The Model

Node properties
- No shared memory
- No global clock

Channel properties:
- FIFO
- Loss free
- Non-duplicating

The Problem

C1:empty

C1:transfer $50

C1:empty

C1:empty
Distributed Snapshot  
(Global State Recording)

Problems:

- recording a “consistent” state of the global computation
- checkpointing for fault tolerance (rollback, recovery)
- testing and debugging
- monitoring and auditing
- detecting stable properties in a distributed system via snapshots. A property is “stable” if, once it holds in a state, it holds in all subsequent states.
- termination
- deadlock
- garbage collection

Local State and Actions:

- local state: $L_{Si}$
- message send: $send(m_{ij})$
- message receive: $rec(m_{ij})$
- time: $time(x)$

Transitions:

\[
send(m_{ij}) \in L_{Si} \iff time(send(m_{ij})) < time(L_{Si})
\]

\[
rec(m_{ij}) \in L_{Sj} \iff time(rec(m_{ij})) < time(L_{Sj})
\]

Predicates:

\[
transit(L_{Si}, L_{Sj}) = \{ m_{ij} | send(m_{ij}) \in L_{Si} \land ! (rec(m_{ij}) \in L_{Sj}) \}
\]

\[
inconsistent(L_{Si}, L_{Sj}) = \{ m_{ij} | !(send(m_{ij}) \in L_{Si}) \land rec(m_{ij}) \in L_{Sj} \}
\]

Consistent Global State:

\[
\forall i, \forall j : 1 \leq i, j \leq n : \text{inconsistent}(L_{Si}, L_{Sj}) = \Phi
\]
Global-State-Detection Algorithm

Marker-Sending Rule for a Process p:
For each channel c, incident on, and directed away from p: p sends one marker along c after p records its state and before p send further messages along c.

Marker-Receiving Rule for a Process q:
if (q has not recorded its state) then
begin q records its state;
    q records the state of c as the empty sequence;
end
else q records the state of c as the sequence of message
    received along c after q's state was recorded and before
    q received the marker along c.

Detecting a Stable Property
begin
record a global snapshot, S*;
test for the stable property in S*;
end;

Global-State-Detection Algorithm

S₀
\[ p \xrightarrow{\text{empty}} q \]

\( p \) records its state (A) and sends marker M on channel

S₁
\[ p \xrightarrow{M} q \]

\( q \) changes its state and sends message D.

S₂
\[ p \xrightarrow{M} q \]

\( q \) receives the marker and records its state (D) and the incoming channel as empty; \( q \) send marker M' on its outgoing channel.

S₃
\[ p \xrightarrow{M} q \]

on receiving the marker, \( p \) records the channel as having D
Snapshot/State Recording Example

Node Recorded state

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>c1 {} c3 {}</td>
</tr>
<tr>
<td>q</td>
<td>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>

Node Recorded state

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>490 {}</td>
</tr>
<tr>
<td>q</td>
<td>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>
**Snapshot/State Recording Example (Step 2)**

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>c1</td>
</tr>
<tr>
<td>p</td>
<td>490</td>
</tr>
<tr>
<td>q</td>
<td>480</td>
</tr>
<tr>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

**Snapshot/State Recording Example (Step 3)**

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>c1</td>
</tr>
<tr>
<td>p</td>
<td>490</td>
</tr>
<tr>
<td>q</td>
<td>480</td>
</tr>
<tr>
<td>r</td>
<td>485</td>
</tr>
</tbody>
</table>
Snapshot/State Recording Example (Step 4)

490  p  c1  c2  500  q  c3  c4
   r  485

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>490 c1 {20} c2 {}</td>
</tr>
<tr>
<td>q</td>
<td>480 {}</td>
</tr>
<tr>
<td>r</td>
<td>485 {}</td>
</tr>
</tbody>
</table>

Snapshot/State Recording Example (Step 5)

515  p  c1  c2  500  q  c3  c4
    r  485

<table>
<thead>
<tr>
<th>Node</th>
<th>Recorded state</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>490 c1 {20} c2 {25}</td>
</tr>
<tr>
<td>q</td>
<td>480 {}</td>
</tr>
<tr>
<td>r</td>
<td>485 {}</td>
</tr>
</tbody>
</table>