Lamport Clocks
Lamport Clocks

Framework for *reasoning* about event ordering

Assign timestamps to events
- Globally valid
- Respects causality
- Using only local information
- No physical clock

What does it mean for *a* to happen before *b*?
Happens-before

1. Happens earlier at same location
2. Transmission before receipt
3. Transitivity
Example

A send M

B recv M

C send M’

D recv M’

E
Logical clock implementation

Keep a local clock $T$
Increment $T$ whenever an event happens
Send clock value on all messages as $T_m$
On message receipt: $T = \max(T, T_m) + 1$
Example

S1

A (T = ?)

send M (T_m = ?)

B (T = ?)

recv M (T = ?)

S2

send M' (T_m = ?)

recv M (T = ?)

C (T = ?)

S3

D (T = ?)

E (T = ?)

recv M' (T = ?)
Example

A (T = 1)

B (T = ?)

send M (T_m = ?)

recv M (T = ?)

C (T = ?)

send M' (T_m = ?)

recv M' (T = ?)

D (T = ?)

E (T = ?)

recv M' (T = ?)

D (T = ?)
Example

S1

A (T = 1)

send M (T_m = 2)

B (T = ?)

recv M (T = ?)

S2

C (T = ?)

send M' (T_m = ?)

recv M (T = ?)

S3

D (T = ?)

E (T = ?)

recv M' (T = ?)
Example

S1
A (T = 1)
send M (T_m = 2)
B (T = 3)

S2
send M' (T_m = ?)
recv M (T = 3)
C (T = ?)

S3
E (T = ?)
recv M' (T = ?)
D (T = ?)
Example

A (T = 1)

B (T = 3)

send M (T_m = 2)

recv M (T = 3)

C (T = 4)

send M' (T_m = ?)

recv M' (T = ?)

D (T = ?)

E (T = ?)
Example

S1

A (T = 1)

send M (T_m = 2)

B (T = 3)

S2

C (T = 4)

recv M (T = 3)

send M’ (T_m = 5)

S3

E (T = ?)

recv M’ (T = ?)

D (T = ?)
Example

A (T = 1)

B (T = 3)
send M (T_m = 2)

C (T = 4)
recv M (T = 3)
send M' (T_m = 5)

D (T = 1)

E (T = ?)
recv M' (T = ?)
Example

A (T = 1)

send M (T_m = 2)

B (T = 3)

recv M (T = 3)

C (T = 4)

send M' (T_m = 5)

recv M' (T = 6)

D (T = 1)

E (T = ?)
Example

- **S1**
  - A \((T = 1)\)
  - send \(M\) \((T_m = 2)\)
  - B \((T = 3)\)

- **S2**
  - recv \(M\) \((T = 3)\)
  - send \(M'\) \((T_m = 5)\)
  - C \((T = 4)\)
  - recv \(M\) \((T = 3)\)

- **S3**
  - E \((T = 7)\)
  - recv \(M'\) \((T = 6)\)
  - D \((T = 1)\)
Goal of Lamport clocks

\[ \text{happens-before}(A, B) \rightarrow T(A) < T(B) \]

Does \( T(A) < T(B) \rightarrow \text{happens-before}(A, B) \)?
Mutual exclusion

Use clocks to implement a lock
  - Using state machine replication

Goals:
  - Only one process has the lock at a time
  - Requesting processes eventually acquire the lock

Assumptions:
  - In-order point-to-point message delivery
  - No failures, all messages delivered
Mutual exclusion implementation

Each message carries a timestamp $T_m$ (and a seq #)

Three message types:
- request (broadcast)
- release (broadcast)
- acknowledge (on receipt)

Each node’s state:
- A queue of request messages, ordered by $T_m$
- The latest message it has received from each node
Mutual exclusion implementation

On receiving a request:
- Record message timestamp
- Add request to queue

On receiving a release:
- Record message timestamp
- Remove corresponding request from queue

On receiving an acknowledge:
- Record message timestamp
Mutual exclusion implementation

To acquire the lock:

- Send *request* to everyone, including self
- The lock is acquired when:
  - My request is at the head of my queue, and
  - I’ve received higher-timestamped messages from everyone
- So my request must be the earliest
Timestamp: 1
Queue: [S1@0; S2@1]
S1_{max}: 0
S3_{max}: 0

Timestamp: 2
Queue: [S1@0; S2@1]
S1_{max}: 0
S2_{max}: 1
S3_{max}: 0

Timestamp: 2
Queue: [S1@0; S2@1]
S1_{max}: 0
S2_{max}: 1
S3_{max}: 0
Timestamp: 1
Queue: [S1@0; S2@1]
S1_{max}: 0
S3_{max}: 0

Timestamp: 3
Queue: [S1@0; S2@1]
S2_{max}: 1
S3_{max}: 0

Timestamp: 3
Queue: [S1@0; S2@1]
S1_{max}: 0
S2_{max}: 1
Timestamp: 6
Queue: [S2@1]
S1$_{max}$: 4
S3$_{max}$: 3

Timestamp: 4
Queue: [S2@1]
S2$_{max}$: 1
S3$_{max}$: 0

Timestamp: 6
Queue: [S2@1]
S1$_{max}$: 4
S2$_{max}$: 1
Questions

• What happens if you don’t have in-order delivery?
• What happens if you eliminate the ack for the request?
• What happens when nodes fail?
Generic State Machine Replication (SMR)

In mutual exclusion:

• State: queue of processes who want the lock
• Commands: $P_i$ \textit{requests}, $P_i$ \textit{releases}

Approach generalizes to other “state machines"

Process a command iff we’ve seen all commands w/ lower timestamp