ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

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

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

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

(the Clock condition)
Causal delivery

Generalizes FIFO to more processes:

\[ send_i(m) \rightarrow send_j(m') \Rightarrow deliver_k(m) \rightarrow deliver_k(m') \]
**Gap detection**

Should $r$ deliver $m'$?

**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**

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!
FROM CLOCKS TO STRONG CLOCKS

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

Clock condition

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

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: \( p \rightarrow q \iff \theta(p) \subseteq \theta(q) \)
Implementing strong clocks
(the hard way)

Strong clock condition: $p \rightarrow q \iff \theta(p) \subseteq \theta(q)$
Vector clocks

Each process keeps a vector of natural numbers $VC$, one for each process.

Update rules

If $e_i$ is a local or send event at process $i$:

$$VC(e_i)[i] := VC[i] + 1$$

If $e_i$ is a receive event of message $m$:

$$VC(e_i) := max\{VC, VC(m)\}$$
$$VC(e_i)[i] := VC[i] + 1$$
Vector clocks

\[ VC(e_i)[i] = \text{number of events executed by process } i \text{ (including } e_i) \]

\[ VC(e_i)[j] = \text{number of events executed by process } j \text{ that causally precede } e_i \]
Comparing vector clocks

Equality

\[ V = V' \equiv \forall k : 1 \leq k \leq n : V[k] = V'[k] \]
(i.e. all elements are the same)

Inequality

\[ V < V' \equiv (V \neq V') \land (\forall k : 1 \leq k \leq n : V[k] \leq V'[k]) \]

Examples: \[ [2,0,0] < [2,0,1] < [3,0,1] < [4,1,1] \]

Strong clock condition: \[ p \rightarrow q \iff VC(p) < VC(q) \]
Comparing vector clocks

Strong clock condition: \( p \rightarrow q \iff VC(p) < VC(q) \)
Causal delivery

A “monitor” process wants to record all messages (e.g. deadlock detection, system snapshot, etc)

- Processes send copies of their messages to the monitor
- Only increment the local component of $VC$ for send events
Causal delivery rules

Monitor keeps an array $D$, where $D[i]$ is the number of messages delivered from process $i$

Monitor delivers message $m$ from process $j$ when:

1. $D[j] = VC(m)[j] - 1$
2. $D[k] \geq VC(m)[k], \forall k \neq j$
Causal delivery

\[
\begin{align*}
D & \quad (0, 0) & \quad (1, 0) & \quad (1, 1) \\
D[j] = VC(m)[j] - 1 & \quad \checkmark & \quad \checkmark & \quad \checkmark \\
D[k] \geq VC(m)[k], \forall k \neq j & \quad \times & \quad \checkmark & \quad \checkmark
\end{align*}
\]
ADMINISTRIVIA

- Still missing a few pictures
- Still missing a few people on Piazza
- Piazza poll about moving class to 3:30-5
- I’ll give 1-2 overrides soon
Clock synchronization

What time is it?
Byzantine generals v2.0

Attack at midnight!

Chaaaaaaarge!

0:00

ZZZZZZ

23:30
Clock drift

**Bound on drift: \( \rho \)**

\[
(1 - \rho)(t - t') \leq H(t) - H(t') \leq (1 + \rho)(t - t')
\]

**\( \rho \) is typically small (10^{-6})**

- \( \rho^2 \approx 0 \)
- \[
\frac{1}{1 - \rho} = 1 + \rho
\]
- \[
\frac{1}{1 + \rho} = 1 - \rho
\]
External vs internal synchronization

External Clock Synchronization:
keeps clock within some maximum deviation from an external time source.

- exchange of info about timing events of different systems
- can take actions at real-time deadlines

Internal Clock Synchronization:
keeps clocks within some maximum deviation from each other.

- can measure **duration** of distributed activities that start on one process and terminate on another
- can totally order events that occur in a distributed system
Probabilistic Clock Synchronization (Cristian)

- Master-Slave architecture
- Master can be connected to external time source
- Slaves read master’s clock and adjust their own

How accurately can a slave read the master’s clock?
Setup and assumptions

Goal: Synchronize the slave's clock with the master

Assume that minimum delay is known
Assume that clock drifts are known ($\rho$ for both)
The protocol

Question: what is $Q(x)$?
Ideal scenario

Assume no clock drift
Problem #1: message delay

\[ P(t) \quad Q(t) \quad T \]

One extreme:

\[ Q(x) = T + 2d - \min \]
\[ \beta = 2d - 2\min \]

Another extreme:

\[ Q(x) = T + \min \]
\[ \beta = 0 \]
Problem #2: slave drift

\[ 2d(1 - \rho) \leq 2D \leq 2d(1 + \rho) \]
Problem #3: master drift

During the master’s clock drifts

Even if you know $\beta$, there is still some uncertainty!
Cristian's algorithm

\[ \text{time} = \min + \alpha + 2d \]

\[ \text{time} = \min + \beta \]

\[ t = x \]

\[ \alpha, \beta \geq 0 \]

\[ \text{slave} \]

\[ P(t) \]

\[ t \]

\[ \text{master} \]

\[ Q(t) \]

\[ T \]

\[ Q(x) = ? \]

\[ \text{"time=?"} \]

\[ \text{"time= T"} \]
Cristian’s algorithm

Naive estimation: $Q(x) = T + (\text{min} + \beta)$

(take master’s drift into account)

$Q(x) \in [T + (\text{min} + \beta)(1 - \rho), T + (\text{min} + \beta)(1 + \rho)]$

$0 \leq \beta \leq 2d - 2\text{min}$ (take delay into account)

$Q(x) \in [T + (\text{min} + 0)(1 - \rho), T + (\text{min} + 2d - 2\text{min})(1 + \rho)]$

$= [T + (\text{min})(1 - \rho), T + (2d - \text{min})(1 + \rho)]$

$2d \leq 2D(1 + \rho)$ (take slave’s drift into account)

$Q(x) \in [T + (\text{min})(1 - \rho), T + 2D(1 + \rho) - \text{min}(1 + \rho)]$

$= [T + (\text{min})(1 - \rho), T + 2D(1 + 2\rho) - \text{min}(1 + \rho)]$
Slave's estimation and precision

Slave's best guess: \[ Q(x) = T + D(1 + 2\rho) - \min \cdot \rho \]

Maximum error: \[ e = D(1 + 2\rho) - \min \]

You can keep trying, until you achieve the required precision.
Adjusting the clock

After synchronizing:

- If slave simply sets $P(x) = Q(x)$, it could create time discontinuities.
Adjusting the clock

Logical clock \( C(t) = H(t) + A(t) \)

Hardware clock  Adjustment function