Goals of the lecture

- Hierarchy of communication modes
- Motivation for synchronous ordering
- Crown
- Sufficient conditions for synchronous ordering
- Implementation rules
- Safety theorem
- Liveness
Communication Modes

• FIFO

\[ s_1 < s_2 \implies \neg (r_2 < r_1) \]

— sequence numbers are sufficient to implement FIFO.

• Causally Ordered

\[ s_1 \rightarrow s_2 \implies \neg (r_2 < r_1) \]

— matrix clocks are sufficient to implement Causal ordering.
Communication Modes [Contd.]

- Synchronous Ordering (SYNC)

\[ \exists T: \mathcal{E} \rightarrow \mathbb{N} : \quad \forall s, r, e, f \in \mathcal{E} \]
\[ s \sim r \implies T(s) = T(r) \]
\[ e < f \implies T(e) < T(f) \]

- time diagram of a synchronous computation can be drawn such that all message arrows are vertical.
- any order clock is insufficient to implement synchronous ordering.
Motivation for Synch.

\[ p_1 \]
1: insert (queue \( q_1 \), c)
2: insert (queue \( q_2 \), d)

\[ p_2 \]
1: insert (queue \( q_2 \), a)
2: insert (queue \( q_1 \), b).

State

\[
\begin{array}{c|ccc}
q_1 & c & b \\
q_2 & d & a \\
\end{array}
\]

\[
\begin{array}{c|ccc}
q_1 & c & b \\
q_2 & a & d \\
\end{array}
\]

\[
\begin{array}{c|ccc}
q_1 & b & c \\
q_2 & a & d \\
\end{array}
\]
Crows in Distributed Computation

- A computation is synchronous iff there does not exist a sequence of send and corresponding receive events such that

\[ s_0 \rightarrow \varepsilon r_1, s_1 \rightarrow \varepsilon r_2, \ldots, s_{k-2} \rightarrow \varepsilon r_{k-1}, s_{k-1} \rightarrow \varepsilon r_0. \]

- such a structure is called crown.

- Example:

(A) : Crown of size 2

(B) : Strong Crown of size 3
Algorithm

- Commit Point of a Message

- Priority Rule (RP)

(a) The message along with the underlying message (b) The resulting message ordering
Algorithm [Contd.]

- Send Condition

\[ s_1 < s_2 \implies r_1 \rightarrow r_2 \]

- Receive Condition

\[ s_1 < s_2 \implies s_1.ack < s_2 \]
Implementation

- **Send Condition**
  
  $(SC)\quad s_1 < s_2 \implies r_1 \rightarrow r_2$

  $(SP)\quad s_1 < s_2 \implies s_1.ack < s_2$ (Wait for ack)

- **Receive Condition**

  $(RC)\quad s_1 < r_2 \implies \neg(r_2 \rightarrow r_1)$

  $(RP)\quad s_1 < r_2 \implies s_1.ack < r_2.ack$ (Send ack if no ack pending)
Basic Lemma

Lemma 1 \((s_1 \rightarrow r_2) \text{ and } (SC) \implies (r_1 \rightarrow r_2) \lor (s_1 \prec r_2)\)

Proof:

Case 1

Case 2

Case 3

\(s_1 \prec s_3 \implies r_1 \rightarrow r_3 \implies r_1 \rightarrow r_2\)
Crown and Strong Crown

Lemma 2 Given $SC$ and $RC$. $\exists CR \implies \exists SCR.$

Proof:

\[
s_1 \to r_2, s_2 \to r_1
\]

Let $\neg(s_1 < r_2)$

\[
\implies r_1 \to r_2 \quad SC
\]

\[
\implies \neg(s_2 < r_1) \quad RC
\]

\[
\neg(s_2 < r_1) \land (s_2 \to r_1) \implies r_2 \to r_1 \quad \square
\]

Lemma 3 Given $SC$, $RC$. $CR(k) \implies SCR (k')$

Proof:

\[
s_{i-1} \to r_i, s_i \to r_{i+1} \text{ s.t. } \neg(s_i < r_{i+1})
\]

\[
\implies r_i \to r_{i+1}
\]

Therefore, $s_{i-1} \to r_i, s_i \to r_{i+1}$ reduced to $s_{i-1} \to r_{i+1}$. $\square$
Safety

\[ \neg SYNCH \iff CR \implies SCR \]

\[ s_0 < r_1, s_1 < r_2, \ldots, s_{k-1} < r_0 \]

i.e.

\[ P(s_i) = P(r_{(i+1) \mod k}) \]

From PR:

\[ P(s_i) > P(r_i) \]

we get

\[ P(s_0) < P(r_0) \]
Liveness

- If $P_k$ wants to send a message then it can eventually succeed.

- $k$ smallest processes will eventually be in active state.
Overhead

- **Priority Rule**
  - for every message \((s, r)\) if \(P(s) < P(r)\):
    + one control message and
    + delay of less than \(2t\) units of time.
  - for every message \((s, r)\) if \(P(s) > P(r)\):
    + no control message and
    + no delay.

- **Send and Receive Protocol**
  + one control message and
  + delay is upper bounded by \(nt\), where \(n\) is equal to the number of processes.