CSCI 8220 Parallel and Distributed Simulation

PDES Introduction
The Null Message Synchronization Algorithm

Parallel and Distributed Computers

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

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

Shared Memory Multiprocessors

- Programming model: shared variables; synchronization via locks

Distributed Memory Multiprocessors

- Programming model: no shared variables; message passing

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

Hardware Platforms

Parallel Computers
- Distributed Memory
  - SIMD machines
- Network of Workstations

Distributed Computers
- Networked workstations
- Distributed Memory (multicomputers)
Parallel Discrete Event Simulation

- Extends example to model a network of airports
  - Encapsulate each airport simulator in a logical process
  - Logical processes can schedule events (send messages) for other logical processes

More generally...
- Physical system
  - Collection of interacting physical processes (airports)
- Simulation
  - Collection of logical processes (LPs)
  - Each LP models a physical process
  - Interactions between physical processes modeled by scheduling events between LPs

The “Rub”

Golden rule for each process:
“Thou shalt process incoming messages in time stamp order”

Local causality constraint
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)
  - 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

Conservative Algorithms

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

**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.

Goal: Ensure LP processes events in time stamp order

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

A Simple Conservative Algorithm

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

```plaintext
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
```

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

Goal: Ensure LP processes events in time stamp order
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.

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)

end-loop

The null message algorithm relies on a "lookahead" ability.

Parallel Discrete Event Simulation: Example

Physical system

- Interactions among physical processes

Logical process

- Time stamped event (message)

Simulation

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

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)