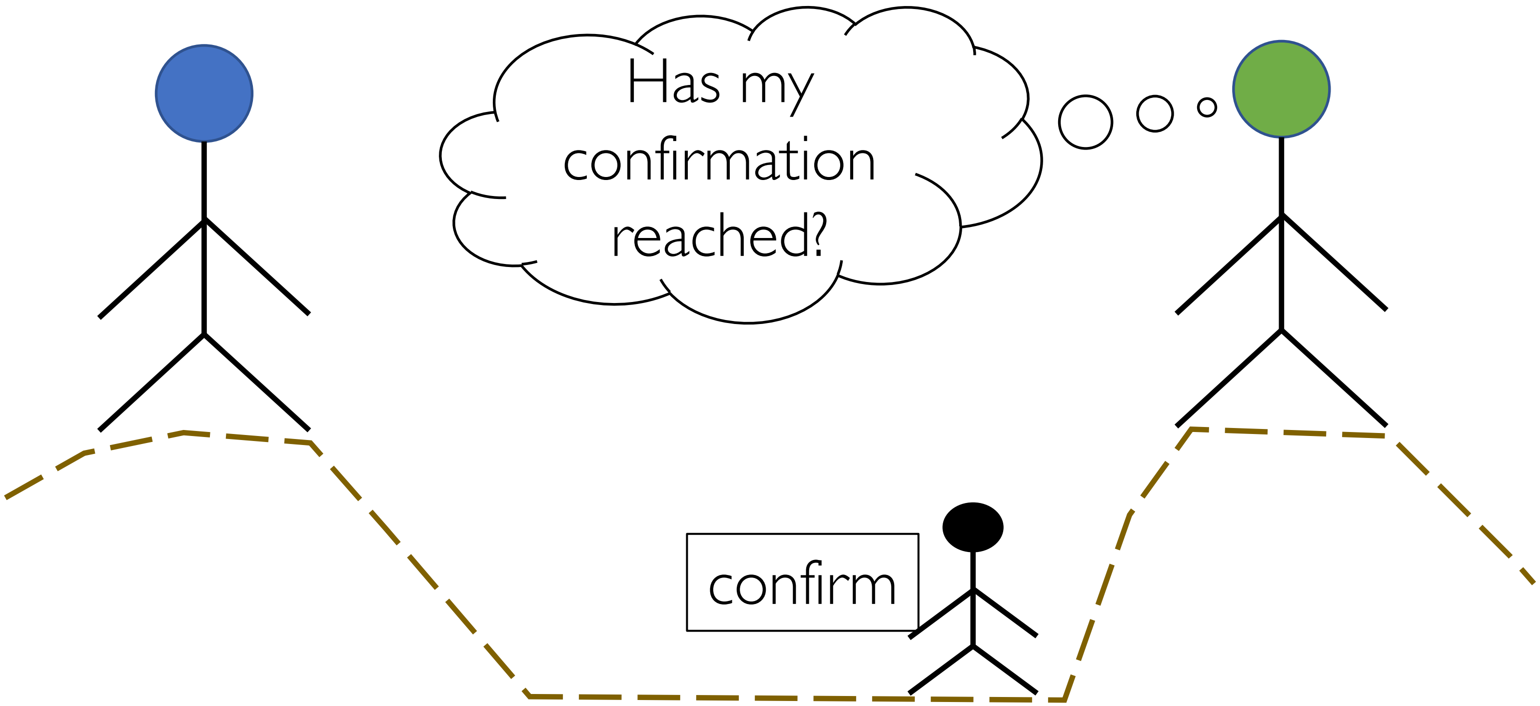
Two Generals Problem



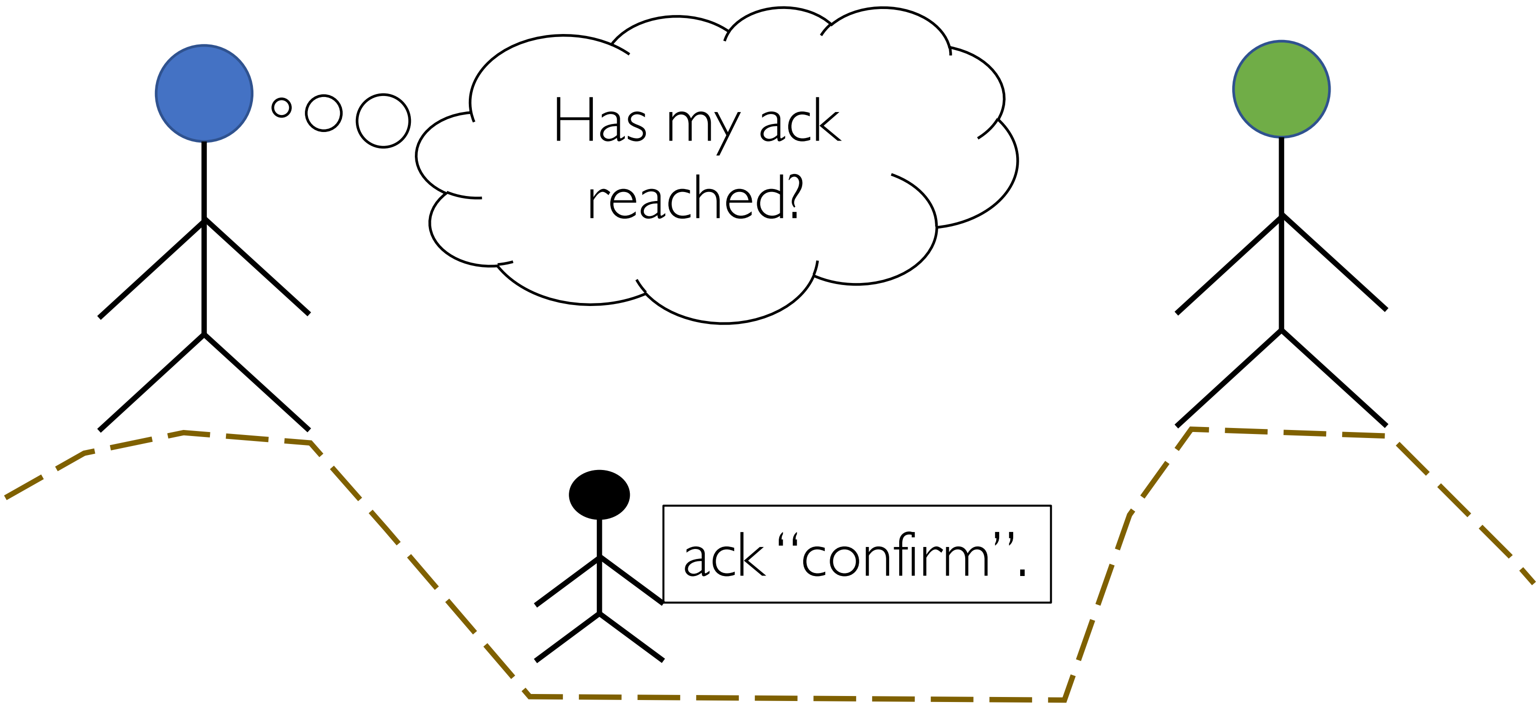
Two Generals Problem



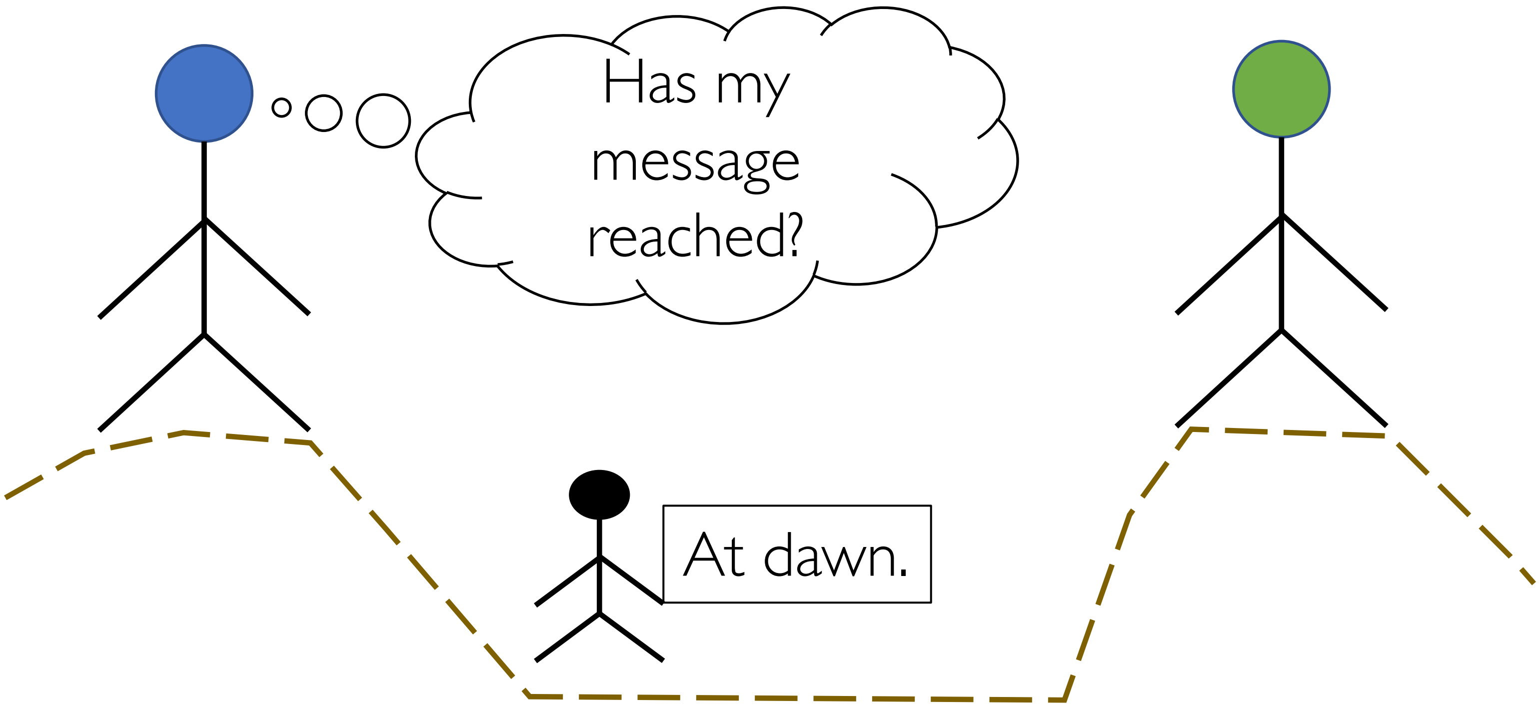
Two Generals Problem



Two Generals Problem

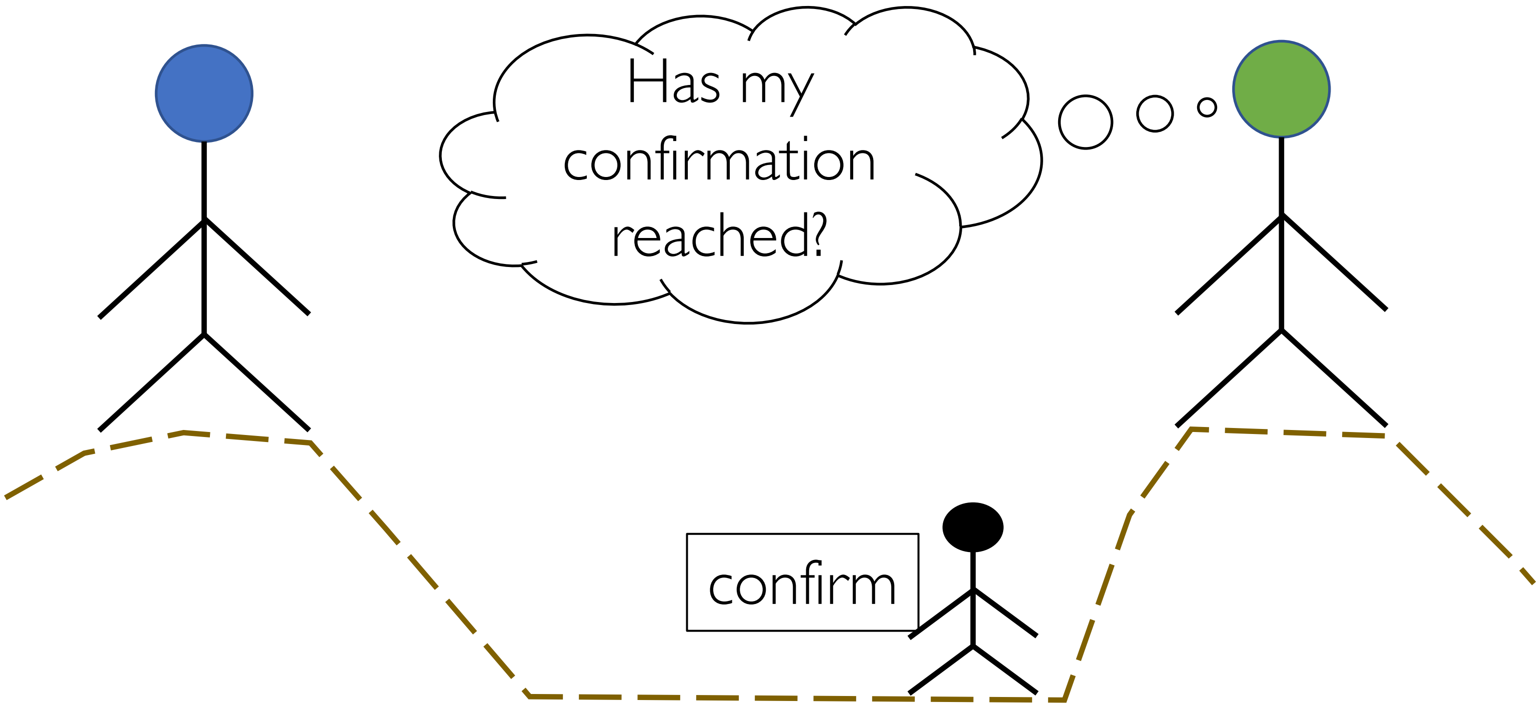


Two Generals Problem



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.

Summary

- Sources of uncertainty
  - Communication time, clock drift rates
- Synchronous vs asynchronous models.
- Types of failures: omission, arbitrary, timing
- Detecting failed a process.