BRINGING ORDER TO THE GALAXY

THE EMPIRE

BRINGING ORDER TO THE GALAXY
SINCE: 1977
SYNCHRONY vs ASYNCHRONY

Synchronous systems
- Known bound on message delivery
- Known bound on processing speed
- Considered a strong assumption

Asynchronous systems
- No bound on message delivery
- No bound on processing speed
- Weak assumption = less vulnerable
- Asynchronous ≠ slow

This lecture: asynchronous + no process failures
Ordering events in a distributed system

What does it mean for an event to “happen before” another event?
What is a distributed system?

A collection of distinct processes that:

- are spatially separated
- communicate with one another by exchanging messages
- have non-negligible communication delay
- do not share fate
- have separate physical clocks

(imperfect, unsynchronized)
Non-distributed system

- A single clock
- Each event has a timestamp
- Compare timestamps to order events

Distributed system

- Each process has its own clock
- Each clock runs at a different speed
- Cannot directly compare clocks

an absolute temporal ordering is not what you want in a distributed system anyway

Leslie Lamport
an absolute temporal ordering is not what you want in a distributed system anyway

Leslie Lamport

Why not?

Because temporal ordering is not observable. You cannot read two separate clocks simultaneously!

Very important point:
if a system is to meet a specification correctly, then that specification must be given in terms of events observable within the system
**Ordering events without physical clocks**

Modeling a process:

- A set of instantaneous events with an a priori total ordering
- Events can be local, sends, or receives.
ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

“Happened-before” relation, denoted: $\rightarrow$

Part 1

- If $a$ and $b$ are events on the same process and $a$ comes before $b$, then $a \rightarrow b$
"Happened-before" relation, denoted: →

Part 2

- If \( a \) is the sending of a message by one process and \( b \) is the receipt of the same message by another process, then \( a \rightarrow b \)
Ordering events without physical clocks

“Happened-before” relation, denoted: \( \rightarrow \)

Part 3

- If \( a \rightarrow b \) and \( b \rightarrow c \), then \( a \rightarrow c \)
ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

Putting it all together

Diagram showing events labeled a, b, c, d, e, f, g, h, i, j with arrows indicating their sequence.
Can arrows go backwards?
Can cycles be formed?

No, because the same event would happen at two different times
ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

Are all events related by $\rightarrow$?
A partial order

The set of events \( q \) such that \( q \rightarrow p \) are the events that could have influenced \( p \) in some way.

\( \{a, b, e, f, h\} \)
If two events could not have influenced each other, it doesn’t matter when they happened relatively to each other.

$h$ and $d$ are concurrent: $h \leftrightarrow d$, $d \leftrightarrow h$
Goal: generate a total order that is consistent with the happened-before partial order
Lamport clocks

Define a function $LC$ such that:

$$p \rightarrow q \Rightarrow LC(p) < LC(q)$$

(the Clock condition)
Lamport clocks

Define a function $\text{LC}$ such that:

$$ p \rightarrow q \Rightarrow LC'(p) < LC'(q) $$

(the Clock condition)

Implement $\text{LC}$ by keeping a local $\text{LC}_i$ at each process $i$
LAMPORT CLOCKS

Single process

\[ p \bullet \quad 1 \quad 2 \quad 3 \quad 4 \]
Lamport clocks

Single process

$p$   $a$  $b$  $c$  $d$

1  2  3  4
6  37 1145
Lamport clocks

Across processes

\[
\begin{align*}
& a & b & c & d \\
& 1 & 2 & 3 & 4
\end{align*}
\]

\[
\begin{align*}
& e & f & g & h \\
& 1 & 2 & 3 & 4
\end{align*}
\]

\[b \rightarrow h \Rightarrow LC(b) < LC(h)\]
\[g \rightarrow h \Rightarrow LC(g) < LC(h)\]
Putting it all together

Diagram of points a, b, c, d, e, f, g, h, i, j with connections described.
Is this correct?
Generating a total order

Order messages by LC
Ties are broken by unique process ID
Lamport clocks implement the Clock condition

\[ p \rightarrow q \Rightarrow LC(p) < LC(q) \]

But is that all we need?
Send me a picture of yourself by next class

Subject: [EECS591] Picture of <first name> (preferred) <last name> <UMID>

Make sure to subscribe to our Piazza forum

Announcements, discussion, etc.

I’ve given a few more overrides
FIFO DELIVERY

FIFO delivery

\[ send_i(m) \rightarrow send_i(m') \Rightarrow deliver_j(m) \rightarrow deliver_j(m') \]
FIFO delivery

\[ send_i(m) \rightarrow send_i(m') \Rightarrow deliver_j(m) \rightarrow deliver_j(m') \]
Causal delivery

When more processes are involved, causal delivery is needed:

\[ send_i(m) \rightarrow send_j(m') \Rightarrow deliver_k(m) \rightarrow deliver_k(m') \]
Gap detection: Given two events $e$ and $e'$, where $LC(e) < LC(e')$, determine whether some other event $e''$ exists such that 

$$LC(e) < LC(e'') < LC(e')$$
**Gap detection:** Given two events $e$ and $e'$, where $LC(e) < LC(e')$, determine whether some other event $e''$ exists such that

$$LC(e) < LC(e'') < LC(e')$$

Lamport clocks don't provide gap detection!
How to implement causal delivery?
(in other words, when is it safe to deliver $m'$?)

a) Wait to receive a message with higher LC from each channel

b) Implement better clocks!
From clocks to strong clocks

\[ p \rightarrow q \Rightarrow \text{LC}(p) < \text{LC}(q) \]

Clock condition

\[ p \rightarrow q \Leftrightarrow \text{LC}(p) < \text{LC}(q) \]

Strong clock condition
CAUSAL HISTORIES

The set of events $q$ such that $q \rightarrow p$ are the events that could have influenced $p$ in some way.

$\theta(g) = \{a, b, e, f, h, g\}$
IMPLEMENTING STRONG CLOCKS
(the hard way)

- Initialize $\theta := \emptyset$
- For send and local events $e$, $\theta(e) := \theta \cup \{e\}$
- For receive events $e = \text{recv}(m)$, $\theta(e) := \theta \cup \{e\} \cup \theta(m)$
IMPLEMENTING STRONG CLOCKS
(the hard way)

Strong clock condition: $p \rightarrow q \iff \theta(p) \subset \theta(q)$