Causal Delivery in Synchronous Systems

We use the upper bound \( \Delta \) on message delivery time

\[ \text{DR1: } \text{At time } t, p_0 \text{ delivers all messages it received with timestamp up to } t - \Delta \text{ in increasing timestamp order.} \]

Causal Delivery with Lamport Clocks

\[ \text{DR1.1: } \text{Deliver all received messages in increasing (logical clock) timestamp order.} \]

\[ p_0 \]
**Causal Delivery with Lamport Clocks**

**DR1.1:** Deliver all received messages in increasing (logical clock) timestamp order.

\[ p_0 \quad \text{Should } p_0 \text{ deliver?} \]

Problem: Lamport Clocks don’t provide gap detection

Given two events \( e \) and \( e' \) and their clock values \( LC(e) \) and \( LC(e') \) —where \( LC(e) < LC(e') \) —determine whether some event \( e'' \) exists s.t.

\[ LC(e) < LC(e'') < LC(e') \]

**Stability**

**DR2:** Deliver all received stable messages in increasing (logical clock) timestamp order.

A message \( m \) received by \( p \) is stable at \( p \) if \( p \) will never receive a future message \( m' \) s.t.

\[ TS(m') < TS(m) \]

**Implementing Stability**

- Real-time clocks
- \( \square \) wait for \( \Delta \) time units
Implementing Stability

- Real-time clocks
  - wait for $\Delta$ time units

- Lamport clocks
  - wait on each channel for $m$ s.t. $TS(m) > LC(e)$

- Design better clocks!

Clocks and STRONG Clocks

- Lamport clocks implement the clock condition:
  
  \[ e \rightarrow e' \Rightarrow LC(e) < LC(e') \]

- We want new clocks that implement the strong clock condition:
  
  \[ e \rightarrow e' \equiv SC(e) < SC(e') \]

Causal Histories

- The causal history of an event $e$ in $(H, \rightarrow)$ is the set
  
  \[ \theta(e) = \{ e' \in H \mid e' \rightarrow e \} \cup \{ e \} \]

Causal Histories

- The causal history of an event $e$ in $(H, \rightarrow)$ is the set
  
  \[ \theta(e) = \{ e' \in H \mid e' \rightarrow e \} \cup \{ e \} \]
**Causal Histories**

The causal history of an event \( e \) in \( (H, \rightarrow) \) is the set

\[
\theta(e) = \{e' \in H \mid e' \rightarrow e\} \cup \{e\}
\]

**How to build \( \theta(e) \)**

Each process \( p_i \):

- initializes \( \theta : \theta := \emptyset \)
- if \( e_i^k \) is an internal or send event, then
  \[
  \theta(e_i^k) := \{e_i^k\} \cup \theta(e_i^{k-1})
  \]
- if \( e_i^k \) is a receive event for message \( m \), then
  \[
  \theta(e_i^k) := \{e_i^k\} \cup \theta(e_i^{k-1}) \cup \theta(\text{send}(m))
  \]

**Pruning causal histories**

- Prune segments of history that are known to all processes (Peterson, Buchholz and Schlichting)
- Use a more clever way to encode \( \theta(e) \)

**Vector Clocks**

- Consider \( \theta_i(e) \), the projection of \( \theta(e) \) on \( p_i \)
- \( \theta_i(e) \) is a prefix of \( h_i^k : \theta_i(e) = h_i^k \) - it can be encoded using \( k_i \)
- \( \theta(e) = \theta_1(e) \cup \theta_2(e) \cup \ldots \cup \theta_n(e) \) can be encoded using \( k_1, k_2, \ldots, k_n \)

Represent \( \theta \) using an \( n \)-vector \( VC \) such that

\[
VC(e)[i] = k \iff \theta_i(e) = h_i^k
\]
Update rules

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

Message $m$ is timestamped with $TS(m) = VC(send(m))$

$VC(e_i) := \max(VC, TS(m))$
$VC(e_i)[i] := VC[i] + 1$

Example

Operational interpretation

$VC(e_i)[i] = \max(VC, TS(m))$
$VC(e_i)[i] := VC[i] + 1$

Operational interpretation

$VC(e_i)[i] = \text{no. of events executed by } p_i \text{ up to and including } e_i$
$VC(e_i)[j] =$
Operational interpretation

Operational interpretation

VC properties: event ordering

Given two vectors \( V \) and \( V' \), less than is defined as: \( V < V' \equiv (V 
\neq V') \land (\forall k: 1 \leq k \leq n : V[k] \leq V'[k]) \)

\[ \text{Strong Clock Condition: } e \rightarrow e' \equiv VC(e) < VC(e') \]

\[ \text{Simple Strong Clock Condition:} \]
Given \( e_i \) of \( p_i \) and \( e_j \) of \( p_j \), where \( i \neq j \)
\( e_i \rightarrow e_j \equiv VC(e_i)[i] \leq VC(e_j)[i] \)

\[ \text{Concurrency} \]
Given \( e_i \) of \( p_i \) and \( e_j \) of \( p_j \), where \( i \neq j \)
\( e_i \parallel e_j \equiv (VC(e_i)[i] > VC(e_j)[i]) \land (VC(e_j)[j] > VC(e_i)[j]) \)

VC properties: consistency

\[ \text{Pairwise Inconsistency} \]
Events \( e_i \) of \( p_i \) and \( e_j \) of \( p_j \) \((i \neq j)\) are pairwise inconsistent (i.e. can't be on the frontier of the same consistent cut) if and only if
\( (VC(e_i)[i] < VC(e_j)[i]) \lor (VC(e_j)[j] < VC(e_i)[j]) \)

\[ \text{Consistent Cut} \]
A cut defined by \( (c_1, \ldots, c_n) \) is consistent if and only if
\( \forall i, j : 1 \leq i \leq n, 1 \leq j \leq n : (VC(c_i)[i] \geq VC(c_j)[j]) \)