Distributed Software Development

Synchronization

Chris Brooks

Department of Computer Science
University of San Francisco

4-0: Time in Distributed Systems

- In systems with multiple CPUs, the clocks are unlikely to have the exact same time.
  - Variations in manufacturing cause clock skew.
- Insight: often, it doesn't matter exactly what time an operation happens, but what order events occur in.
- (exception: hard real-time systems)

4-1: Logical clocks

- A logical clock is just a counter(or set of counters)
- What's important is that all processes in the system can use it to produce a consistent ordering.
- This lets all processes in the system construct a global view of the system state.

4-2: Global state

- Many interesting problems in distributed computing can be expressed in terms of determining whether some property p is satisfied.
  - Consensus
  - Deadlock
  - Termination
  - Load balancing
- The general version of this problem is called the *Global Predicate Evaluation Problem*

4-3: Synchronization

- For example, to detect deadlock, we construct a wait-for graph.
  - Edge from process p_i to p_j if p_i is waiting on a message from p_j.
- If this graph has a cycle, we have deadlock.
- How do we do this while the computation is running?
- We can't pause the computation and collect all the information in one place.

4-4: Distributed Computation

- We define a distributed computation to consist of a set of processes $p_1, p_2, \ldots, p_n$.
- Unidirectional channels exist between each process for passing messages.
- We assume these channels are reliable, but not necessarily FIFO.
  - Messages may arrive out of order.
- Assume the communication channels are asynchronous
  - No shared clock
  - No bound on message delay

4-5: Events and time

- Most of the time, we don't necessarily care about the exact time when each event happens.
- Instead, we care about the order in which events happen on distributed machines.
- If we do care about time, then the problem becomes one of synchronizing about the global value of a clock.

4-6: Distributed Computation

- A process p_i consists of a series of events $e_i^1, e_i^2, ...$
- There are three types of events:
  - Local events - no communication with other processes
  - Send events
  - Receive events
- a *local history* is a sequence of events $e_i^1, e_i^2, ...$ such that order is preserved.

4-7: Distributed Computation

- The *initial prefix* of a history containing the first k events is denoted: $h_i^k = e_i^1, e_i^2, ..., e_i^k$
- $h_i^0 = <>$
- The *Global history* of a computation is the union of all local histories.
  - $h_1 \cup h_2 \cup ... \cup h_n$
- Notice that this doesn't say anything about order of events between processes.
- Since an asynchronous system implies that there is no global time frame between events, we need some other way to order events on different processes.

4-8: Cause and Effect

- Cause and effect can be used to produce a partial ordering.
- Local events are ordered by identifier.
- Send and receive events are ordered.
  - If p_1 sends a message m_1 to p_2, $send(m_1)$ must occur before $receive(m_1)$.
  - Assume that messages are uniquely identified.
- If two events do not influence each other, even indirectly, we won't worry about their order.

4-9: Happens before

- The *happens before* relation is denoted $\rightarrow$.
- Happens before is defined:
  - If e_i^k, e_i^l and $k < l$, then $e_i^k \rightarrow e_i^l$
  - (sequentially ordered events in the same process)
  - If $e_i = send(m)$ and $e_j = receive(m)$, then $e_i \rightarrow e_j$
  - (send must come before receive)
  - If $e \rightarrow e'$ and $e' \rightarrow e''$, then $e \rightarrow e''$
  - (transitivity)
- If $e \not\rightarrow e'$ and $e' \not\rightarrow e$, then we say that e and e' are concurrent. $(e \| e')$
- These events are unrelated, and could occur in either order.

4-10: Happens before

- Happens before provides a partial ordering over the global history. $(H, \rightarrow)$
- We call this a distributed computation.
- A distributed computation can be represented with a space-time diagram.

4-11: ***Space-time diagram***