Modeling and Simulation

PDES Introduction
The Null Message Synchronization Algorithm

Outline

- Parallel / Distributed Computers
- Air Traffic Network Example
- Parallel Discrete Event Simulation
  - Logical processes & time stamped messages
  - Local causality constraint and the synchronization problem
- Chandy/Misra/Bryant - Null Message Algorithm
  - Ground rules
  - An algorithm that doesn’t work
  - Deadlock avoidance using null messages

Parallel & Distributed Computers

- Parallel computers (tightly coupled processors)
  - Shared memory multiprocessors
  - Distributed memory multicomputers
- Distributed computers (loosely coupled processors)
  - Networked workstations

<table>
<thead>
<tr>
<th>Parallel Computers</th>
<th>Distributed Computers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical extent</td>
<td>Machine room</td>
</tr>
<tr>
<td></td>
<td>Building, city, global</td>
</tr>
<tr>
<td>Processors</td>
<td>Homogeneous</td>
</tr>
<tr>
<td></td>
<td>Often heterogeneous</td>
</tr>
<tr>
<td>Comm. Network</td>
<td>Custom switch</td>
</tr>
<tr>
<td></td>
<td>Commercial LAN / WAN</td>
</tr>
<tr>
<td>Comm. Latency</td>
<td>A few to tens of</td>
</tr>
<tr>
<td></td>
<td>microseconds</td>
</tr>
<tr>
<td></td>
<td>hundreds of microseconds to seconds</td>
</tr>
</tbody>
</table>

Shared Memory Multiprocessors

<table>
<thead>
<tr>
<th>Processor 1</th>
<th>Processor 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{shared int i: L} </code></td>
<td><code>{shared int i: L} </code></td>
</tr>
<tr>
<td><code>i := i + 1;</code></td>
<td><code>i := i + 1;</code></td>
</tr>
<tr>
<td><code>Unlock(L)</code></td>
<td><code>Unlock(L)</code></td>
</tr>
</tbody>
</table>

| Examples: |
| Sun Enterprises |
| SGI Origin |

Distributed Memory Multiprocessors

<table>
<thead>
<tr>
<th>Processor 1</th>
<th>Processor 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{int j: ... Send(2, &amp;j, sizeof(int)) ...}</code></td>
<td><code>{int j: ... Receive(2, &amp;j, sizeof(int)) ...}</code></td>
</tr>
</tbody>
</table>

| Examples: |
| IBM SP |
| Intel Paragon |

Hardware Platforms

Parallel Computers

Distributed Computers

Shared Memory

Distributed Memory
(multicomputers)

SIMP machines

Network of Workstations
### Event-Oriented World View

**State variables**
- Integer: InTheAir;
- Integer: OnTheGround;
- Boolean: RunwayFree;

**Simulation application**

- **Arrival Event**
  - \( E = \text{smallest time stamp event in PEL} \)
  - Remove \( E \) from PEL
  - \( \text{Now} := \text{time stamp of } E \)
  - Call event handler procedure

- **Landed Event**
  - \( \text{InTheAir} := \text{InTheAir} - 1; \)
  - \( \text{OnTheGround} := \text{OnTheGround} + 1; \)
  - Schedule Departure event (local)
    - \( \text{Now} + 1; \)
    - Call event handler procedure

- **Departure Event**
  - \( \text{OnTheGround} := \text{OnTheGround} - 1; \)
  - Schedule Arrival Event (remote)
    - \( \text{Now} + \text{Delay to reach another airport}; \)

---

### Parallel Discrete Event Simulation

- **LP paradigm appears well suited to concurrent execution**
- **Map LPs to different processors**
  - Multiple LPs per processor OK
- **Communication via message passing**
  - All interactions via messages
  - No shared state variables

#### Parallel Discrete Event Simulation: Example

- **Physical system**
  - LAX
  - ORD
  - JFK

- **Logical process**
  - Interactions among physical processes

- **Simulation**
  - One LP per airport

---

### The “Rub”

- **Golden rule for each process:**
  - “Thou shalt process incoming messages in time stamp order”
- **Local causality constraint**

---

#### Parallel Discrete Event Simulation: Example

- **LP Simulation Example**
  - Now: current simulation time
  - \( \text{InTheAir} \): number of aircraft landing or waiting to land
  - \( \text{OnTheGround} \): number of landed aircraft
  - RunwayFree: Boolean, true if runway available

- **Arrival Event**
  - \( \text{InTheAir} := \text{InTheAir} + 1; \)
  - if( RunwayFree )
    - RunwayFree := false;
    - Schedule Landed event (local)
    - \( \text{Now} + \text{R}; \)

- **Landed Event**
  - \( \text{InTheAir} := \text{InTheAir} - 1; \)
  - \( \text{OnTheGround} := \text{OnTheGround} + 1; \)
  - Schedule Departure event (local)
    - \( \text{Now} + \text{G}; \)
  - if( \( \text{InTheAir} > 0 \) ) Schedule Landed event (local)
    - \( \text{Now} + \text{R}; \)
  - else RunwayFree := true;

- **Departure Event**
  - \( \text{OnTheGround} := \text{OnTheGround} - 1; \)
  - Schedule Arrival Event (remote)
    - \( \text{Now} + \text{Delay to reach another airport}; \)
    - \( \text{Now} + \text{R}; \)

---

#### Parallel Discrete Event Simulation: Example

- **Safe to Process?**
  - ORD
  - SFO
  - SFO
The Synchronization Problem

**Synchronization Problem:** An algorithm is needed to ensure each LP processes events in time stamp order.

**Observation:** Ignoring events with the same time stamp (for now), adherence to the local causality constraint is sufficient to ensure that the parallel simulation will produce exactly the same results as a sequential execution where all events across all LPs are processed in time stamp order.

Synchronization Algorithms

- **Conservative synchronization:** Avoid violating the local causality constraint (wait until it's safe to process an event)
  - deadlock avoidance using null messages (Chandy/Misra/Bryant)
  - deadlock detection and recovery
  - synchronous algorithms (e.g., execute in "rounds")
- **Optimistic synchronization:** Allow violations of local causality to occur, but detect them at runtime and recover using a rollback mechanism
  - Time Warp (Jefferson)
  - numerous other approaches

Outline

- Parallel / Distributed Computers
- Air Traffic Network Example
- Parallel Discrete Event Simulation
  - Logical processes
  - Local causality constraint
- Chandy/Misra/Bryant Null Message Algorithm
  - Ground rules
  - An algorithm that doesn’t work
  - Deadlock avoidance using null messages

Conservative Algorithms

**Assumptions:**
- logical processes (LPs) exchanging time stamped events (messages)
- static network topology, no dynamic creation of (and connection of LPs)
- messages sent on each link are sent in time stamp order
- network provides reliable delivery, preserves order (received in same order that they are sent)

**Observation:** The above assumptions imply the time stamp of the last message received on a link is a lower bound on the time stamp (LBTS) of subsequent messages received on that link.

A Simple Conservative Algorithm

**Goal:** Ensure LP processes events in time stamp order

**Algorithm A** (executed by each LP):
**Goal:** Ensure events are processed in time stamp order:

```
while( simulation is not over )
  wait until each FIFO contains at least one message
  remove smallest time stamped event from its FIFO
  process that event
end-loop
```

- process time stamp 2 event
- process time stamp 4 event
- process time stamp 5 event
- wait ( block ) until a message is received from ORD.
Deadlock Example

A cycle of LPs forms where each is waiting on the next LP in the cycle. No LP can advance; the simulation is deadlocked.

**Observation:** Algorithm A is prone to deadlock! (cycle of empty queues...)

Deadlock Avoidance Using Null Messages

**Null Message Algorithm** (executed by each LP):

**Goal:** Ensure events are processed in time stamp order and avoid deadlock

```
while (simulation is not over)
    wait until each FIFO contains at least one message
    remove smallest time stamped event from its FIFO
    process that event
    send null messages to neighboring LPs with time stamp indicating a lower bound on future messages sent to that LP (current time plus lookahead)
```

The null message algorithm relies on a “lookahead” (flight time in the example) ability.

Summary

- **Parallel Discrete Event Simulation**
  - Collection of sequential simulators (LPs) possibly running on different processors
  - Logical processes communicating exclusively by exchanging messages

- **Chandy/Misra/Bryant Null Message Algorithm**
  - Null messages: Lower bound on the time stamp of future messages the LP will send
  - Null messages avoid deadlock (non-zero lookahead)

Parallel Discrete Event Simulation: Example

All interactions between LPs must be via messages (no shared state).