CSE 124 Discussion
Lamport & Vector Clocks

11/7/17
Quick review of Lamport clocks

The Lamport clock is just a single value

Each actor in the system has its own local version of the Lamport clock

Every action is tagged with the current value of the Lamport clock

There’s three types of actions: event, send, and recv

Actions also affect the actor’s local Lamport clock before the action is tagged

We denote the tagged value of an event as $C(E)$ for the event named $E$
How actions affect Lamport clocks

Event: Increase the local Lamport clock by 1

Send: Increase the local Lamport clock by 1, then send the new local Lamport clock

Recv: Set the local lamport clock to \( \max(\text{local\_clock}, \text{received\_clock}) + 1 \)
Simple Lamport Clock Example

α

β

σ
Simple Lamport Clock Example

\[ \beta \]  
\[ \sigma \]
Simple Lamport Clock Example

α

β

σ

A (1)

B (1)
Simple Lamport Clock Example

\[ \alpha \rightarrow A(1) \rightarrow C(2) \rightarrow B(1) \]

\[ \beta \rightarrow \]

\[ \sigma \rightarrow \]
Simple Lamport Clock Example

\[ \alpha \quad A \quad (1) \quad C \quad (2) \quad \beta \quad (1) \quad \sigma \quad B \quad (1) \quad D \quad (2) \]
Simple Lamport Clock Example

α

B (1)

D (2)

C (2)

E (3)

σ

A (1)

E (3)

β


Simple Lamport Clock Example

- **\(\alpha\)**: A (1) \(\rightarrow\) C (2)
- **\(\beta\)**: B (1) \(\rightarrow\) E (3)
- **\(\sigma\)**: D (2) \(\rightarrow\) F (3)

The diagram illustrates the sequence of events with timestamps (1, 2, 3) associated with each process.
Simple Lamport Clock Example
Simple Lamport Clock Example

A (1)  C (2)  E (3)  B (1)  D (2)  F (3)  G (4)  H (5)
Simple Lamport Clock Example

\[2\]
\[\alpha\]
\[\beta\]
\[\sigma\]

\[A\] (1)
\[B\] (1)
\[C\] (2)
\[D\] (2)
\[E\] (3)
\[F\] (3)
\[G\] (4)
\[H\] (5)
\[I\] (4)
Simple Lamport Clock Example

\[ \alpha \]
\[ \beta \]
\[ \sigma \]
Happens-before relations

In any concurrent system, we need ordering guarantees.

This can be quantified using the “happens-before” relation.

A → B means that “A is guaranteed to occur before B”.

If we know that B happened, then we know A happened as well!

Happens-before is transitive, irreflexive, and antisymmetric.

Transitive: A → B ∧ B → C ⇒ A → C
Irreflexive: A ↛ A
Antisymmetric: A → B ⇒ B ↛ A
Happens-before and Lamport Clocks

If \( A \rightarrow B \), then \( C(A) < C(B) \)

The converse isn’t true:

If \( C(A) < C(B) \), then \textit{maybe} \( A \rightarrow B \)!

Unfortunately, this makes Lamport Clocks not too useful on their own.
Lamport clocks don’t imply happens-before
Lamport clocks don’t imply happens-before

Clearly $C(A) < C(D)$

Is $A \rightarrow D$ true?
Lamport clocks don’t imply happens-before

No, A \nrightarrow D
D could happen before A!
Let’s see how this can cause problems...
Determining order at the server using Lamport clocks
Determining order at the server using Lamport clocks
Determining order at the server using Lamport clocks

Server knows $C(C) < C(I)$

But it doesn’t know $C \rightarrow I$!

If the server went off receive events, it would think $I \rightarrow C$
Quick review of vector clocks

Vector clocks are an array of values with length = number of actors.

Each actor has their own local version of the vector clock.

Each value in the vector clock represents the local Lamport clock of each actor.

This does require each actor to have a unique ID (usually use order of process IDs).

Every action increases the local Lamport clock for the actor that performed it.

Actions are tagged with the current local vector clock.

Same actions: event, send, recv.
How actions affect vector clocks

Event at $P_i$: Increase the local Vector clock’s $i$-th value by 1.

Event at $P_0$: $(1, 0, 0) \rightarrow (2, 0, 0)$

Send: Increase the local vector clocks’ $i$-th value by 1, then send the entire new local vector clock

Recv at $P_j$: Increase the local Vector clock’s $j$-th value by 1, then for every value $k$ in the local vector clock, set $V_k = \max(V_k, R_k)$, where $R$ is the received vector clock, and $V$ is the local vector clock
Comparing vector clocks

$V_1(X) < V_2(Y)$ if and only if:

For all $k$, $V_{1,k}(X) \leq V_{2,k}(Y)$

For some $k$, $V_{1,k}(X) < V_{2,k}(Y)$
Simple vector Clock Example

α

β

σ
Simple vector Clock Example

α

β

σ
Simple vector Clock Example

\[ \alpha \]

\[ \beta \]

\[ \sigma \]
Simple vector Clock Example

α

β

σ

A

1,0,0

C

2,0,0

B

0,0,1

2,0,0

0,0,0

0,0,1
Simple vector Clock Example

\[ \alpha \]

\[ 2,0,0 \]

A

1,0,0

C

2,0,0

\[ \beta \]

\[ 0,0,0 \]

\[ \sigma \]

\[ 0,0,2 \]

B

0,0,1

D

0,0,2
Simple vector Clock Example

- \( \alpha \) with nodes A (1,0,0) and C (2,0,0)
- \( \beta \) with nodes B (0,0,1) and D (0,0,2)
- \( \sigma \) with node E (2,1,0)
Simple vector Clock Example

\[\begin{align*}
\text{α} & : 2,0,0 \\
\beta & : 2,1,0 \\
\sigma & : 0,0,3
\end{align*}\]
Simple vector Clock Example

\[ \alpha \]

\[ \beta \]

\[ \sigma \]
Simple vector Clock Example

\[ \alpha \] 2,0,0

A 1,0,0

C 2,0,0

\[ \beta \] 2,1,0

\[ \sigma \] 0,0,5

B 0,0,1

D 0,0,2

E 2,1,0

F 0,0,3

G 0,0,4

H 0,0,5
Simple vector Clock Example

α

2,0,0

A

1,0,0

C

2,0,0

β

2,2,0

E

2,1,0

I

2,2,0

σ

2,2,0

B

0,0,1

D

0,0,2

F

0,0,3

G

0,0,4

H

0,0,5
Simple vector Clock Example

\[ \alpha = 2,0,0 \]

\[ \beta = 2,2,0 \]

\[ \sigma = 2,2,6 \]
Vector clocks do imply happens-before

\[ V_1(A) < V_2(B) \text{ then } A \rightarrow B \]

\[ V_1(A) < V_2(B) \text{ then } C_1(A) < C_2(B) \]

It can be the case that \( V_1(A) \not< V_2(B) \text{ and } V_2(B) \not< V_1(A) \)

I.e. it is possible that \( A \not\rightarrow B \text{ and } B \not\rightarrow A \! \)

This means the events are *independent* from each other.

Note that if \( A \not\rightarrow B \), that doesn’t mean \( B \rightarrow A \! \)
Independent events

\[ \begin{align*}
\alpha & : 2,0,0 \\
\beta & : 2,2,0 \\
\sigma & : 2,2,6
\end{align*} \]
Independent events

$\sigma$ (2,2,6)

$\beta$ (2,2,0)

$\alpha$ (2,0,0)

$\text{V}_\alpha(A) \nleq \text{V}_\sigma(D)$ and $\text{V}_\sigma(D) \nleq \text{V}_\alpha(A)$

So $A \not\leftrightarrow D$ and $D \not\leftrightarrow A$!
But now servers can tell which event happened first!
Determining order at the server using Vector clocks

2,0,0
\(\alpha\)

2,2,0
\(\beta\)

2,2,6
\(\sigma\)

A: 2,0,0
C: 2,0,0
E: 2,1,0
I: 2,2,0
B: 0,0,1
D: 0,0,2
F: 0,0,3
G: 0,0,4
H: 0,0,5
J: 2,2,6
Determining order at the server using Vector clocks

Server knows $V(\text{C}) < V(\text{I})$

Which does mean $\text{C} \rightarrow \text{I}$!

Now the server knows what happened first.