Message ordering and group communication

Message ordering paradigms

The order of delivery of messages in a distributed system is an important aspect of system executions because it determines the messaging behavior that can be expected by the distributed program. Distributed program logic greatly depends on this order of delivery.

Several orderings on messages have been defined:

(i) non-FIFO

(ii) FIFO

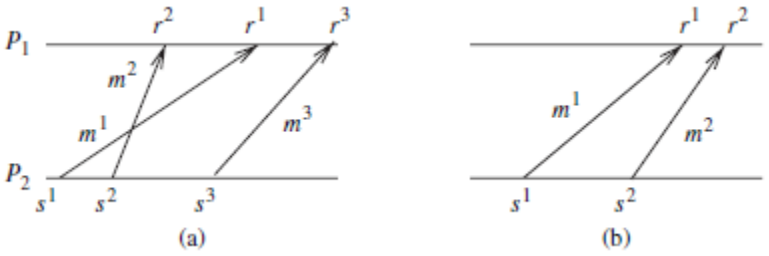
(iii) causal order, and

(iv) synchronous order

Asynchronous executions

An asynchronous execution (or A-execution) is an execution (E, $\prec$) for which the causality relation is a partial order**.**

On any logical link between two nodes in the system, messages may be delivered in any order, not necessarily first-in first-out. Such executions are also known as non-FIFO executions. Although each physical link typically delivers the messages sent on it in FIFO order due to the physical properties of the medium, a logical link may be formed as a composite of physical links and multiple paths may exist between the two end points of the logical link. As an example, the mode of ordering at the Network Layer in connectionless networks such as IPv4 is non-FIFO. The following Figure (a) illustrates an A-execution under non-FIFO ordering.



a) An A-execution that is not a FIFO execution.

(b) An A-execution that is also a FIFO

FIFO executions

A FIFO execution is an A-execution in which,

for all (s, r) and $(s', r') \in$ T, $(s \sim s' and\ r \sim r' and\ s \prec s') \Rightarrow r \prec r'$

On any logical link in the system, messages are necessarily delivered in the order in which they are sent. Although the logical link is inherently non- FIFO, most network protocols provide a connection-oriented service at the transport layer.

A simple algorithm to implement a FIFO logical channel over a non-FIFO channel would use a separate numbering scheme to sequence the messages on each logical channel. The sender assigns and appends a (sequence_num, connection_id) tuple to each message. The receiver uses a buffer to order the incoming messages as per the sender's sequence numbers, and accepts only the "next" message in sequence. The above Figure (b) illustrates an A-execution under FIFO ordering.

Causally ordered (CO) executions

A CO execution is an A-execution in which,

for all (s, r) and $(s', r') \in$ T, $(r \sim r' and\ s \prec s') \Rightarrow r \prec r'$

If two send events s and s' are related by causality ordering (not physical time ordering), then a causally ordered execution requires that their corresponding receive events r and r' occur in the same order at all common destinations. Note that if s and s' are not related by causality, then CO is vacuously satisfied because the antecedent of the implication is false.

Causal order is useful for applications requiring updates to shared data, implementing distributed shared memory, and fair resource allocation such as granting of requests for distributed mutual exclusion.

To implement CO, we distinguish between the arrival of a message and its delivery. A message *m* that arrives in the local OS buffer at P_i may have to be delayed until the messages that were sent to P_i causally before *m* was sent (the "overtaken" messages) have arrived and are processed by the application. The delayed message *m* is then given to the application for processing. The event of an application processing an arrived message is referred to as a delivery event (instead of as a receive event) for emphasis.

Definition of causal order (CO) for implementations

If $send(m^1) \prec send(m^2)$ then for each common destination d of messages m^1 and m^2, $deliver_d(m^1) \prec deliver_d(m^2)$ must be satisfied.

Observe that if the definition of causal order is restricted so that m^1 and m^2 are sent by the same process, then the property degenerates into the FIFO property. In a FIFO execution, no message can be overtaken by another message between the same (sender, receiver) pair of processes. The FIFO property which applies on a per-logical channel basis can be extended globally to give the CO property. In a CO execution, no message can be overtaken by a chain of messages between the same (sender, receiver) pair of processes.

Message order (MO)

A MO execution is an A-execution in which,

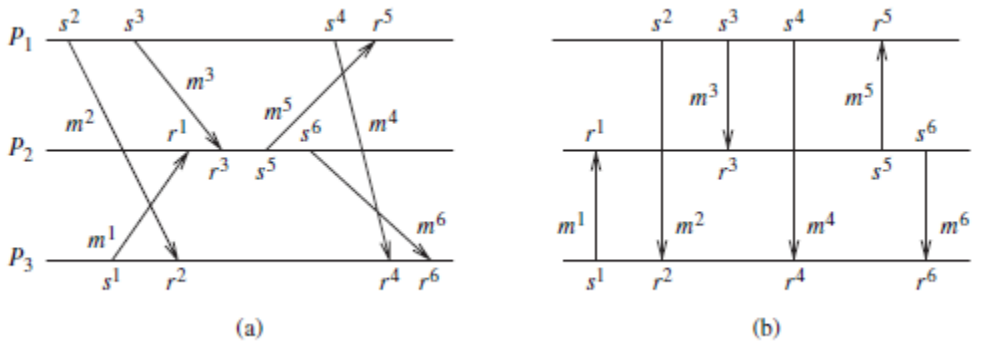
$$\text{for all } (s, r) \text{ and } (s', r') \in \mathrm{T}\ s \prec \mathrm{s}' \Rightarrow \neg(r \prec \mathrm{r}')$$

Empty-interval execution

An execution (E,$\prec$) is an empty-interval (EI) execution if for each pair of events $(s, r) \in T$, the open interval set $\{x \in E | s \prec x \prec r\}$ in the partial order is empty.

Synchronous execution (SYNC)

When all the communication between pairs of processes uses synchronous send and receive primitives, the resulting order is the synchronous order. As each synchronous communication involves a handshake between the receiver and the sender, the corresponding send and receive events can be viewed as occuring instantaneously and atomically.



a) Execution in an Asynchronous system

b) Equivalent instantaneous communication

In a timing diagram, the "instantaneous" message communication can be shown by bidirectional vertical message lines. Figure (a) shows a synchronous execution on an asynchronous system. Figure (b) shows the equivalent timing diagram with the corresponding instantaneous message communication.

The "instantaneous communication" property of synchronous executions requires a modified definition of the causality relation because for each $(s, r) \in T$, the send event is not causally ordered before the receive event. The two events are viewed as being atomic and simultaneous, and neither event precedes the other.

Causality in a synchronous execution

The synchronous causality relation $\ll$ on E is the smallest transitive relation that satisfies the following:

S1: If x occurs before y at the same process, then $x \ll y$.
S2: If $(s, r) \in \mathcal{T}$, then for all $x \in E$, $[(x \ll s \iff x \ll r)$ and $(s \ll x \iff r \ll x)]$.
S3: If $x \ll y$ and $y \ll z$, then $x \ll z$.

Synchronous execution

A synchronous execution (or S-execution) is an execution (E, $\ll$) for which the causality relation $\ll$ is a partial order.

Timestamping a synchronous execution

An execution (E, $\prec$) is synchronous if and only if there exists a mapping from E to T (scalar timestamps) such that

for any message M, $T(s(M)) = T(r(M))$;
for each process P_i, if $e_i \prec e_i'$ then $T(e_i) < T(e_i')$.

By assuming that a send event and its corresponding receive event are viewed atomically, i.e., s(M) $\prec$ r(M) and r(M) $\prec$ s(M), it follows that for any events e_i and e_j that are not the send event and the receive event of the same message, $e_i \prec e_j \Rightarrow T(e_i) < T(e_j)$.