Implementing strong clocks
(the hard way)

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

\[ VC(e_i)[j] = \text{number of events executed by process } j \text{ that causally precede } e_i \]
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: $p \rightarrow q \Leftrightarrow VC(p) < VC(q)$
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:

$D[j] = VC(m)[j] - 1$

$D[k] \geq VC(m)[k], \forall k \neq j$
Causal Delivery

\[
D
\]

\[
D[j] = VC(m)[j] - 1
\]

\[
D[k] \geq VC(m)[k], \forall k \neq j
\]
ADMINISTRIVIA

- Still missing a few pictures
  - Please send them today

- At capacity
Clock synchronization

What time is it?
Roman generals v2.0

Attack at midnight!

Chaaaaaaarge!

12:00am

ZZZZZZ

11:30pm
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

\[ t = x \]

\[ Q(x) = T + \text{min} \rightarrow \text{Perfect synchronization!} \]
Problem #1: message delay

\[ P(t) \quad Q(t) \quad t \]  
\[ min + \alpha \quad min + \beta \]  
\[ t \quad T \]  
\[ Q(x) = T + 2d - \min \]  
\[ \beta = 2d - 2\min \]  

one extreme

\[ t \quad P(t) \quad Q(t) \quad T \]  
\[ min \quad 2d - min \]  
\[ Q(x) = T + 2d - \min \]  

another extreme

\[ t \quad P(t) \quad Q(t) \quad T \]  
\[ 2d - \min \quad min \]  
\[ Q(x) = T + \min \]  
\[ \beta = 0 \]
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

\[ \text{time} = \min + \alpha, \quad \min + \beta, \quad t = x \]

\[ 2d \quad 2D \]

slave \[ P(t) \]

master \[ Q(t) \]

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

\[ T \]

\[ Q(x) = ? \]
Cristian’s algorithm

Naive estimation: \( Q(x) = T + (\min + \beta) \)

(take master’s drift into account)

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

0 \( \leq \beta \leq 2d - 2\min \) (take delay into account)

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

= \( [T + (\min)(1 - \rho), T + (2d - \min)(1 + \rho)] \)

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

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

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

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

Maximum error: \( e = D(1 + 2\rho) - \text{min} \)

You can keep trying, until you achieve the required precision
(if that precision is reasonable)
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 \[ H(t) \]

Adjustment function \[ A(t) \]