Logical clocks

Assign sequence numbers to messages
- All cooperating processes can agree on order of events
- vs. physical clocks: time of day

Assume no central time source
- Each system maintains its own local clock
- No total ordering of events
- No concept of happened-when

Frobenius "happened-before" notation

\[ a \rightarrow b \] event \( a \) happened before event \( b \)

e.g.: \( a \): message being sent, \( b \): message receipt

Transitive:
if \( a \rightarrow b \) and \( b \rightarrow c \) then \( a \rightarrow c \)

Event counting example

- Three systems: \( P_0, P_1, P_2 \)
- Events \( a, b, c, \ldots \)
- Local event counter on each system
- Systems occasionally communicate
**Event counting example**

- P1: a 1, b 2, c 3, d 4, e 5, f 6
- P2: i 1, j 2, h 3, g 4, e 5
- P3: k 1

Bad ordering:
- e → h
- f → k

**Lamport’s algorithm**

- Each message carries a timestamp of the sender’s clock
- When a message arrives:
  - if receiver’s clock < message timestamp
    set system clock to (message timestamp + 1)
  - else do nothing
- Clock must be advanced between any two events in the same process

**Summary**

- Algorithm needs monotonically increasing software counter
- Incremented at least when events that need to be timestamped occur
- Each event has a Lamport timestamp attached to it
- For any two events, where a → b: L(a) < L(b)

**Problem: Identical timestamps**

- a → b, b → c, ...: local events sequenced
- i → c, f → d, d → g, ...: Lamport imposes a send→receive relationship

Concurrent events (e.g., a & i) may have the same timestamp ... or not
Unique timestamps (total ordering)

We can force each timestamp to be unique
- Define global logical timestamp \((T_i, i)\)
  - \(T_i\) represents local Lamport timestamp
  - \(i\) represents process number (globally unique)
- Compare timestamps:
  \((T_i, i) < (T_j, j)\)
  if and only if
  \(T_i < T_j\) or
  \(T_i = T_j\) and \(i < j\)

Does not relate to event ordering

Problem: Detecting causal relations

If \(L(e) < L(e')\)
- Cannot conclude that \(e \rightarrow e'\)

Looking at Lamport timestamps
- Cannot conclude which events are causally related

Solution: use a vector clock

Comparing vector timestamps

Define
\[ V = V' \text{ iff } V[i] = V'[i] \text{ for } i = 1 \ldots N \]
\[ V \preceq V' \text{ iff } V[i] \preceq V'[i] \text{ for } i = 1 \ldots N \]

For any two events \(e, e'\)
- if \(e \rightarrow e'\) then \(V(e) < V(e')\)
  - Just like Lamport’s algorithm
  - if \(V(e) < V(e')\) then \(e \rightarrow e'\)

Two events are concurrent if neither
\(V(e) \preceq V(e')\) nor \(V(e') \preceq V(e)\)
Summary: Logical Clocks & Partial Ordering

- Causality
  - If $a \rightarrow b$ then event $a$ can affect event $b$
- Concurrency
  - If neither $a \rightarrow b$ nor $b \rightarrow a$ then one event cannot affect the other
- Partial Ordering
  - Causal events are sequenced
- Total Ordering
  - All events are sequenced