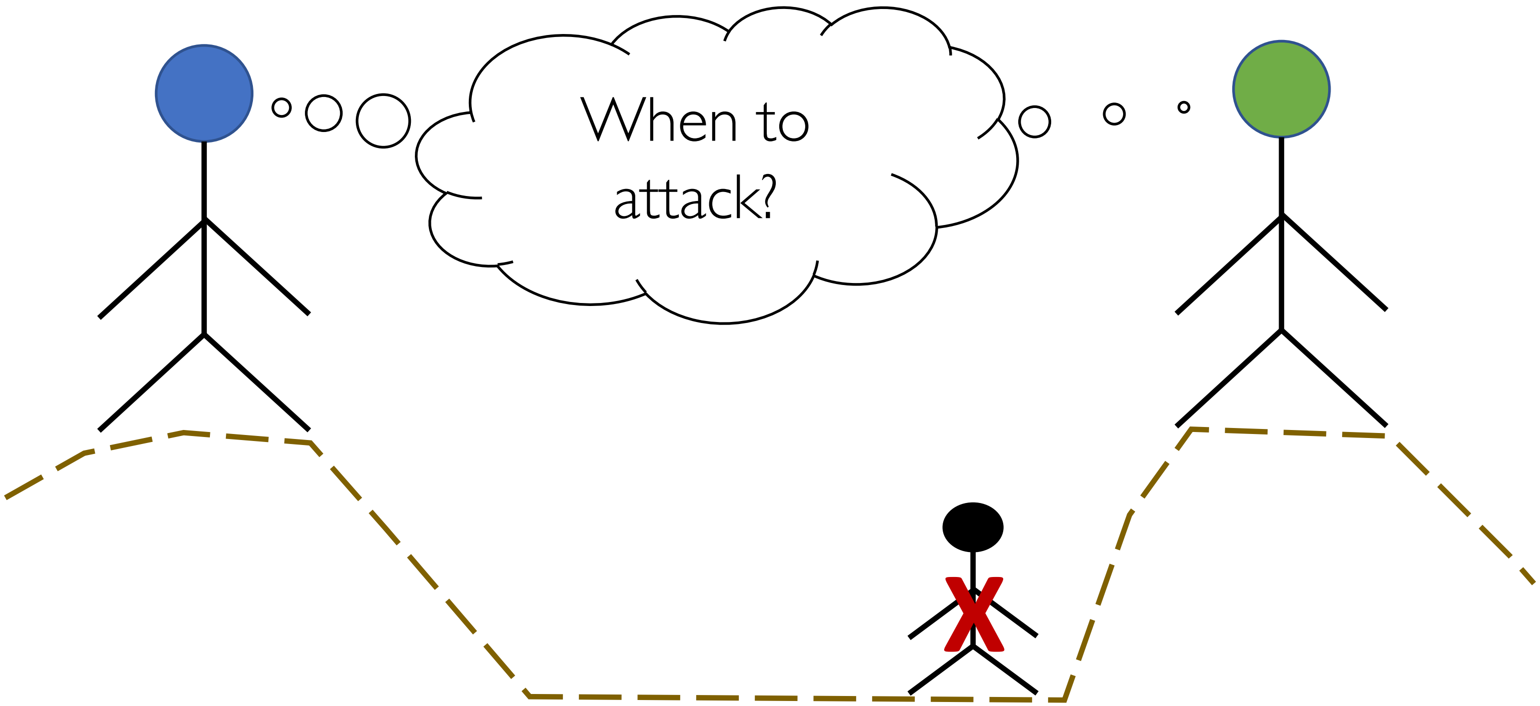
Distributed Systems

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?

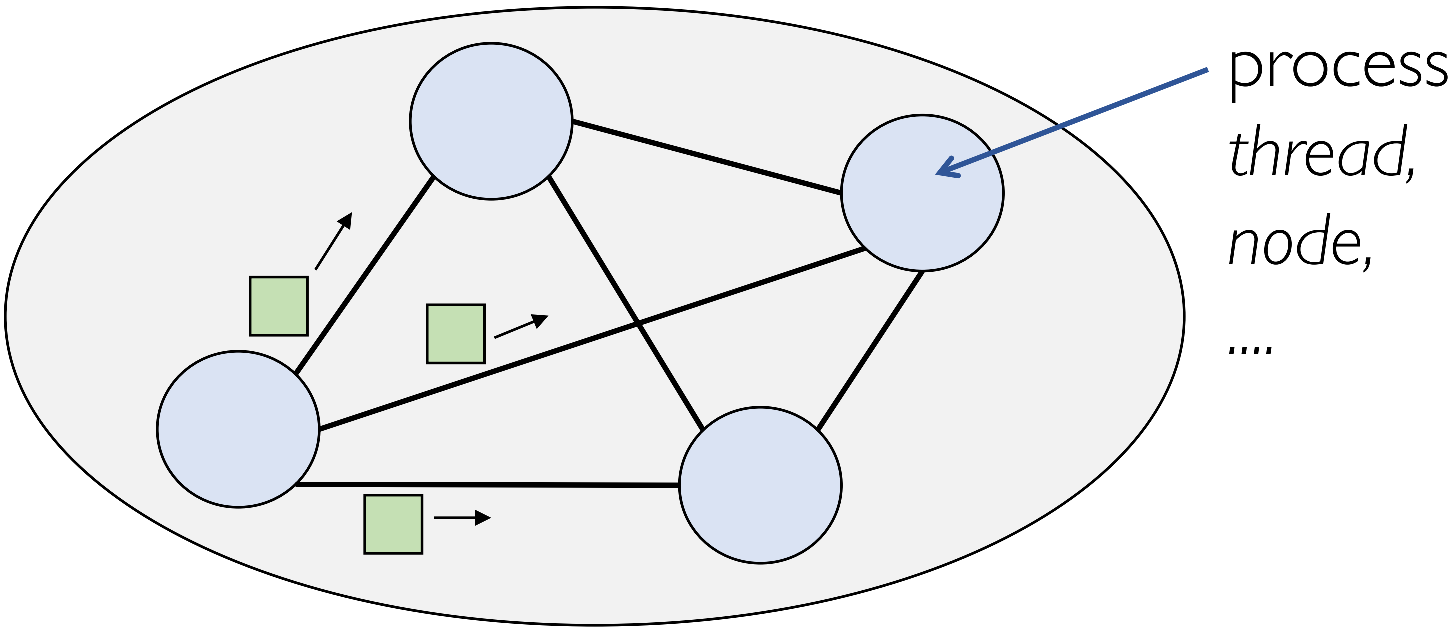
Logistics Related

- OHs information is up on website *(with maybe a couple of TBDs that will be resolved soon).*
  - We will start from next week.
- Sign-up forms for VM clusters is available on CampusWire.
  - Please fill it up by Wednesday, Jan 29th, 11:59pm.
- Reach out to sie2@illinois.edu if you need access to Campuswire.
- 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?



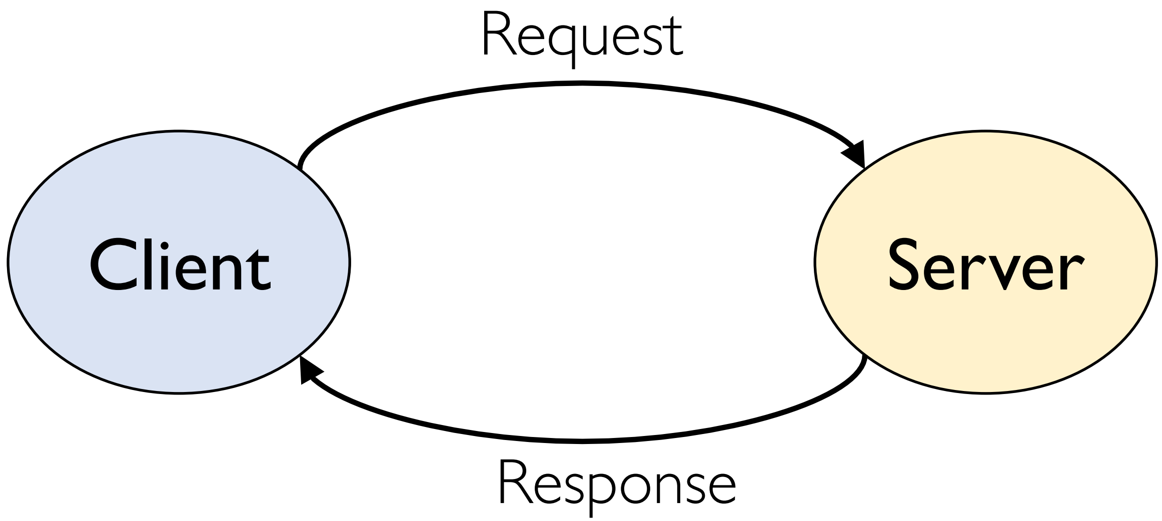
Independent components that are **connected by a network** and communicate by **passing messages** to achieve a common goal, appearing as **a single coherent system**.

Relationship between processes

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

Relationship between processes

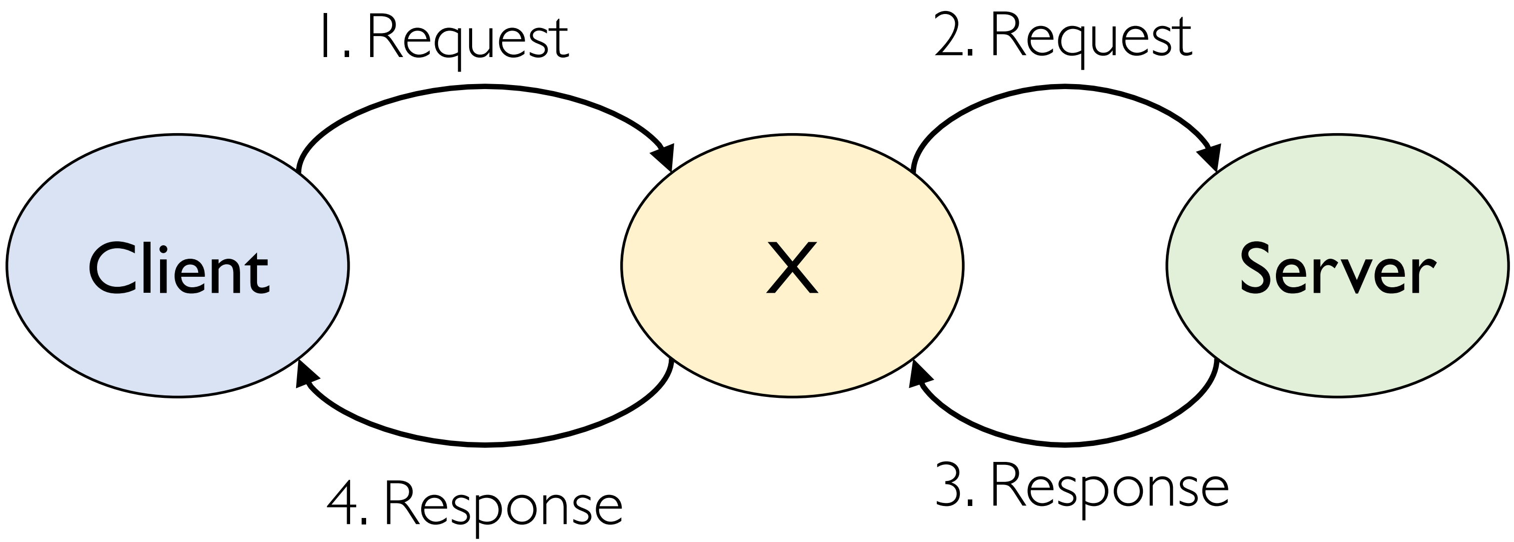
- Client-server



Clear difference in roles.

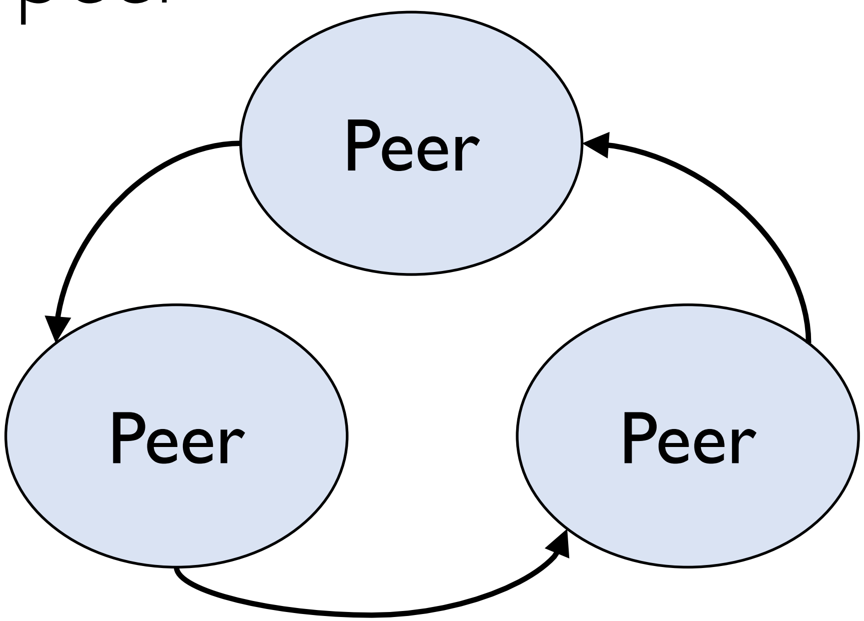
Relationship between processes

- Client-server



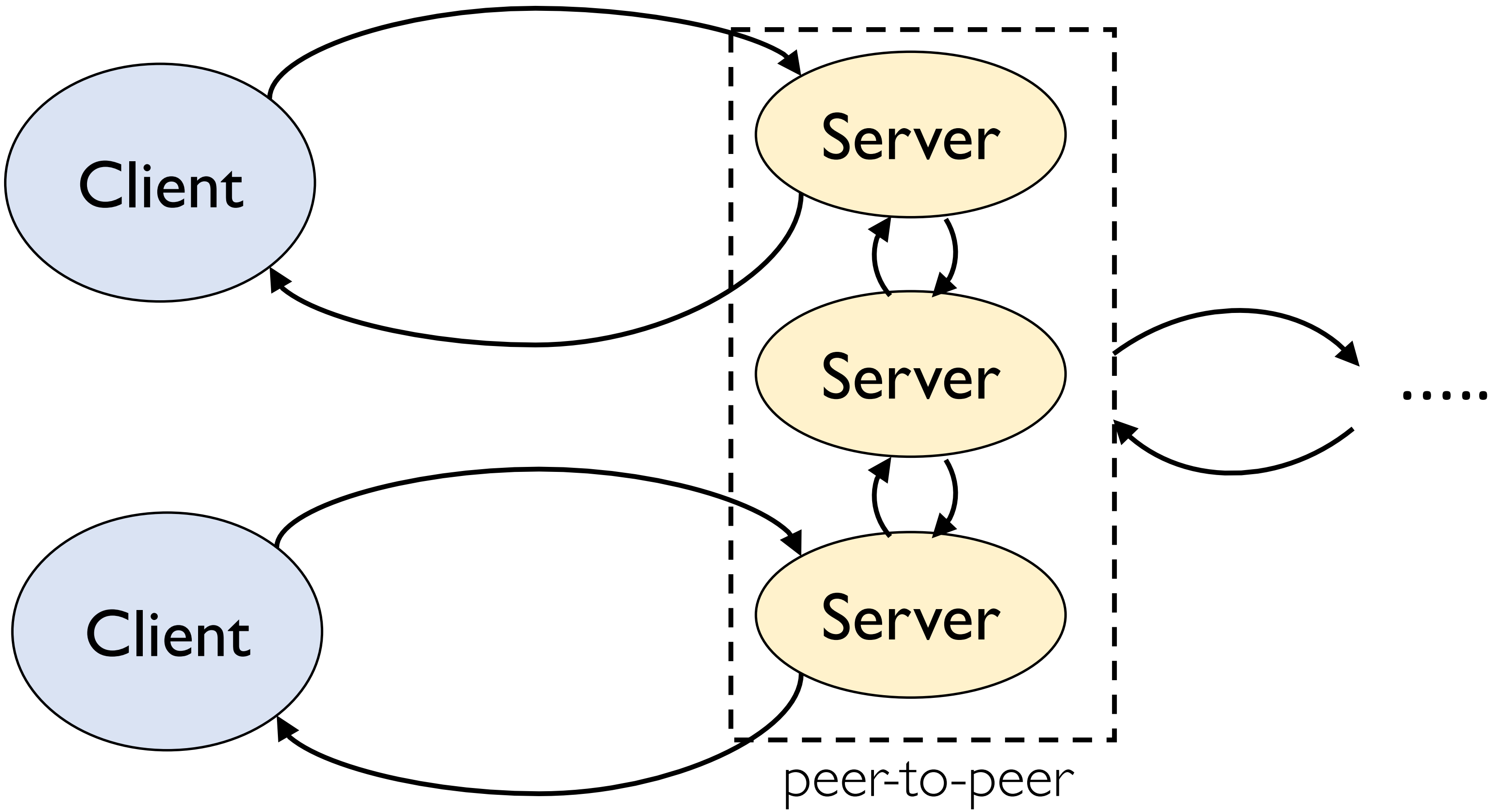
Relationship between processes

- Peer-to-peer



Similar roles.
Run the same program/algorithm.

Relationship between processes



Relationship between processes

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

Distributed algorithm

- Algorithm on a single process
  - Sequence of steps taken to perform a computation.
  - *Steps are strictly sequential.*

- Distributed algorithm
  - Steps taken by each of the processes in the system (including transmission of messages).
  - *Different processes may execute their steps concurrently.*

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.

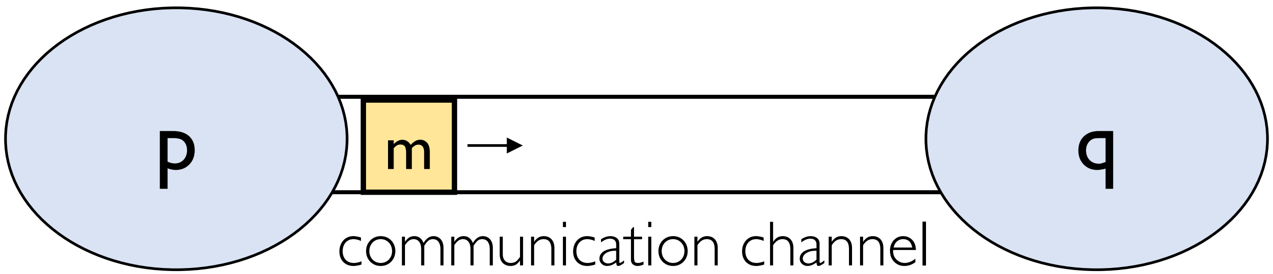
- Different processes (on different computers) have different clocks!

- Processes and communication channels may fail.

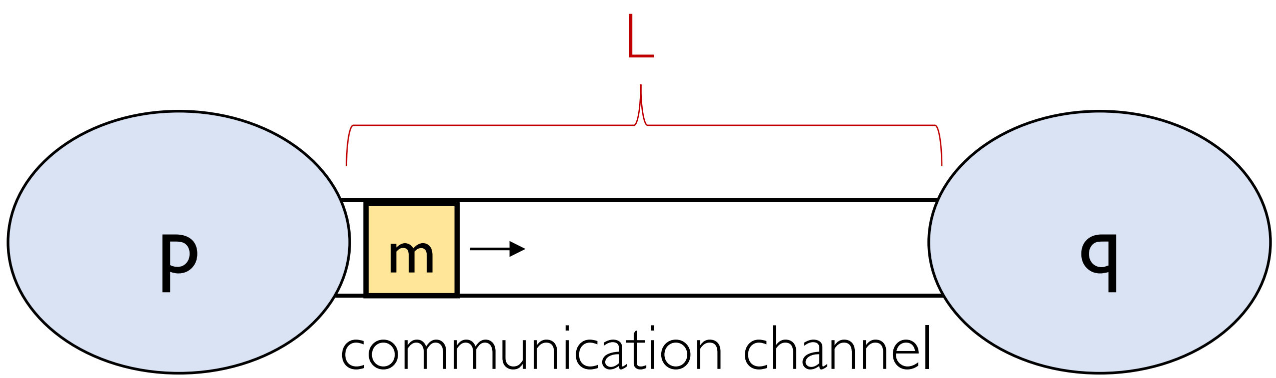
How processes communicate

- Directly using network sockets.
- Abstractions such as remote procedure calls, publish-subscribe systems, or distributed share memory.
- Differ with respect to how the message, the sender or the receiver is specified.

How processes communicate

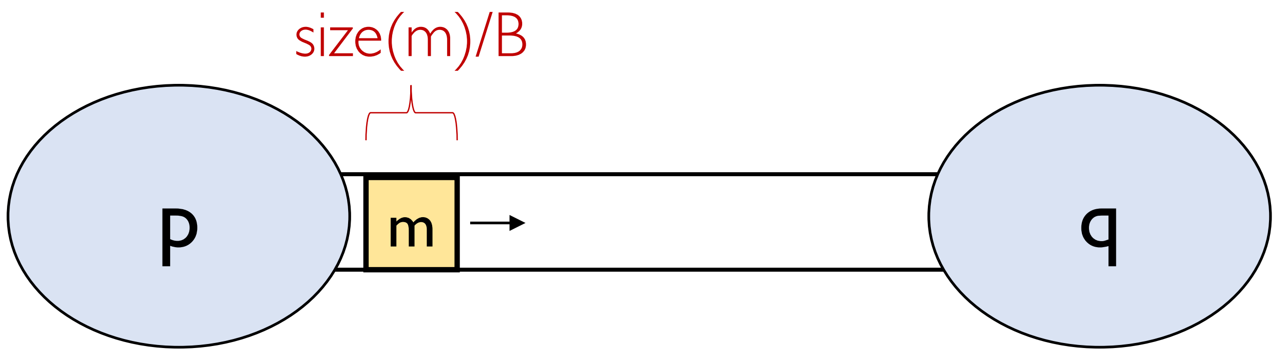


Communication channel properties



- Latency (L): Delay between the start of **m**'s transmission at **p** and the beginning of its receipt at **q**.
  - Time taken for a bit to propagate through network links.
  - Queuing that happens at intermediate hops.
  - Overheads in the operating systems in sending and receiving messages.
  - …..

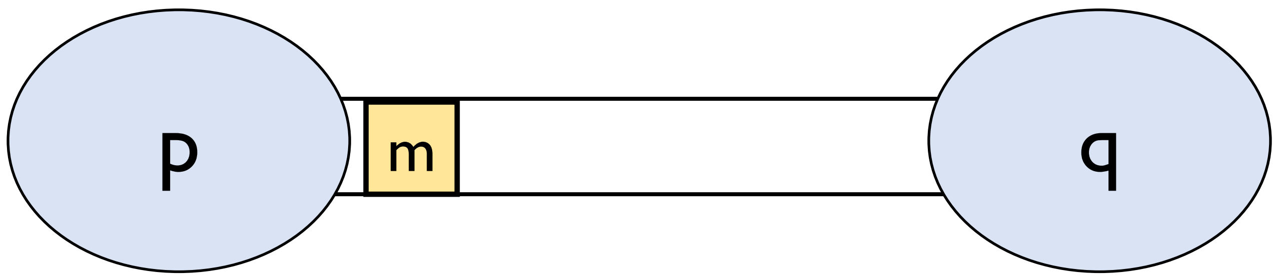
Communication channel properties



- Latency (L): Delay between the start of **m**'s transmission at **p** and the beginning of its receipt at **q**.

- Bandwidth (B): Total amount of information that can be transmitted over the channel per unit time.

Communication channel properties



- Total time taken to pass a message is governed by latency and bandwidth of the channel.
  - Both latency and available bandwidth may vary over time.

- *Sometimes useful to measure "bandwidth usage" of a system as amount of data being sent between processes per unit time.*

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.

Differing clocks

- Each computer in a distributed system has its own internal clock.
- Local clock of different processes show different time values.
- Clocks *drift* from perfect times at different rates.

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.

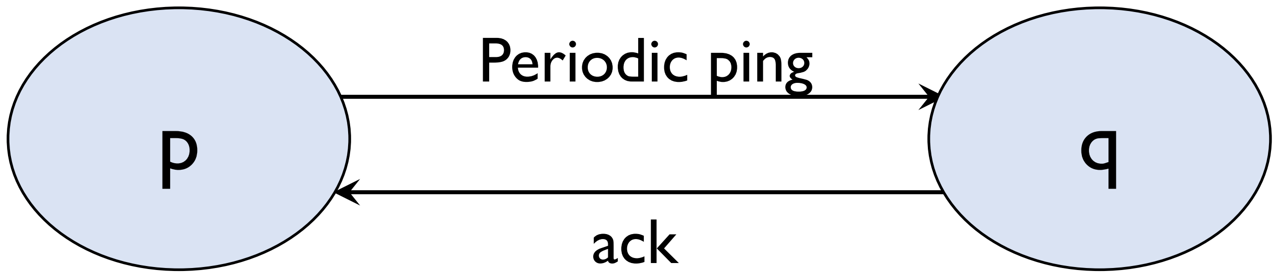
Today's agenda

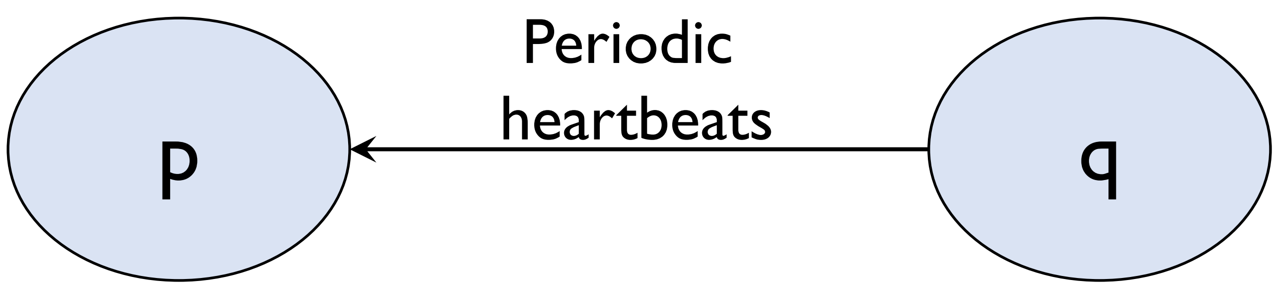
- System Model
    - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3
- Failure Detection
    - Chapter 15.1

Types of failure

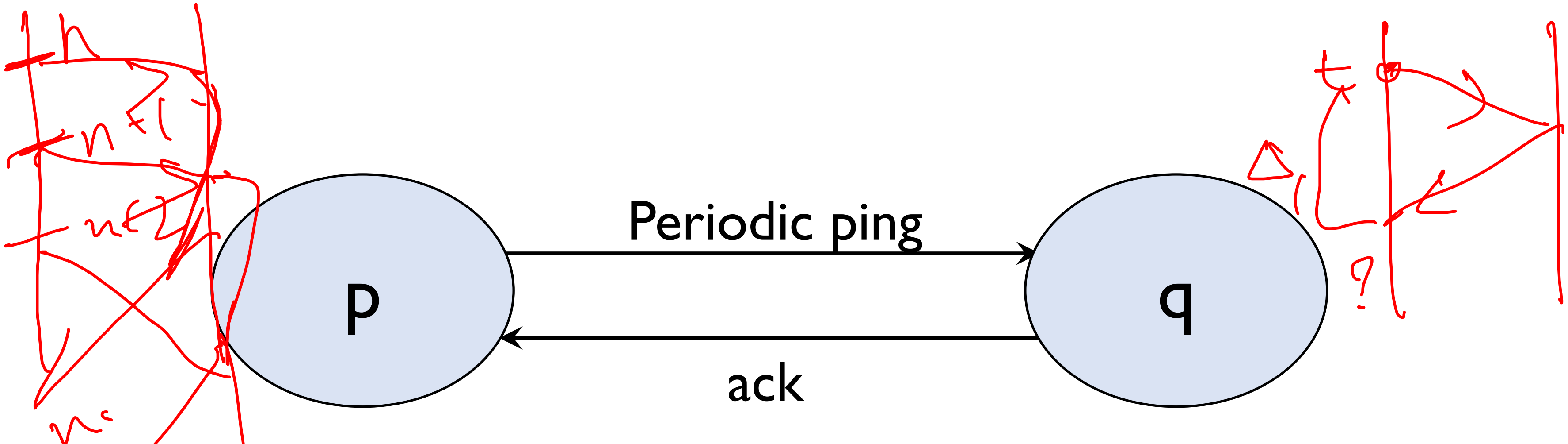
- **Omission:** when a process or a channel fails to perform actions that it is supposed to do.
  - Process may **crash**.

How to detect a crashed process?





How to detect a crashed process?



p sends pings to q every T seconds.

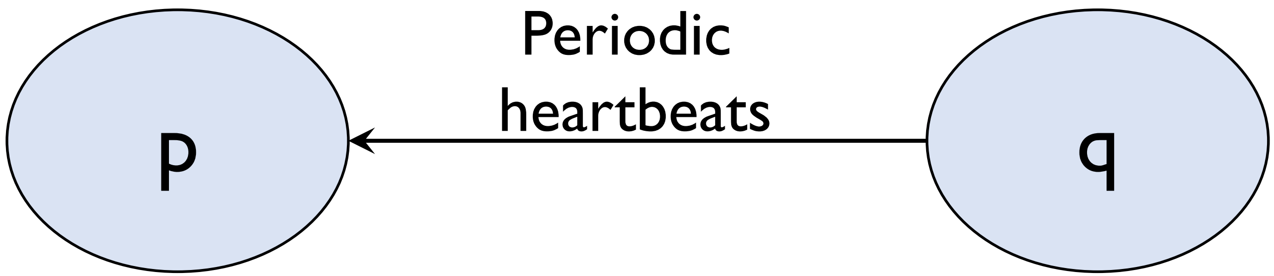
Δ_1 is the *timeout* value at p.

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); k ≥ 1

How to detect a crashed process?



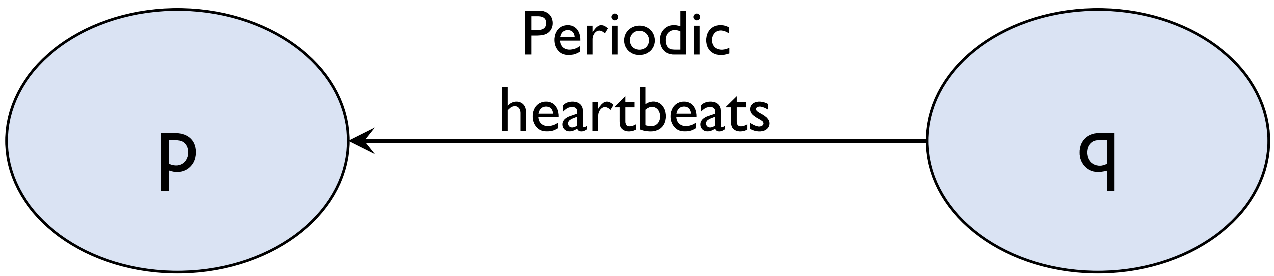
q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ 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

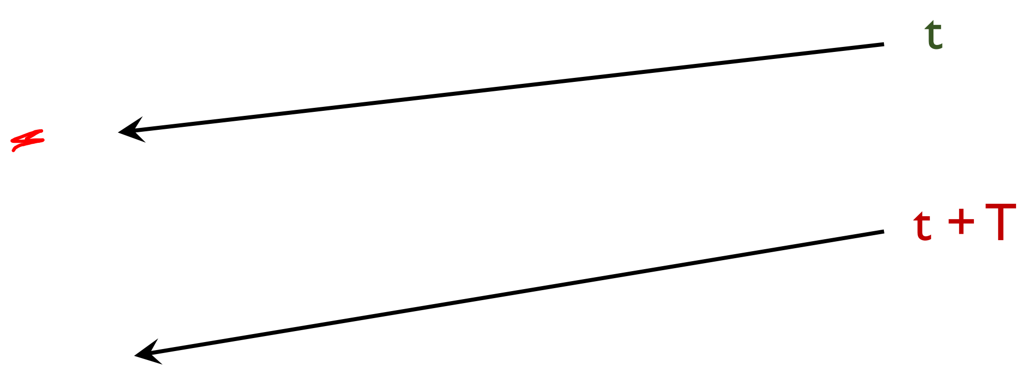
How to detect a crashed process?



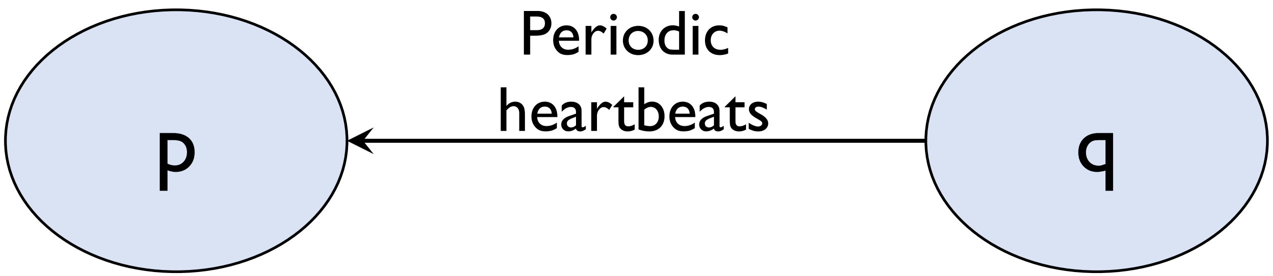
q sends heartbeats to p every T seconds.

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

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



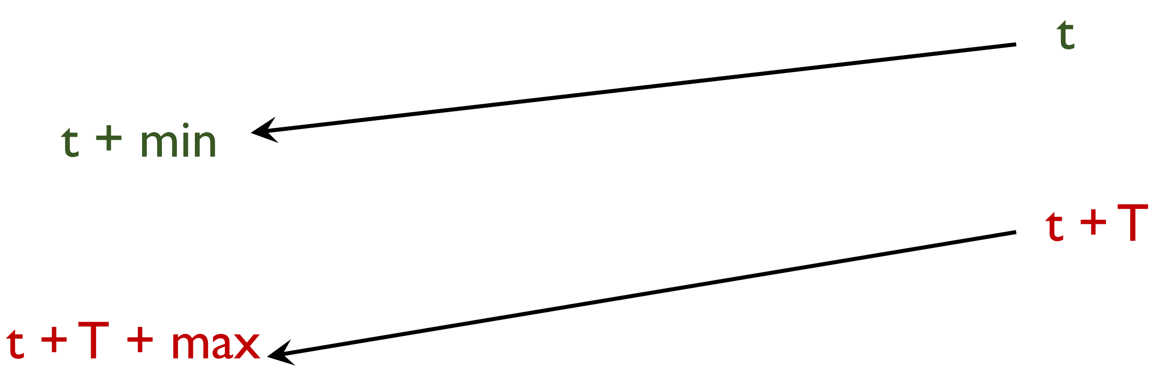
How to detect a crashed process?



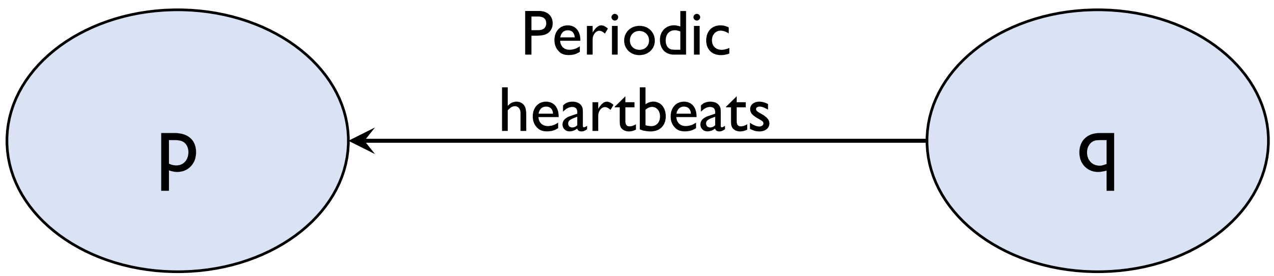
q sends heartbeats to p every T seconds.

$(T + \Delta_2)$ 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, Δ_2 = max network delay – min network delay

If asynchronous, Δ_2 = k(observed delay); k >= 1

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

- 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?

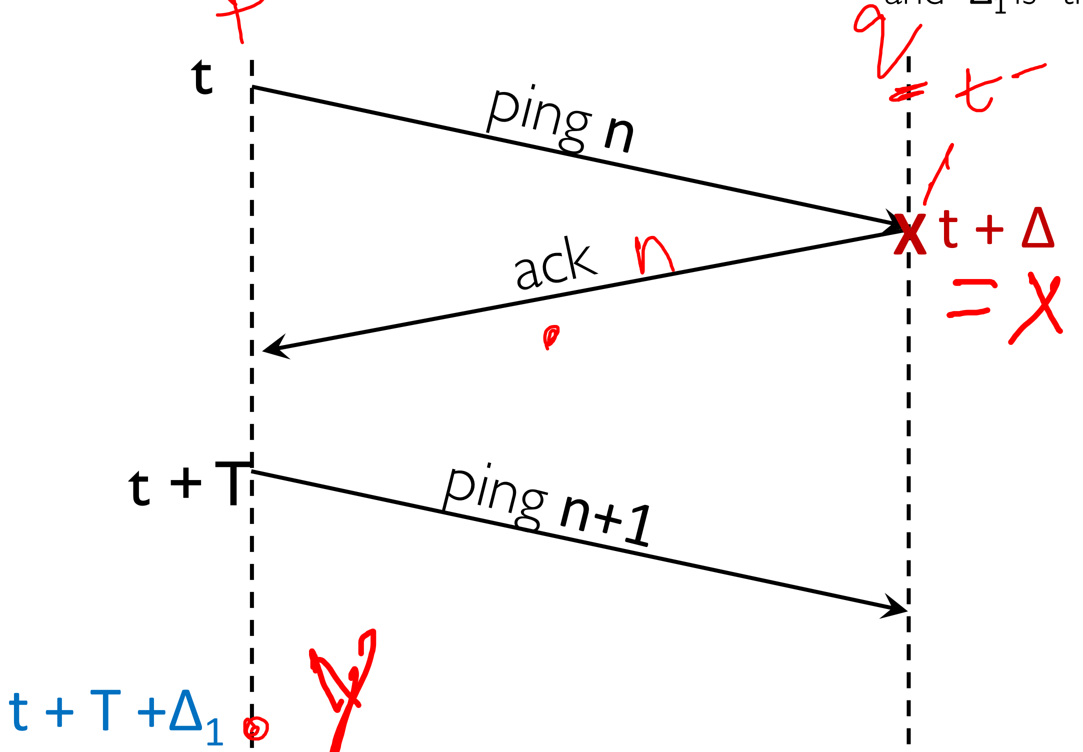
Metrics for failure detection

- 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!

To be continued next class