**The Model**

Node properties:
- No shared memory
- No global clock

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

**The Problem**

![Diagram of node and channel with states and transactions]

- C1:empty
- C1:transfer $50
- C1:empty
- C2:empty
- C2:empty
- C2:empty
- C2:empty
- C2: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

**Definitions**

Local State and Actions:
- Local state: \( L_{Si} \)
- Message send: \( send(m_{ij}) \)
- Message receive: \( rec(m_{ij}) \)
- Time: \( time(x) \)

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

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

Consistent Global State:
\[
\forall i, \forall j : 1 \leq i, j \leq n \rightarrow inconsistent(L_{Si}, L_{Sj}) = \emptyset \]

**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 sends 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 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;
Snapshot/State Recording Example

500
p c1 c2 c4
r
500
c3

M = Marker

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

Snapshot/State Recording Example (Step 1)

490
p c1 c2 c3
r q
500
c4

<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>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>

Snapshot/State Recording Example (Step 2)

490
p c1 c2
r q
500
c4

20
M

<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>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>

Snapshot/State Recording Example (Step 3)

470
p c1 c2
r q
500
c4

20
M

<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>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>

Snapshot/State Recording Example (Step 4)

490
p c1 c2 c4
r q
500
c3

20
M

<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>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>

Snapshot/State Recording Example (Step 5)

515
p c1 c2 c4
r q
500
c3

<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>{}</td>
</tr>
<tr>
<td>r</td>
<td>{}</td>
</tr>
</tbody>
</table>