Lecture #2.

A Model of Distributed Computations

Distributed System Model:
- a set of processes connected by a communication network.

System Model:
- provides the facility to exchange information among processes.
- finite but unpredictable communication delay.
- no common global memory and communicate solely by passing messages.
- no physical global clock.
- messages may be delivered out-of-order, may be lost, garbled, or duplicated.
- processors may fail.
- communication links may go down.

=> System can be modeled as a graph
vertices: processes
edges: links
A distributed program is composed of a set of n asynchronous processes labeled \( P_1, P_2, \ldots, P_n \).

WLOG: assume each \( P_i \) on a different process.

\( C_{ij} \): communication channel between \( P_i \) and \( P_j \).

\( m_{ij} \): denotes a message sent from \( P_i \) to \( P_j \).

Global state:

\[
\{\text{States of } P_i\} \cup \{\text{States of } C_{ij}\} \\
\downarrow \\
\text{local memory} \quad \text{messages in transit}
\]
A model of distributed executions.
- the execution of a process consists of a sequential execution of its actions.
- Actions are atomic and are of three types:
  1. Internal events
  2. Message send events
  3. Message receive events.

Event $e_i^x$: $x^i$ event at $P_i$.

msg $m$: send($m$)
  recv($m$)

Events cause:

- state transition of process
- state transition of channel

$\Rightarrow$ affects the global state.

Internal event: only affect the process.

Send event: affects the sender and the channel.

Receive event: affects the receiver and the channel.
events at $P_i$: linear sequence.

$e_i^1, e_i^2, \ldots, e_i^x, e_i^{x+1}, \ldots$

$H_i = (b_i, \rightarrow_i)$

ordering relation

$e_i^x \rightarrow e_i^{x+1}$

$\rightarrow_i$: causal ordering or dependence.

send and receive events capture the flow of information between processes

$\rightarrow_{msg}$: causal dependency due to messages

For every $m$:

$send(m) \rightarrow_{msg} recv(m)$

A natural way to visualize the evolution of a distributed computation is to use space-time diagrams.
Causal Precedence Relation

\[ H = \bigcup_i h_i \]

Define a binary relation

\[ (H, \rightarrow) \]
∀e_i, ∀e_j ∈ H  e_i → e_j

iff

1. e_i → e_j (i.e., i = j ∧ x < y), or
2. e_i →_{msg} e_j , or
3. ∃e_k st. e_i → e_k ∧ e_k → e_j

Called the happens-before relation.

Concurrent events:

   e_i and e_j are concurrent e_i \parallel e_j

iff

   e_i \rightarrow e_j ∧ e_j \rightarrow e_i

For any two events e_i + e_j,

either

   e_i \rightarrow e_j

or

   e_j \rightarrow e_i

or e_i \parallel e_j
Models of Communication NWS.

**FIFO**: channel is a queue

**non-FIFO**: channel is a set

Causal Order: channels satisfy happen-before

\[ m_{ij} \rightarrow m_{kj} \]

if \( \text{send}(m_{ij}) \rightarrow \text{send}(m_{kj}) \)

then \( \text{recv}(m_{ij}) \rightarrow \text{recv}(m_{kj}) \).

causally related messages destined to the same destination are delivered in an order consistent with the causality:

\[ \text{CO} \subset \text{FIFO} \subset \text{NON-FIFO}. \]

CO considerably simplifies the design of distributed algorithms.

\[ \text{e.g. update to replica} \]
Global state of a distributed system.

\[ \text{state of } P_i^j U \text{ state of } C_i^j. \]

\[ L_S^{x_i} : \text{state of } P_i \text{ after } e_i^x, \text{ (initial state: } L_S^{x_i}) \]

\[ SC_{ij}^{x,y} = \{ m_{ij} | \text{send}(m_{ij}) < L_S^{x_i} \land \text{recv}(m_{ij}) \leq L_S^{y_i} \} \]

denotes all messages that \( P_i \) sent up to event \( e_i^x \) and \( P_j \) has not received until \( e_j^y \).

\[ GS = \{ U_i L_S^{x_i}, U_{j,k} SC_{jk}^{y,j,z} \} \]

Meaningful \( GS \): all states of all components must be recorded at a single instant.

\[ \downarrow \]

This is generally not possible.

So what can go wrong? Construct an example.

\[ P_1 \]

\[ \Rightarrow m \]

\[ P_2 \]

\[ \Rightarrow P_2 \text{ says } m \text{ is recorded} \]

but \( P_2 \) has no memory of \( m \) being sent.
$\Rightarrow$ when a snapshot is taken

not all snapshots be consistent.

$GS$ is consistent if

$$\forall m_{ij} : s_{\text{end}}(m_{ij}) \leq LS_x \Rightarrow m_{ij} \notin S_{ij}$$

$\land$ recr$(m_{ij}) \not\subseteq LS_y$

CUTS