Models of distributed computation; clocks.
A Model of a Distributed System

A distributed system is a fixed set of agents (processes) and a fixed set of directed channels. A channel is directed from one agent to another.

A distributed system is represented by a labeled directed graph with a labeled edge representing a channel, and a vertex representing an agent.
Agents and Channels

- Agent: a message-passing automaton

- State transitions:
  - Change state without sending or receiving a message
  - Change state and send a message on a channel
  - Change state and receive a message on a channel.
  - *(We don’t consider transitions in which the state is changed and multiple messages are sent and/or received. Why?)*

- Channel state: sequence of messages. Initially each channel is empty.
Channel Properties: Safety

- Safety: Channels are first-in-first-out, i.e., messages delivered in the order sent.

Let \( c.\text{sent} \) and \( c.\text{received} \) be the sequence of messages sent and received (respectively) on a channel. Then:

**always**: ?
Channel Properties: Safety

Safety: Channels are first-in-first-out, i.e., messages delivered in the order sent.

Let c.sent and c.received be the sequence of messages sent and received (respectively) on a channel. Then:

\textbf{always}: ?

c.received is an initial prefix of c.sent

e.g. c.sent = [10, 20, 30, 40] and c.received = [10, 20]

\[ \square \ c.\text{received} \sqsubseteq c.\text{sent} \]

symbol for initial prefix
Channel Properties: Progress

If $\textit{seq} \sqsubseteq \textit{c.sent}$ then ?
Channel Properties: Progress

If \( \text{seq} \sqsubseteq \text{c.sent} \) then ?

\[(\text{seq} \sqsubseteq \text{c.sent}) \rightsquigarrow (\text{seq} \sqsubseteq \text{c.received})\]
Why message-passing can be safer:

A crucial property of message-passing systems which makes message-passing easier to use than shared variables: *Once a message is sent, it cannot be deleted.* So:

\[
\begin{align*}
\text{stable } (\text{seq} \sqsubseteq \text{c.received}) \\
\text{stable } (\text{seq} \sqsubseteq \text{c.sent}) \\
(\text{seq} \sqsubseteq \text{c.sent}) \leadsto \Box(\text{seq} \sqsubseteq \text{c.received})
\end{align*}
\]
The system state (also called the global state) is a tuple with an element of the tuple for the state of each agent and an element of the tuple for the state of each channel.

Example: global state with agents A, B and channels c, d is a 4-tuple \((s_A, s_B, s_c, s_d)\)
Graph must be acyclic. Why?
Process Time Line

- Send event
- Receive event
- Internal event
System Time Lines: Process States

Global State at time $T$: is a tuple of states of processes and channels at time $T$

State of process R at time $T$

State of process Q at time $T$
System Time Lines: Channel States

State of channel from R to P at time T (assume only one such channel)

State of channel from P to Q at time T (assume only one such channel)
The state of channel \( c \) at time \( T \) is a queue \([m_1, m_2, m_3]\).
Logical Clock: Total order consistent with partial order

- Assign a number to each point in the time-line graph so that if there is a path from point j to point k, then clock[k] > clock[j]

\[ \text{path}(j \rightarrow k) \text{ IMPLIES } (\text{clock}[k] > \text{clock}[j]) \]

- Necessary and sufficient condition:
  \[ \text{edge}(j,k) \text{ IMPLIES } (\text{clock}[k] > \text{clock}[j]) \]

- How can we establish this formula in a distributed way?
System Time Lines and Logical Clocks

What are distributed ways to update logical clocks for each process?
What are distributed ways to update logical clocks for each process?

Timestamp a message with the local clock time of the sender.

Receiver ensures its local clock is greater than the timestamp.
Vector clocks:

path(j -> k) \equiv (\text{clock}[k] > \text{clock}[j])

What to do to ensure path(j -> k) if and only if clock[k] > clock[j]?

Timestamp a message with the local clock time of the sender.

Receiver ensures its local clock is greater than the timestamp.
Vector clocks:
\[ \text{path}(j \rightarrow k) \equiv (\text{clock}[k] > \text{clock}[j]) \]

Each clock is a vector: 
\[ [T_p, T_q, T_r] \]

Timestamp a message with the local clock time of the sender.

Receiver ensures its local clock is greater than the timestamp.
Next class:

- First of many distributed algorithms: Gossip or Diffusion Computation Algorithms