Logical time and clocks (11.4)

- Can't synchronize physical clocks perfectly
- Absolute time might not be necessary, just need ordering of events
- Logical clocks: Lamport, 1978
- Happened-before relationship among events
- Potential causal ordering

Happen-before relation

- \( e \rightarrow e' \)
  - Two events occur in the same process
  - A message is sent between two processes

Happened-before relation:

- HB1: if \( e \rightarrow e' \) in process \( i \), then \( e \rightarrow e' \)
- HB2: for any message \( m \), send\((m) \rightarrow \) receive\((m) \)
- HB3: if \( e \rightarrow e' \) and \( e' \rightarrow e'' \), then \( e \rightarrow e'' \)

Logical clocks (Lamport)

- Each process \( i \) keeps a local clock \( L_i \)
- Updating logical clocks:
  - LC1: \( L_i := L_i + 1 \) for each event in process \( i \)
  - LC2:
    - When a process \( i \) sends a message \( m \)
      - It piggybacks on \( m \) the value of \( t = L_i \)
    - Upon receiving \((m, t)\), process \( j \)
      - Computes \( L_j := \max(L_j, t) \) and then
      - Applies LC1 before timestamping the event receive\((m) \)
  - If \( e \rightarrow e' \), then \( L(e) < L(e') \)
  - But \( L(e) < L(e') \) doesn't imply \( e \rightarrow e' \). Why? (Fig. 11.6)

Events occurring at three processes

Happen-before vs. causality

- \( e \parallel e' \) if the two events aren't in a particular order (concurrent)
- Potential causality: \( e \rightarrow e' \) doesn't mean that \( e \) causes \( e' \)
- Naturally, if \( e \) causes \( e' \), \( e \rightarrow e' \)
Lamport timestamps (Fig 11.6)

- Lamport timestamps: \( L(e) < L(e') \) doesn't imply \( e \to e' \)
- Each process keeps its own vector clock \( V_i \)
- Piggyback timestamps on messages
- Updating vector clocks:
  - VC1: Initially, \( V[i] := 0 \) for \( p_i, j=1..N \) (\( N \) processes)
  - VC2: before \( p_i \) timestamps an event, \( V[i] := V[i]+1 \)
  - VC3: \( p_i \) piggybacks \( t = V_i \) on every message it sends
  - VC4: when \( p_j \) receives a timestamp \( t \), it sets \( V[k] := \max(V[k], t[k]) \) for \( k=1..N \) (merge operation)

Totally ordered logical clocks

- Different processes, same Lamport time
- Time: \( (T_i, i) \)
- \( (T_i, i) < (T_j, j) \) iff
  - \( T_i < T_j \) or
  - \( T_i = T_j \) and \( i < j \)

Vector clocks

- At \( p_i \)
  - \( V[i] \) is the number of events \( p_i \) timestamped locally
  - \( V[j] \) is the number of events that have occurred at \( p_j \) (that has potentially affected \( p_i \))
- Could more events than \( V[j] \) have occurred at \( p_j \)?
Vector timestamps

- if $e \rightarrow e'$, then $V(e) < V(e')$
- if $V(e) < V(e')$, then $e \rightarrow e'$. (Exercise 11.13)

Figure 11.7

- neither $V(c) < V(e)$ nor $V(c) > V(e)$
- $c \parallel e$

Disadvantage compared to Lamport timestamps?