6.852: Distributed Algorithms
Fall, 2015

Lecture 14, Part 1

Weak Logical Time and Vector Timestamps

Weak Logical Time

- Logical time imposes a total ordering on events, assigning them values from a totally-ordered set T.
- Sometimes we don't need to order all events---it may be enough to order just the ones that are causally dependent.
- Mattern (also Fidge) developed an alternative notion of logical time based on a partial ordering of events, assigning them values from a partially-ordered set P.
- Function ltime from events in α to partially-ordered set P is a weak logical time assignment if:
  1. $ltimes$ are distinct: $ltime(e_1) \neq ltime(e_2)$ if $e_1 \neq e_2$.
  2. $ltimes$ of events at each process are monotonically increasing.
  3. $ltime$(send) < $ltime$(receive) for the same message.
  4. For any t, the number of events e with $ltime(e) < t$ is finite.
- Same as for logical time, but using partial order.

Weak Logical Time

- In fact, Mattern's partially-ordered set P represents causality exactly.
- Logical times of two events are ordered in P if and only if the two events are causally related (related by the causality ordering).
- Might be useful in distributed debugging: A log of local executions with weak logical times could be observed after the fact, used to infer causality relationships among events.

Algorithm for weak logical time

- Based on vector timestamps: vectors of nonnegative integers indexed by processes.
- Algorithm:
  - Each process maintains a local vector clock, called $vclock$.
  - When a non-receive event occurs at process i, it increments its own component of its $vclock$, which is $vclock(i)$, and assigns the new $vclock$ to be the vector timestamp of the event.
  - Whenever process i sends a message, it attaches the vector timestamp of the send event.
  - When i receives a message, it first increases its $vclock$ to the component-wise maximum of its current $vclock$ and the incoming vector timestamp. Then it increments its $vclock(i)$ as before, and assigns the new $vclock$ to the receive event.
- A process' $vclock$ represents the latest known "tick values" for all processes.
- Partially ordered set P:
  - The vector timestamps, ordered based on $\leq$ in all components.
  - $V \leq V'$ if and only if $V(i) \leq V'(i)$ for all i.

Key theorems about vector clocks

- Theorem 1: The vector clock assignment is a weak logical time assignment.
- Lemma 1: If event π causally precedes event π', then the logical times are ordered, in the same order.
- Proof:
  - True for direct causality.
  - Use induction on the number of direct causality relationships.
- Claim this assignment exactly captures causality:
- Lemma 2: If the vector timestamp V of event π is (component-wise) $\leq$ the vector timestamp V' of event $\pi' \neq \pi$, then π causally precedes π'.
- Proof: Prove the contrapositive: Assume π does not causally precede π' and show that V is not $\leq V'$.

Proof of Lemma 2

- Lemma 2: If the vector timestamp V of event π is (component-wise) $\leq$ the vector timestamp V' of event $\pi' \neq \pi$, then π causally precedes π'.
- Proof:
  - Prove the contrapositive: Assume π does not causally precede π' and show that V is not $\leq V'$.
  - Case 1: π and π' are events of the same process i.
    - Then since π does not causally precede π', it must be that π' precedes π in time.
    - Then $V'(i) < V(i)$.
    - So V is not $\leq V'$.
  - Case 2: π is an event of process i and π' an event of another process $j \neq i$.

Proof of Lemma 2

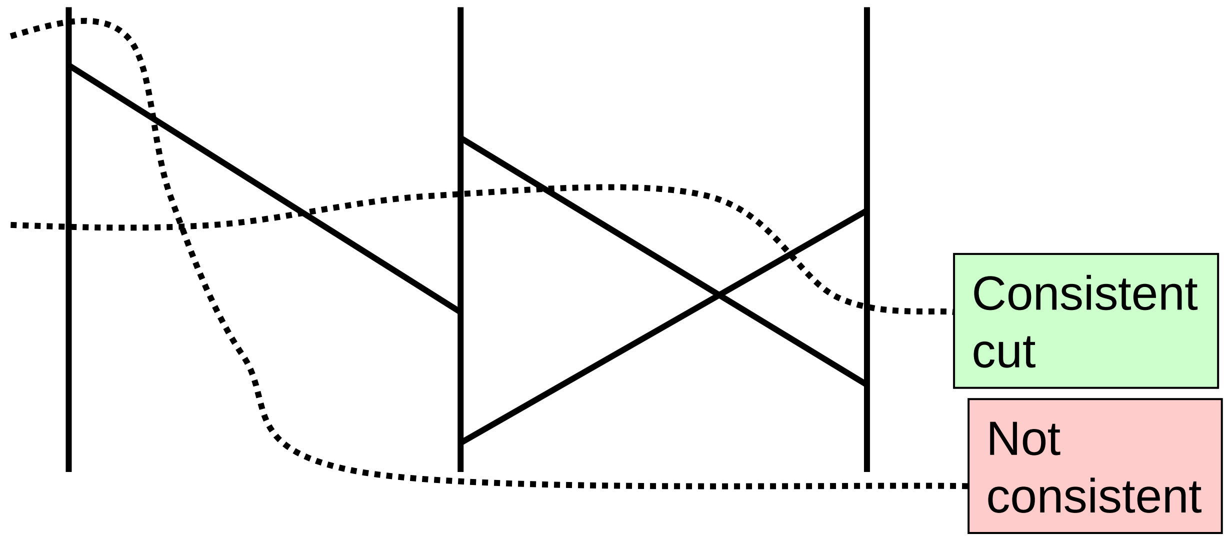
- Lemma 2: If the vector timestamp V of event π is (component-wise) $\leq$ the vector timestamp V' of event $\pi' \neq \pi$, then π causally precedes π'.
- Proof:
  - Prove the contrapositive: Assume π does not causally precede π' and show that V is not $\leq V'$.
  - Case 2: π is an event of process i and π' an event of process $j \neq i$.
    - i increases its $vclock(i)$ for event π, say to value t.
    - Without causality, there is no way for this value t for i to propagate to j before π' occurs.
    - So, when π' occurs at process j, j's $vclock(i) < t$.
    - So V is not $\leq V'$.

Back to Theorem 1

- Theorem 1: The vector clock assignment is a weak logical time assignment.
- Lemma 1: If event π causally precedes event π', then the logical times are ordered, in the same order.
- Lemma 2: If the vector timestamp V of event π is (component-wise) $\leq$ the vector timestamp V' of event $\pi' \neq \pi$, then π causally precedes π'.
- Proof of Theorem 1:
  - The ordering is a partial order.
  - Lemma 1 yields Properties 2 and 3.
  - Lemma 2 yields Property 1 (uniqueness).
  - Property 4 (non-Zeno) LTTR.

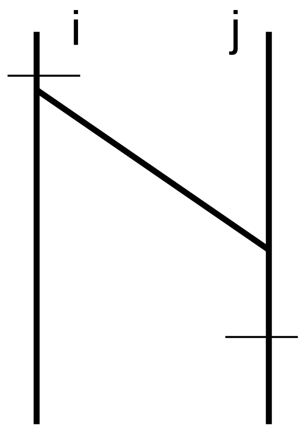
Another important theorem about vector timestamps [Mattern]

- Relates timestamps to consistent cuts of the causality graph.
- Cut: A point between events at each process.
  - Specify a cut by a vector giving the number of preceding steps at each location.
- Consistent cut: "Closed under causality": If event π causally precedes event π' and π' is before the cut, then so is π.
- Example:



The theorem

- Consider any particular cut.
- Let V_i be the vector clock of process i exactly at i's cut-point.
- Then $V = \max(V_1, V_2, \ldots, V_n)$ gives the maximum information obtainable by combining everyone knowledge at the cut-points.
  - Component-wise max.
- Theorem 2: The cut is consistent iff, for every i, $V(i) = V_i(i)$.
- That is, the maximum information about i that anyone knows at its cut point is the same as what i knows about itself at its cut point.
- "No one else knows more about i than i itself knows."
- Rules out j receiving a message before its cut point that i sent after its cut point; in that case, j would have more information about i than i had about itself.



The theorem

- Let V_i be the vector clock of process i exactly at i's cut-point.
- $V = \max(V_1, V_2, \ldots, V_n)$.
- Theorem 2: The cut is consistent iff, for every i, $V(i) = V_i(i)$.
- Stated slightly differently:
- Theorem 2: The cut is consistent iff, for every i and j, $V_j(i) \leq V_i(i)$.
- Proof: LTTR (see Mattern's paper).

- Q: What is this good for?

Application: Debugging

- Theorem 2: The cut is consistent iff, for every i and j, $V_j(i) \leq V_i(i)$.
- Example: Debugging
  - Each node keeps a log of its local execution, with vector timestamps for all events.
  - Collect information, find a cut for which $V_j(i) \leq V_i(i)$ for every i and j. (Mattern gives an algorithm to do this.)
  - By Theorem 2, this is a consistent cut.
  - Such a cut yields:
    - States for all processes at the cut, and
    - Information about messages sent before the cut and not received until after the cut, i.e., messages "in transit" at the cut.
  - Put this together, get a "consistent" global state (we will study this next).
  - Use this to check correctness properties for the execution, e.g., invariants.