

# Parallel Discrete-Event Simulation

- J. Misra, "Distributed Discrete-Event Simulation," ACM Computing Surveys, 18(1):39--65, March 1986.
- D. Jefferson, B. Beckman, and F. Wieland et al. "Distributed Simulation and the Time Warp Operating System," In Symposium on Operating Systems Principles, Austin, Texas, October 1987.
- Richard M. Fujimoto, "Parallel Discrete Event Simulation," Communications of the ACM, Vol.33, No.10, Oct.1990.
- R. Bagrodia and W.-T. Liao, "Maisie: A Language for the Design of Efficient Discrete-Event Simulations," IEEE Transactions on Software Engineering, Vol. 20(4), April 1994, pp. 225-238.

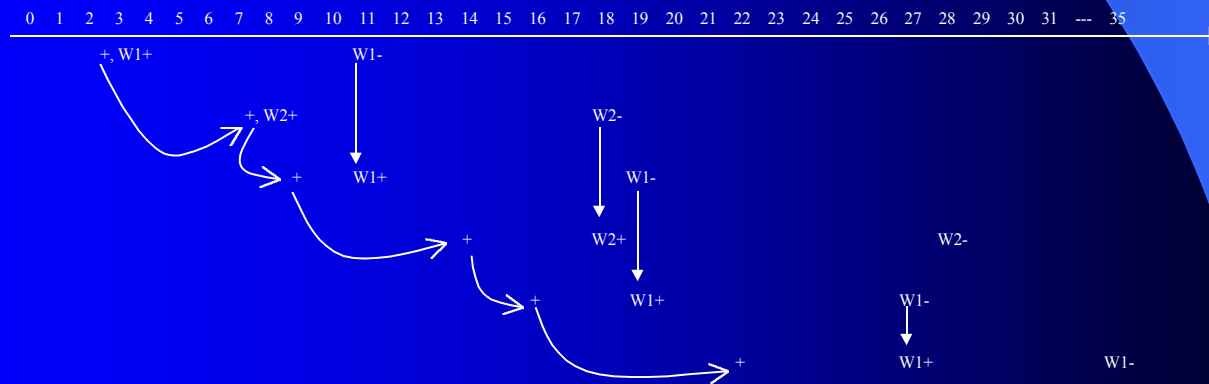
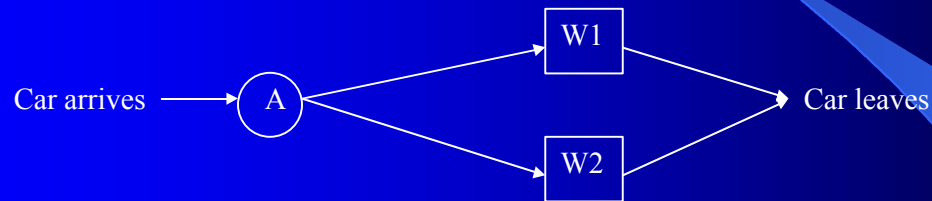
# 1. System Simulation

- Traditional Approach of System Simulation
  - Simulation Clock & FEL (Future Event List)
  - FEL dictates a sequential simulation
  - Complex computer and communication systems are intractable mathematically and must resort to simulation
- Distributed Simulation
  - Shared data objects, clock & FEL, are discarded
  - One machine simulate a single physical process and messages are simulated by message transmission
  - Synchrony is captured by encoding time in messages

## 2. Sequential Simulation

- Physical Systems
  - A system = number of physical processes (pp's)
  - A “pp” is described by a set of events (with time)
  - Dependency relation among all events : eg (e1 -> e2)
- Example : Car Wash System
  - One attendant (A), two car wash machines (W1, W2)
  - W1 takes 8 min., W2 takes 10 min.
  - Rule: guide to an idle machine, guide to W1 if both are idle
  - Events: a car arrives at A, W1 or W2, or a car leaves (3 events per car)
  - 6 cars arrive at times 3, 8, 9, 14, 16, 22 => 18 events

# Ex: Car Wash System (cont'd)



- \* Arrows denote dependency between events.
- \* Dependencies between events on the same line are not marked for convenience.

## Ex: Car Wash System (cont'd)

- 5 pp's : source, A, W1, W2, exit (sink)
- A does not know whether W1 and W2 are idle
- 26 messages = 18 events + 6 done's + 2 init's
  - t=0, W1 to A, "W1 is idle"
  - t=0, W2 to A, "W2 is idle"
  - t=3, S to A, "Car arrived"
  - t=3, A to W1, "Car sent"
  - ...
  - t=11, W1 to E, "Car left"
  - t=11, W1 to A, "W1 is idle"
  - t=11, A to W1, "Car sent"
  - ...

# Sequential Simulation Algorithm

- Event List : a set of tuples  $(t, m)$ ;  $t \geq$  simulation clock,  $m$ : message
- $(t, m)$  is conditional
  - may not actually occur because its transmission can be cancelled (example: preemption)
- However, for the  $(t, m)$ 
  - if “ $t$ ” is the smallest time component among all entries in the event list then the message “ $m$ ” is guaranteed to transmit at time  $t$ , and no other message is transmitted between the simulation clock and “ $t$ ”

## 3. Parallel (Distributed) Simulation

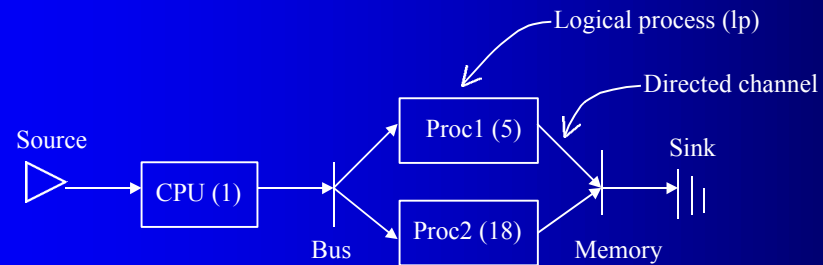
- Motivation
  - To reduce large model's turnaround time
  - Interoperability: connecting multiple, autonomous simulators - each of which might simulate a different components of a complex system and run on geographically distributed machines
- Simulation is carried out by a set of communicating processes

# Distributed Simulation - Basics

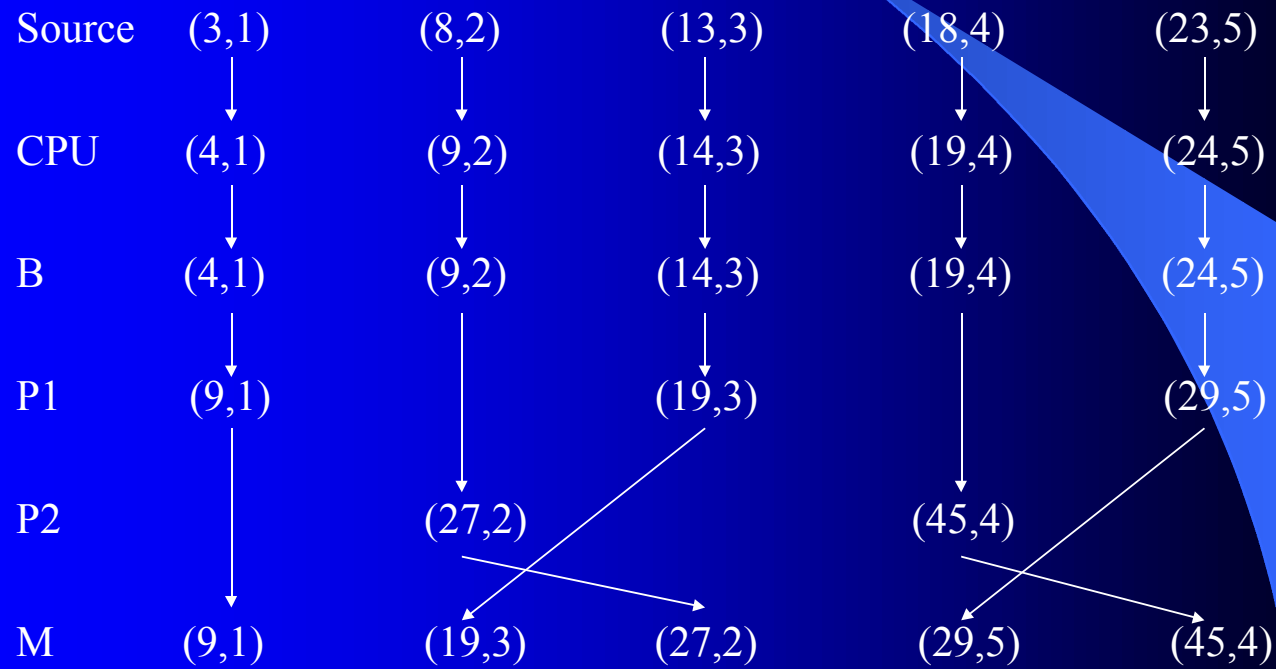
- Logical Systems
  - Number of logical processes (lp's) + directed channels (network)
- Logical Process (lp)
  - Lp's may not operate at the same speed
  - But simulates the exact sequence of message transmissions
  - Each lp executes sequential code and receive/send
- Clock Value (cv)
  - Channel cv: time value of the last received message along that channel
  - Process cv: min. channel clock value of all incoming channels to the lp
  - Simulator cv : min. process clock value of all lp's

# Example

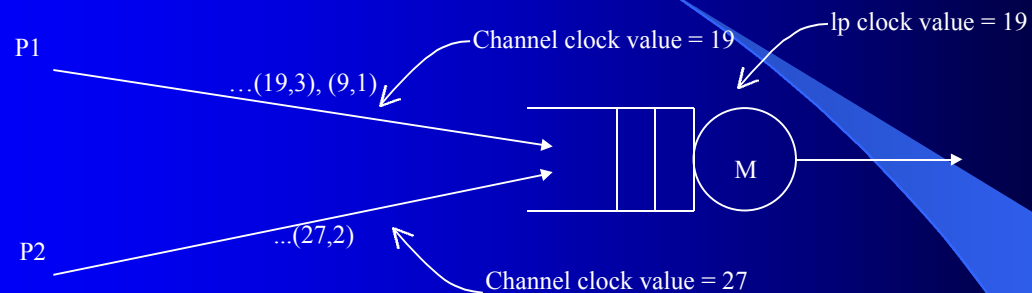
- Jobs arrive every 5 seconds starting from  $t=3$  to 23
- B selects P1 and P2 alternatively
- CPU takes 1 second, Proc1 takes 5 and Proc2 takes 18 seconds.



## Example (cont'd)

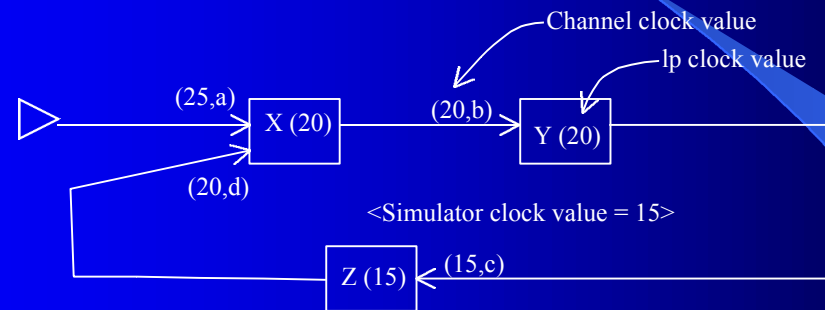


## Example (cont'd)



- This situation is possible since lp's may operate at different speeds
  - $M$  processes  $(9,1)$  and  $(19,3)$
  - but cannot process  $(27,2)$  because  $P1$  might send  $(t,m)$  where  $t < 27$
  - new message  $(29,5)$  from  $P1$ , now  $M$  can process  $(27,2)$
  - new message  $(45,4)$  from  $P2$ , now process  $(29,5)$
  - but cannot process  $(45,4)$  forever
  - \* If only  $(9,1)$  is there initially,  $M$  cannot process it until  $(27,2)$  arrives
  - \* If  $B$  does not send jobs to  $P2$ ,  $M$  cannot process any job forever (deadlock)

# Distributed Simulation - Deadlock

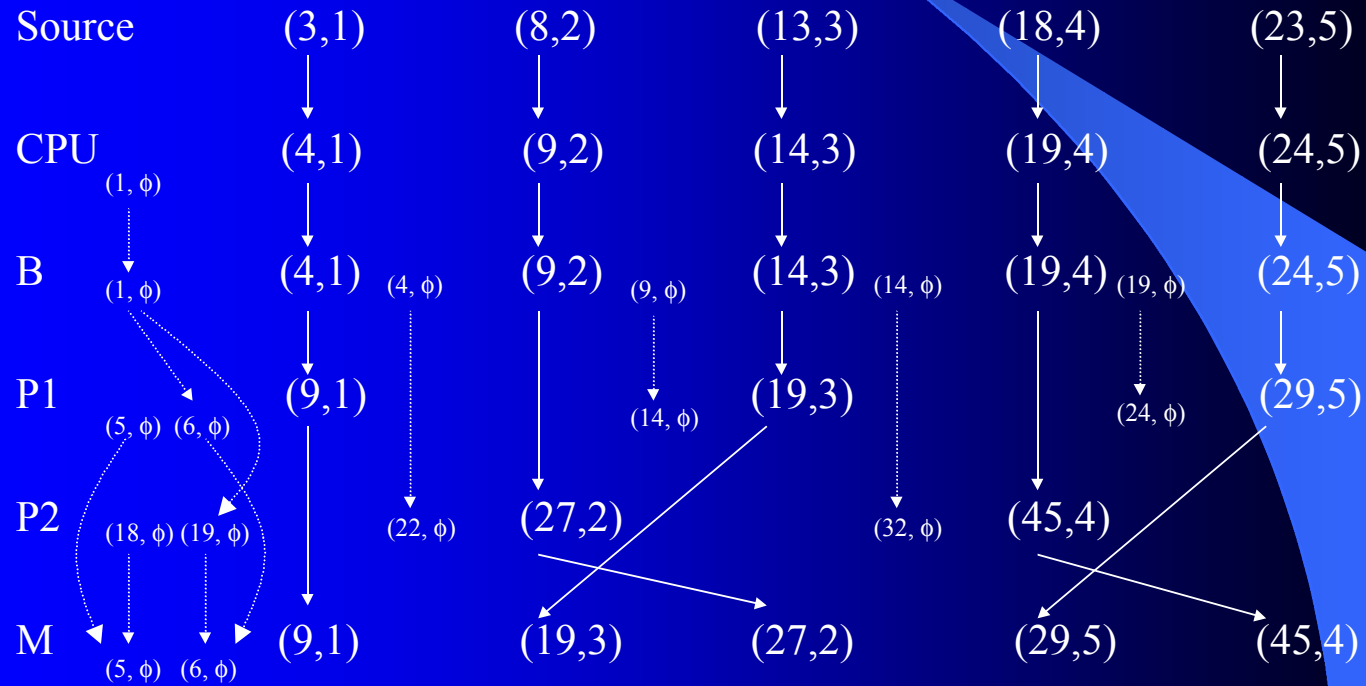


- X,Y,Z would not send new message unless they receive a message
- Deadlock
  - Z would not send a message to X unless X send one to Y
  - There is a new message at the source (25,a) but X cannot process it since X does not sure whether Z will send (t,m) where  $20 < t < 25$
  - Global view says that X can process (25,a) but no local view sure of it.

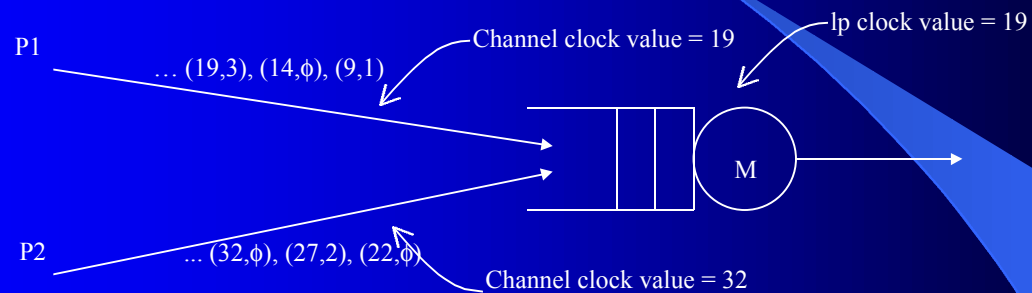
# Distributed Simulation - Deadlock Resolution

- Null message  $lp_i \rightarrow lp_j : (t, \text{null})$ 
  - $lp_i$  will not send any other message between the channel clock value and “t”
  - When  $lp_i$  receives a new message
    - update  $lp$  clock value
    - send a message to “all” its outgoing channels
    - if there is no message to send, send ( $lp$  clock value, null)
  - Null messages tend to be created at “branches”
- Optimization
  - A null message has no effect if it is followed by another message  
=> delay transmission of null messages for efficiency
  - Any message put in the buffer annihilates any null messages ahead of it
  - Demand-driven null message transmission : query-reply message

# Example - Null message



## Example (cont'd)



- M sure that
  - P1 will not send  $(t,m)$  where  $t < 19$
  - P2 will not send  $(t,m)$  where  $t < 32$
- Even if B does not send jobs to P2 (deadlock case)
  - B still send (null) messages to P2 and P2 send to M
  - so that M can process all the messages from P1

## 4. Time Warp Operating System (TWOS)

- Studied and implemented in 1985-
  - at NASA JPL (Jet Propulsion Laboratory) with Caltech
  - on Mark III hypercube multiprocessor and on network of 7 SUN w/s
- Why OS for a simulation ?
  - Performs synchronization by a general distributed process rollback mechanism using virtual time (GVT: global virtual time)
  - It affects every aspect of system software
  - Solves virtual time synchronization problem

# TWOS - Time Warp Mechanism

- Conservative approach
  - Chandy-Misra approach : null-message based
- Optimistic approach
  - Assume (9,m2) is the true next message and process it
  - If (8,m1) arrives later
    - roll back the process to  $t=8$
    - execute (8,m1)
    - re-execute (9,m2)
  - Save periodic snapshot of the state of each process for the rollback
  - Antimessage for the re-execution by cancelling some prior messages

# TWOS - Antimessage

- When lp sends a message,
  - TWOS actually creates message-antimessage pair
  - “+m” is sent to the receiver’s input queue
  - “-m” is retained in sender’s output queue
- In case of rollback:
  - every message is compared with those in the output queue
  - if antimessage is found, discard both of them (it is already sent)
  - if not found, send it and leave the antimessage in the output queue
  - For the remaining antimessages (-m),
    - send it to the receiver’s input queue
    - if “+m” is in the receiver’s input queue, discard both of them
    - otherwise, rollback the receiver process (???)

# Chandy-Misra vs TWOS

- Commons

- Simulation is decomposed of logical processes
- lp's communicate via time-stamped event messages
- asynchronous: some lp's can be ahead of time than the others

- Differences

- different DES paradigms: sequence of intervals vs seq. of discrete events
- requirement: lp's and channels with static topology vs. dynamic topology
- synchronization: conservative vs optimistic
- difficulty: deadlock handling vs state-saving and antimessage handling
- Overhead where there is "inactivity" vs "activity"

## 5. Research Groups

- Parallel Simulation and related projects in UCLA, led by Rajive Bagrodia
  - parallel simulation languages (Masie, Parsec) and protocols
  - VLSI designs, networks, and parallel programs
- PADS group in Georgia Tech, led by Richard Fujimoto
  - Time Warp based parallel simulation (GTW) of telecommunication networks (Ted project)
- ALSP (Aggregate Level Simulation Protocol) at MITRE
- Web-based Simulation at MITRE
- HLA (High Level Architecture) at DoD
- The Warp Group at Univ. of Calgary, Canada, led by Brian Unger
  - Parallel Discrete Event Simulation with optimistic synchronization schemes
  - Current projects are focused on efficient GVT Computation, Memory Management, Incremental State Saving, Visual Interactive Parallel Simulation
  - modeling telecommunication system : SS7 and ATM systems

- PARADISE Project at Stanford , led by David Cheriton
  - PARADISE (Performance ARchitecture for Advanced Distributed Interactive Simulation Environments)
  - large-scale internetworked simulation environment for multi-player interactive, 3D-simulations running over a wide-area network.
- TEMPEST (Territorially Explicit Massively Parallel Ecological Simulation Tool) at RPI
  - simulation of both population dynamics and epidemiological phenomena (Implemented on MasPar)
- Wisconsin Wind Tunnel Project
  - execution-driven, distributed, discrete-event simulation to accurately calculate program execution time
- ParaSol at Purdue led by Vernon Rego
  - Parallel discrete event simulation system with optimistic and adaptive synchronization methods
  - on clusters of workstations or on shared memory multiprocessors
- Computer Architecture Design Laboratory, in University of Cincinnati, led by Philip Wilsey
  - Time Warp simulation kernels and parallel simulation of VHDL
- SimLab, Department of Teleinformatics, Royal Institute of Technology, Sweden by Rassul Ayani
  - parallel simulation of computer architectures and mobile telecommunication systems.