The Synchronization Problem

Richard M. Fujimoto
Professor

Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

http://www.cc.gatech.edu/~fujimoto/

Problem Statement

Execute a discrete event simulation (DES) program on a parallel computer

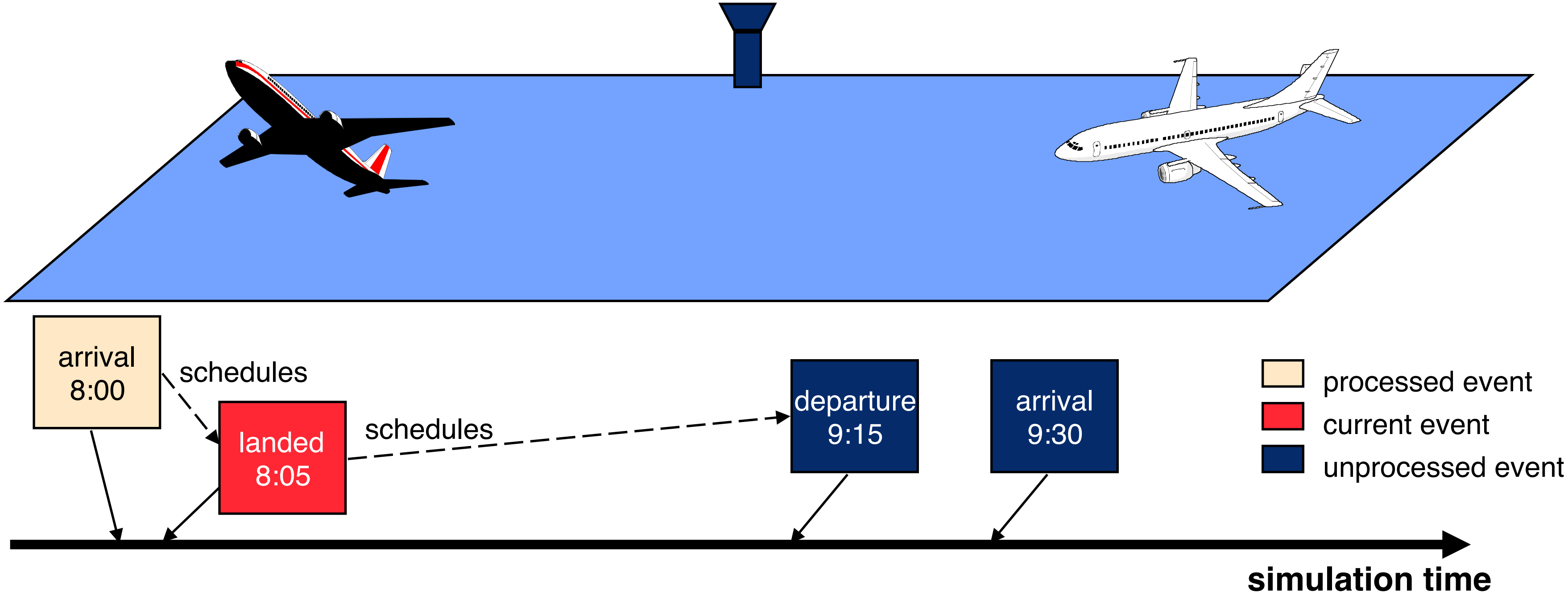
Outline

- Start with a sequential DES
- Extend to a parallel simulation
- Synchronization problem
- Solution approaches
  - Conservative synchronization / Lookahead
  - Optimistic synchronization

Example: Air Traffic Simulation

Air traffic at an airport; single runway for incoming flights

Aircraft arrive, queue to use runway, land, spend time at gate & depart



Events

- Aircraft arrival
- Aircraft landed
- Aircraft departure

Event-Oriented Sequential Simulation

state variables

```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

Event handler procedures

```
Arrival
   Event
{
    …
}

Landed
   Event
{
    …
}

Departure
   Event
{
    …
}
```

Simulation application

Simulation executive

Event processing loop

Now = 8:45

Future Event List (FEL)

9:00

9:16

10:10

```
While (simulation not finished)
   E = smallest time stamp event in FEL
   Remove E from FEL
   Now := time stamp of E
   call event handler procedure
```

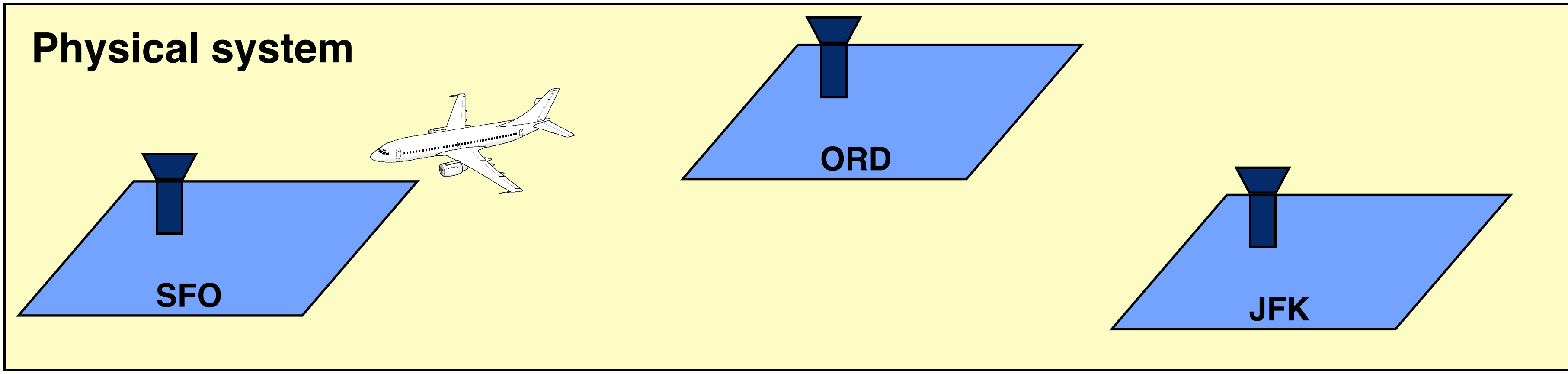
Parallel Discrete Event Simulation

- Example: 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...

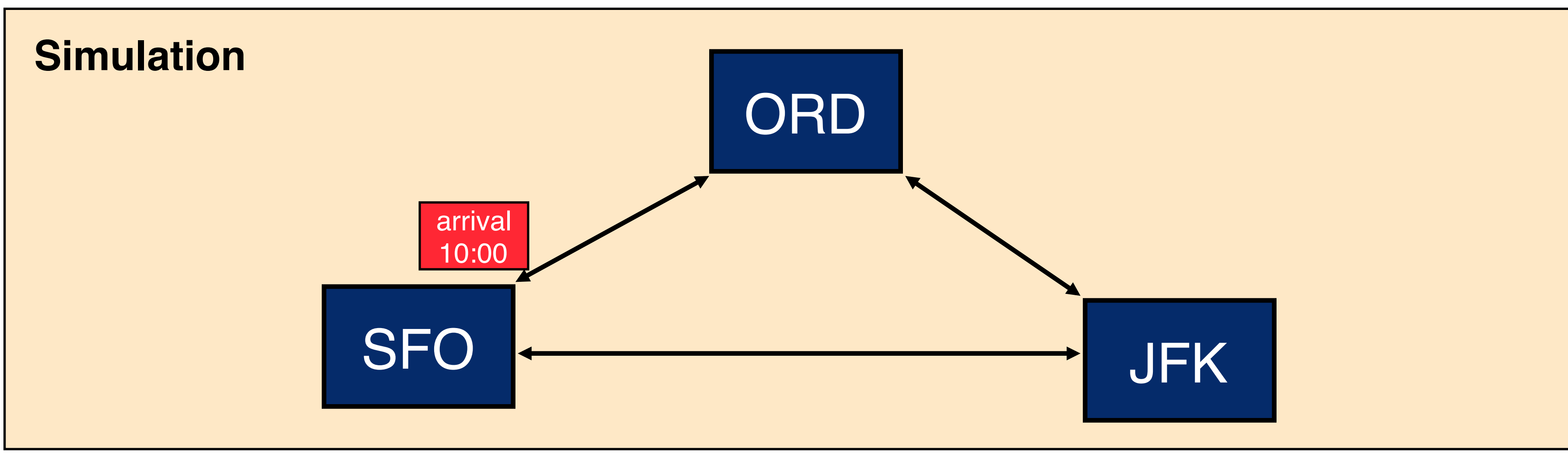
- Physical system
  - Collection of interacting physical processes (airports)
- Simulation
  - Collection of logical processes (LPs)
  - Each LP models a physical process
  - Interactions between physical processes modeled by scheduling events between LPs

Parallel Discrete Event Simulation Example



physical process → **logical process**

interactions among physical processes → **time stamped event (message)**



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

LP Simulation Example

- `Now`: current simulation time
- `InTheAir`: number of aircraft landing or waiting to land
- `OnTheGround`: number of landed aircraft
- `RunwayFree`: Boolean, true if runway available

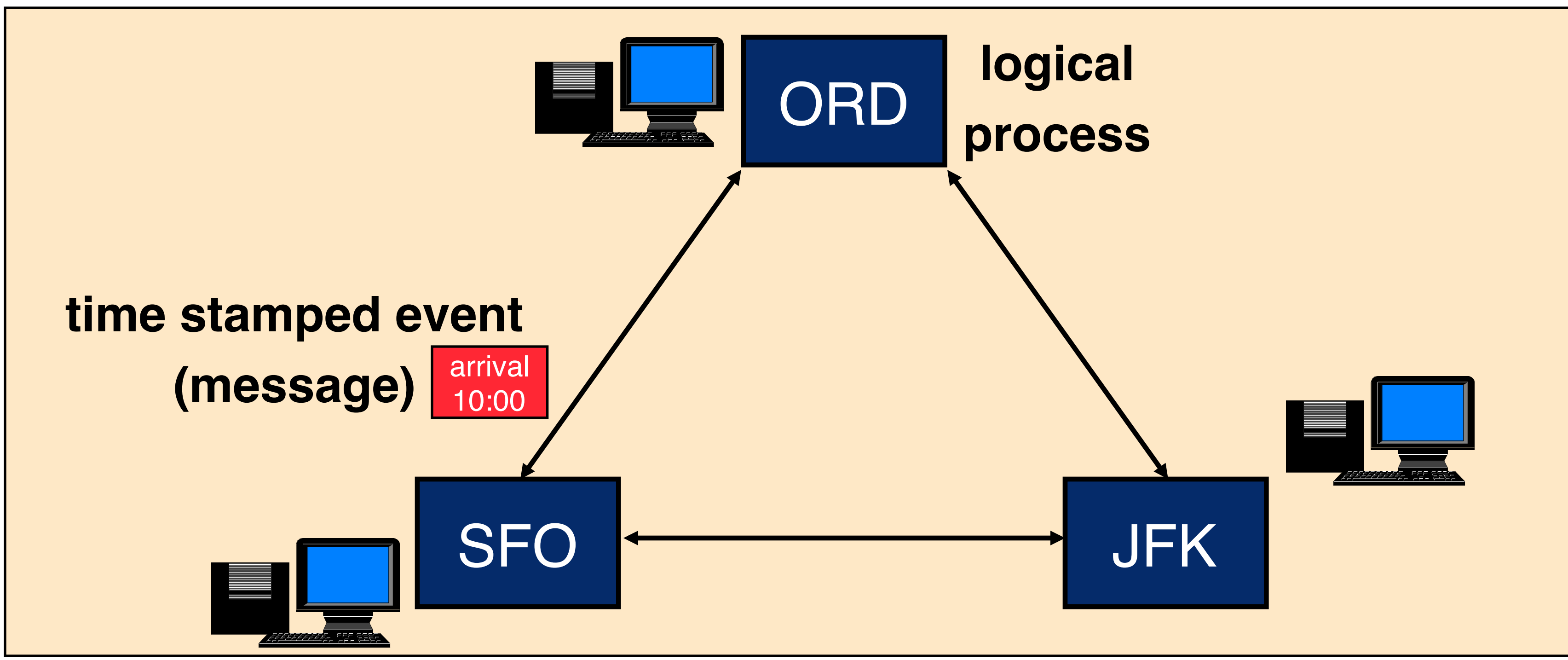
```
Arrival Event:
InTheAir := InTheAir+1;
If (RunwayFree)
   RunwayFree:=FALSE;
   Schedule Landed event (local) @ Now+R;
```

```
Landed Event:
InTheAir:=InTheAir-1;     OnTheGround:=OnTheGround+1;
Schedule Departure event (local) @ Now + G;
If (InTheAir>0) Schedule Landed event (local) @ Now+R;
Else RunwayFree := TRUE;
```

```
Departure Event (D = delay to reach another airport):
OnTheGround := OnTheGround - 1;
Schedule Arrival Event (remote) @ (Now+D) @ another airport
```

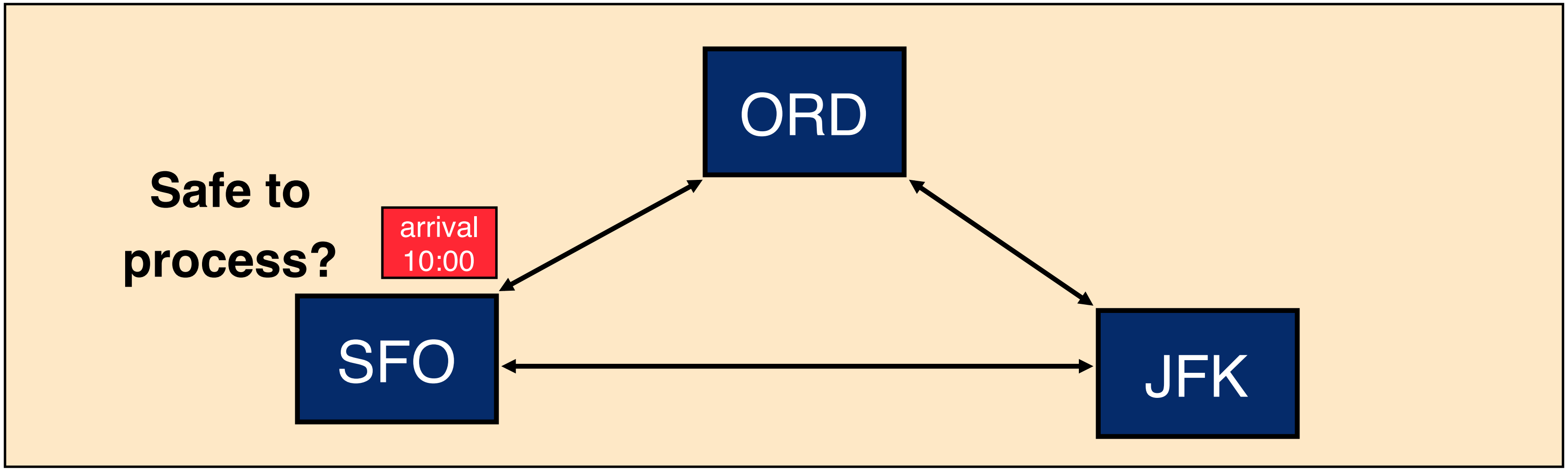
Approach to Parallel/Distributed Execution

- 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



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, 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

Summary

- A parallel discrete event simulation can be viewed as a set of sequential discrete event simulations (logical processes) that exchange time-stamped events (messages)
- Each LP should process events in timestamp order
- This leads to the synchronization problem; solutions include conservative approaches that prevent LPs from processing events out of timestamp order to optimistic algorithms that detect out of order processing of events, and recover using a rollback mechanism