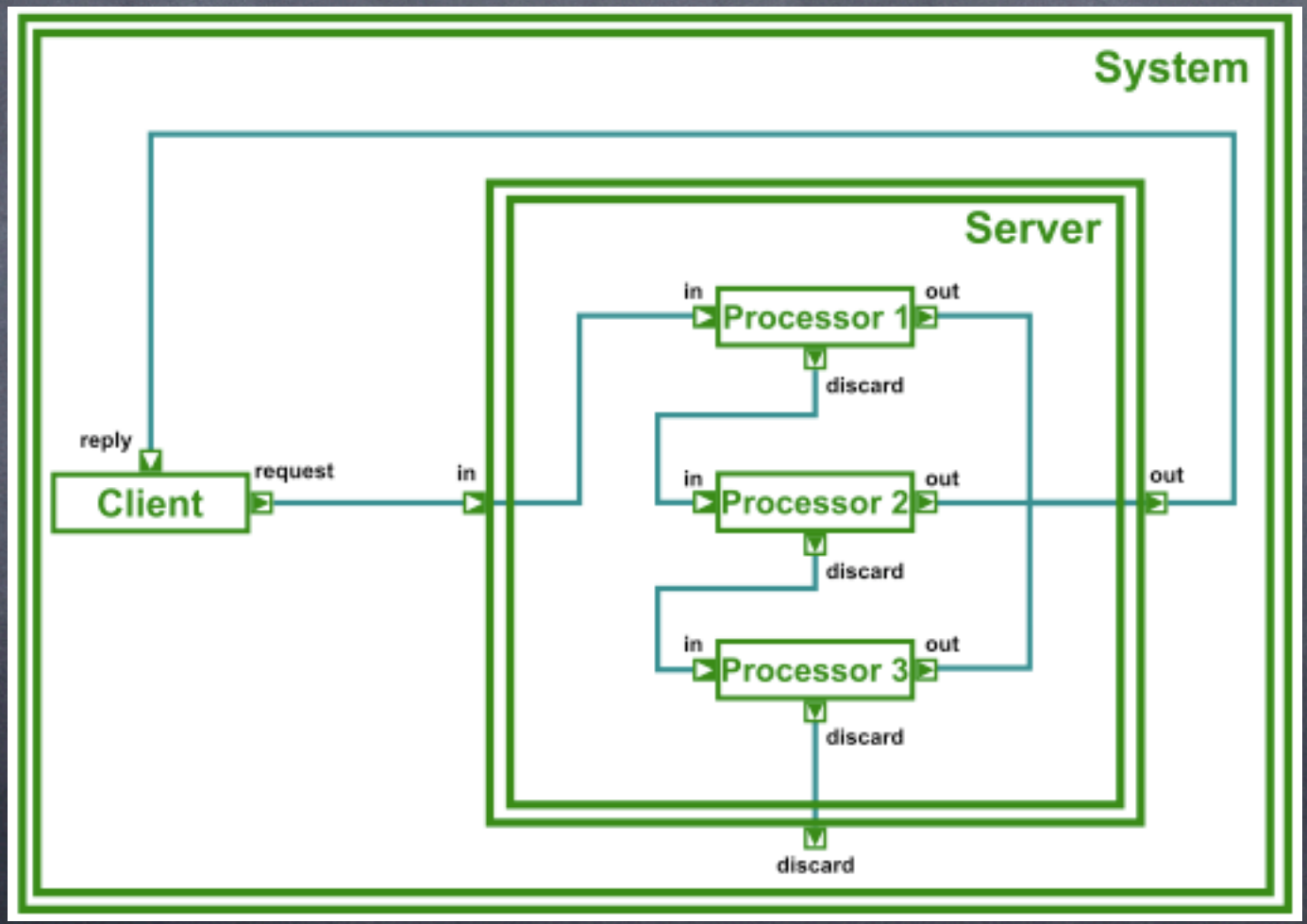
Distributed classic DEVS simulator with TimeWarp

Amr Al Mallah

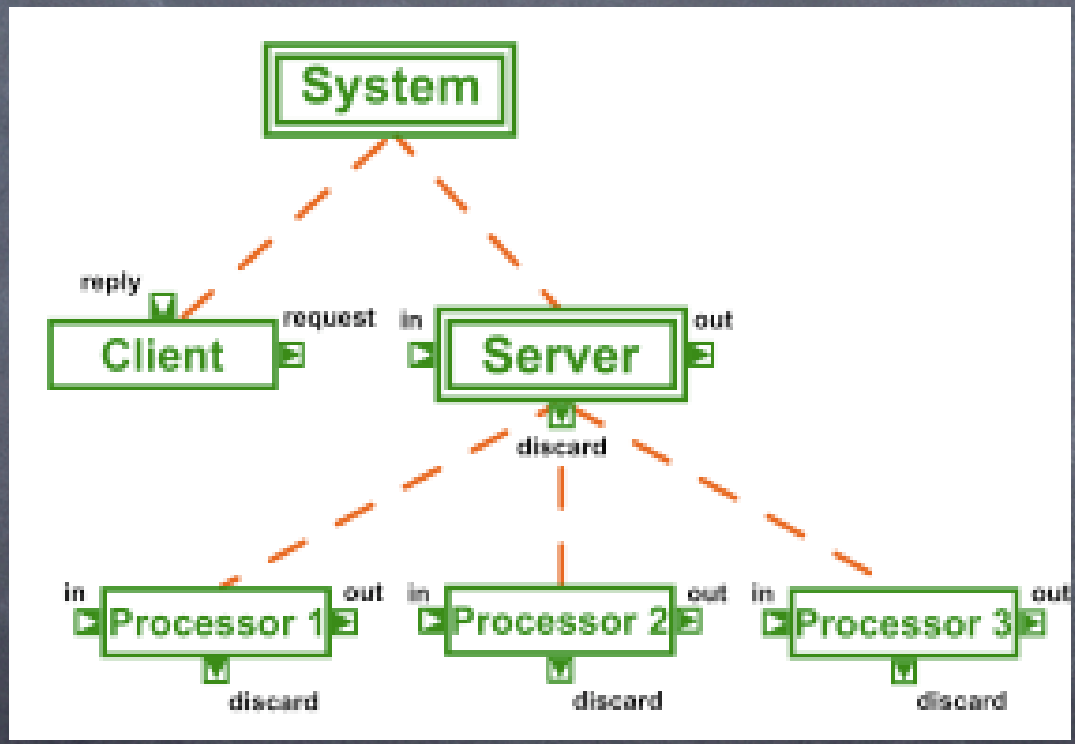
Outline

- Classical DEVS Simulator
- Distributed DEVS Simulator
- Conservative Parallel Simulator
- **Optimistic (Time warp) Parallel Simulator**
- Conclusion

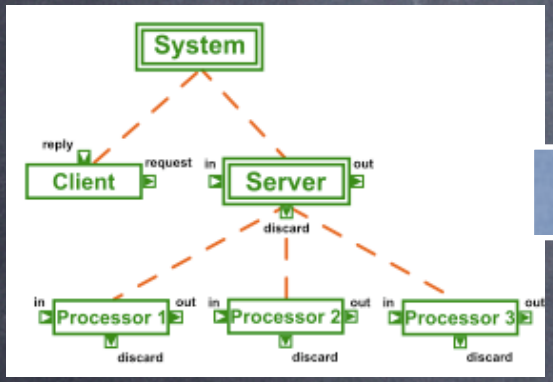
Classical DEVS Simulator

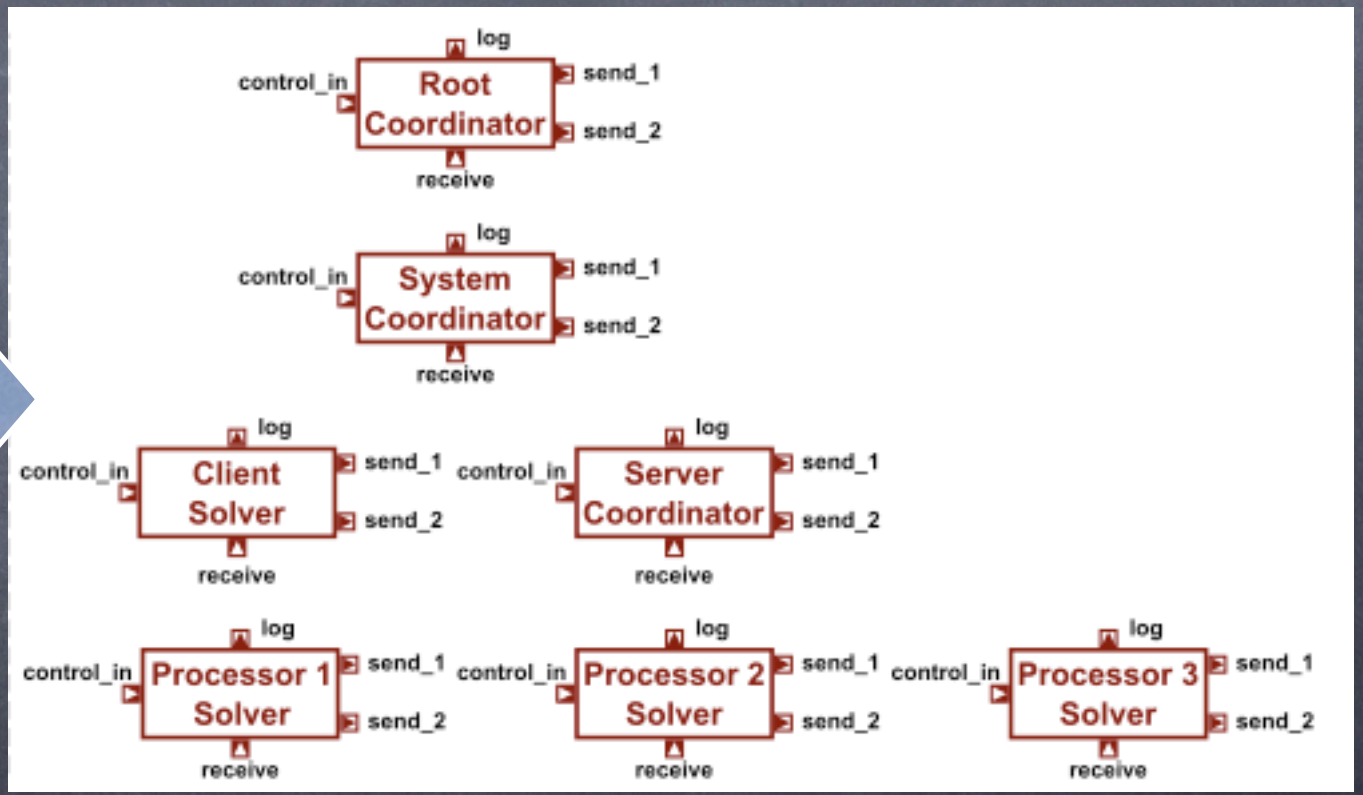


Classical DEVS Simulator

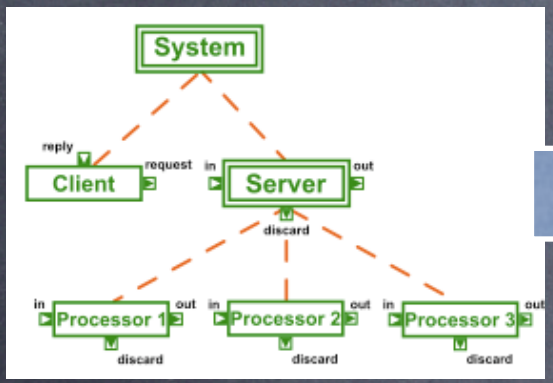


Classical DEVS Simulator

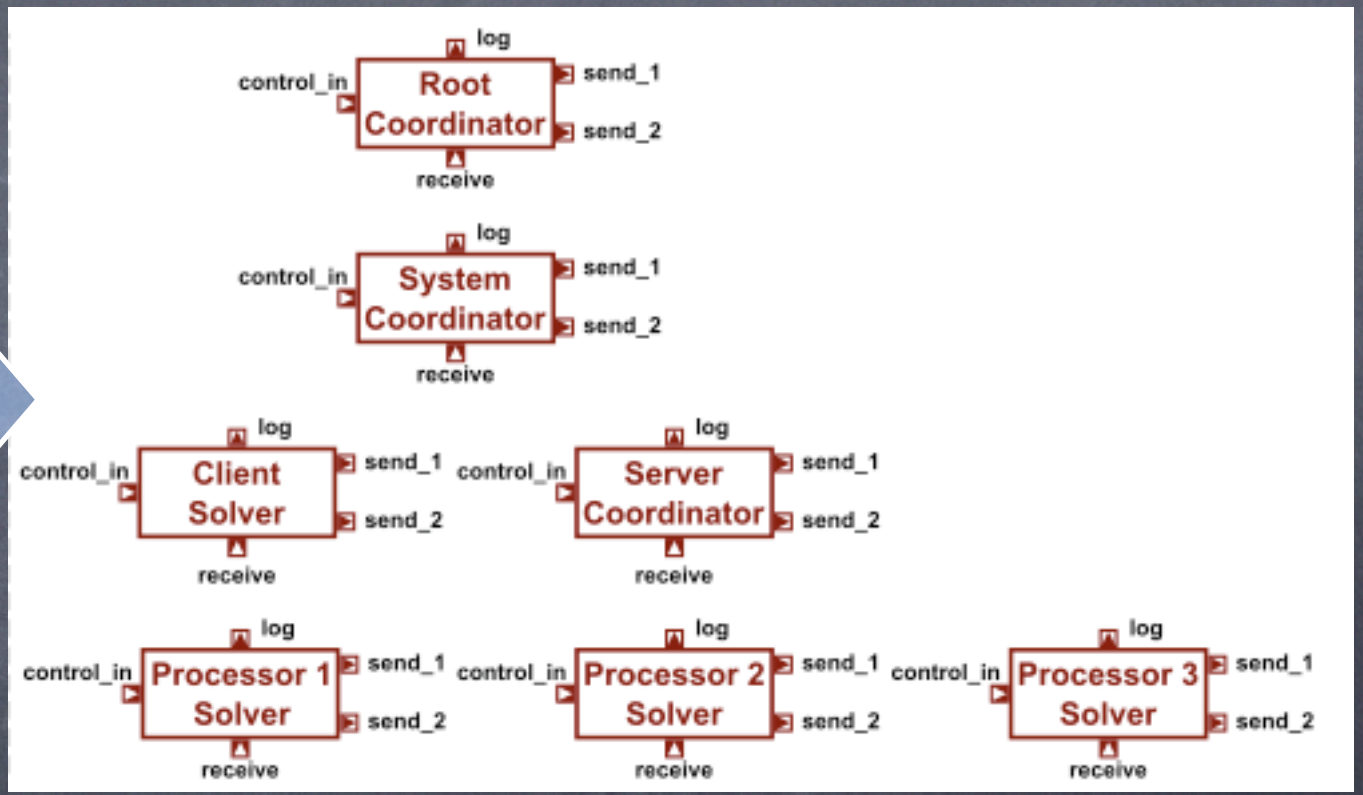




Classical DEVS Simulator



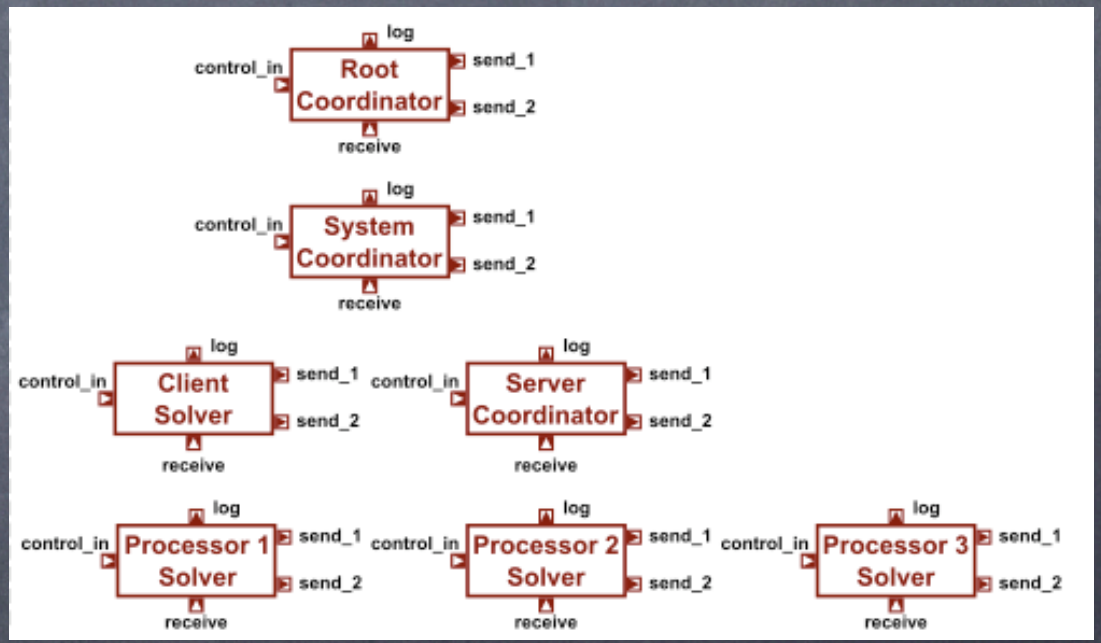
Data structures



Data structures + simulation state (Time Last, Time Next, Event List)

Classical DEVS Simulator

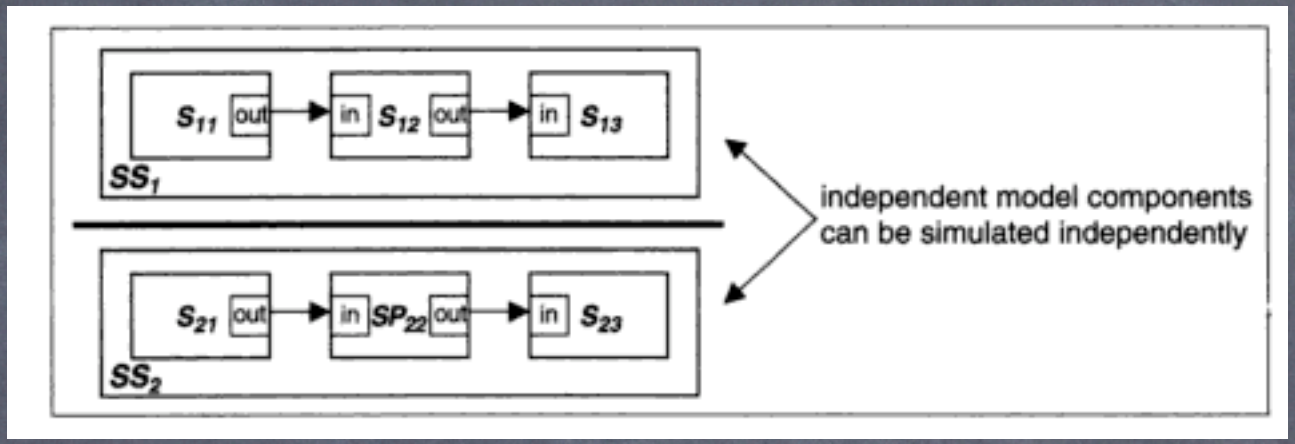
- Sequential and single thread of execution.



Distributed DEVS Simulation

- Models and their simulators can be spread over several machines allowing scalability and resource sharing.
  - Increased Speed
  - Size of models
  - Specialized nodes capabilities
  - Interoperability.

Distributed DEVS Simulation



- order doesn't matter:
  - generate the complete behavior of one then the other.
  - or generate on different processors.

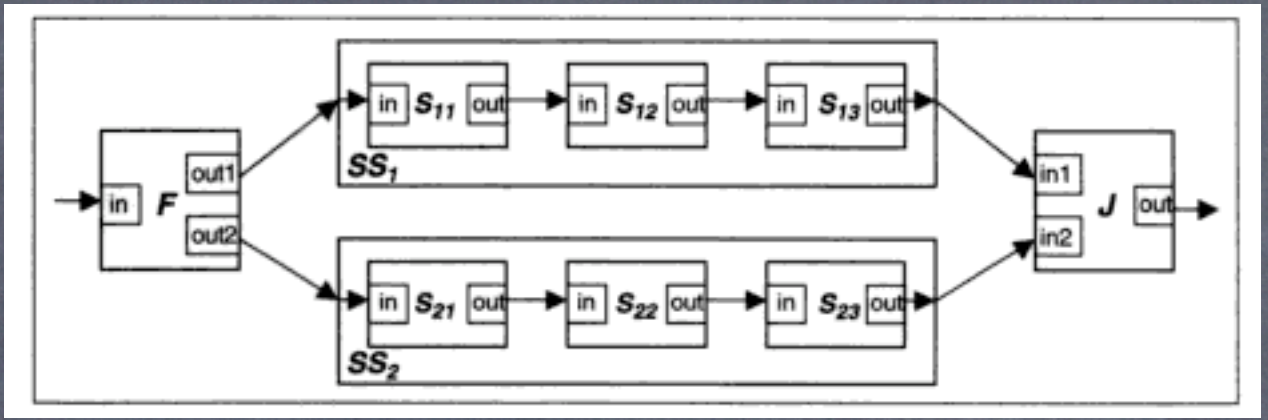
Distributed DEVS Simulation

- Almost all systems have dependent components
- Components usually run independently for a period of time.
- Components could show dependency at some intervals by means of events.
- How to optimize the simulation to exploit these aspects ?

Distributed DEVS Simulation

- Causal dependency:
  - When event in one component affect events in another components
- The correct execution needs to preserve the causality of events.
  - for event x, all events Yi on which x depends, must be processed before x is.

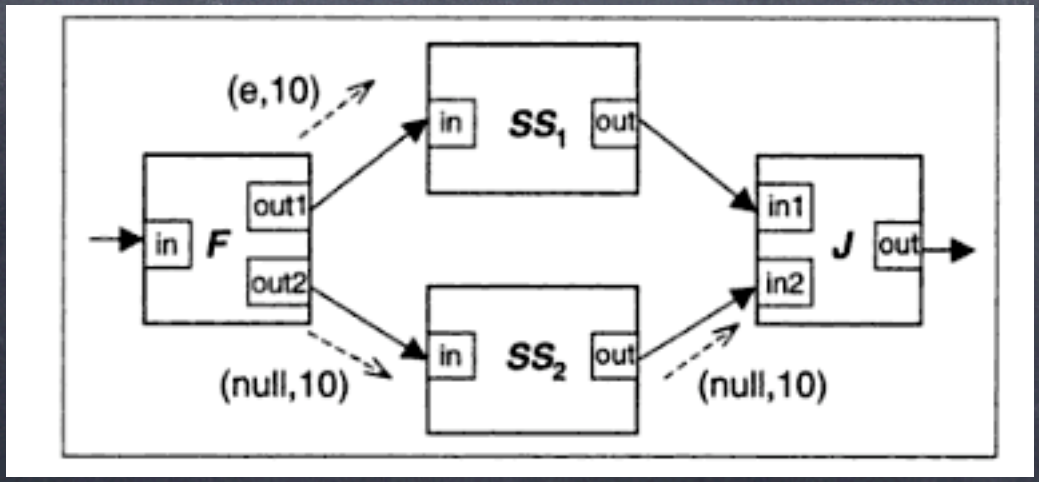
Distributed DEVS Simulation



- No direct dependency between SS1 & SS2, they can be simulated independently.
- But Events Have to be Synchronized at outputs to F, and inputs of J.
- When Should J wait ?

Conservative Parallel DEVS simulator

- Idea: Make sure It's safe before processing
  - Send Extra messages around to do the synchronization.
  - Null messages and look ahead properties.

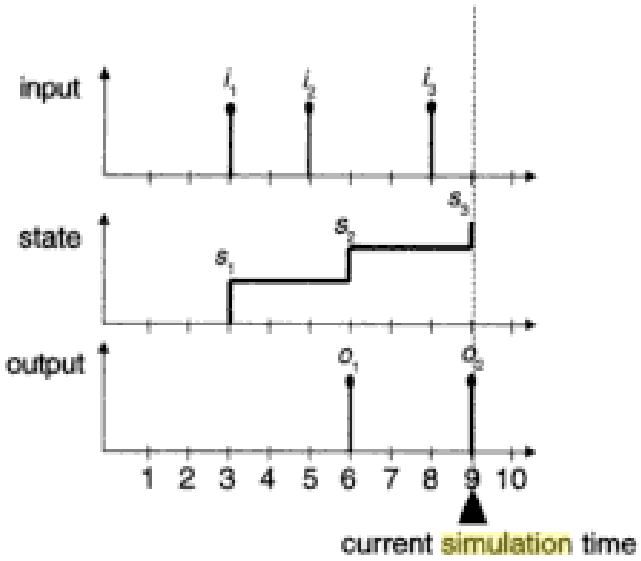


Optimistic Parallel DEVS simulator

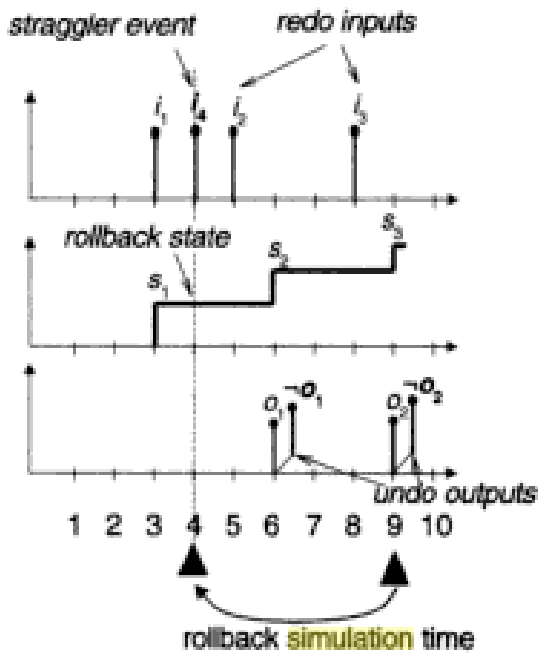
- Idea:
  - Process events, even if they might break overall causality.
  - Component may receive straggler event, with simulated time less than expected.
  - Component has to roll back.

Optimistic Parallel DEVS simulator

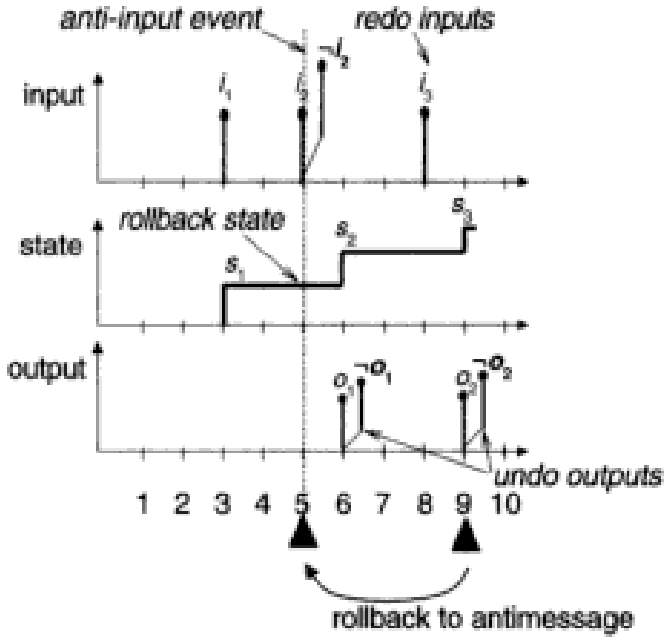
- Component "roll back" will have to:
  - set back the state to before the straggler event
  - Annihilate any outputs already sent with simulated events larger than the straggler event time.



(a) simulation situation with events processed until time 9



(b) straggler event occurred at time 4



(c) anti-input ¬*j2* received at time 5

Optimistic Parallel DEVS simulator

- Components have to store:
  - States: to restore them
  - Inputs: to redo them
  - Outputs: to annihilate them
- But eventually will run out of memory or disk space.

Optimistic Parallel DEVS simulator

- Fossil Collection:
  - Remove states, inputs, and outputs when it's safe to remove them.
  - When it's guaranteed that the component will not receive any event with time which causes a rollback.

Optimistic Parallel DEVS simulator

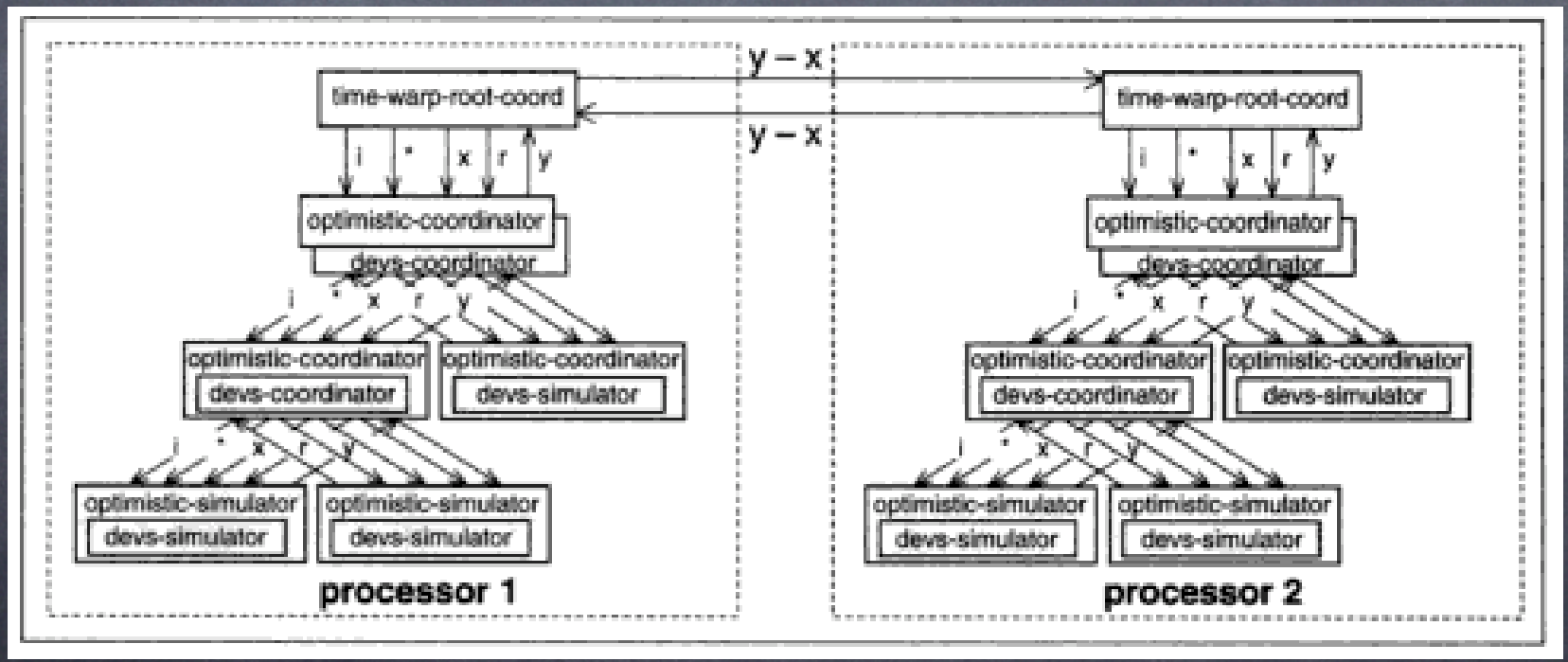
Optimistic Parallel DEVS simulator

- Global Virtual Time (GVT):
  - The time it's safe to remove old states.
  - Found as the minimum of :
    - all processors last executed time.
    - all the sent but not yet received events.

Optimistic Parallel DEVS simulator

- At initialization, the DEVS model is partitioned into distinct components where each component is assigned to be simulated at one processor.
- Each processor has a root coordinator.
- Each processor uses hierarchical scheduling with some additions.
- Time-warp scheduling across processors

Optimistic Parallel DEVS simulator



Optimistic Parallel DEVS simulator

- Three Types of objects:
  - Solver -> wrapped in optimistic solvers.
  - Coordinator -> wrapped in optimistic coordinator.
  - Root coordinator -> time-warp root coordinator at each processor.

Optimistic Parallel DEVS simulator

- Optimistic Atomic Solver:
  - Checkpoints states.
  - Forwards messages to the classic solver.
  - Roll back to a specific state when it needs to.

Optimistic Parallel DEVS simulator

- Optimistic Coordinator:
  - Take care of rollback messages.
    - Forwards only to sub simulators which ran ahead.
  - Forward other messages normally

Optimistic Parallel DEVS simulator

- Root coordinator
  - Main event loop:
    - simulating internal component, and updating GVT.
    - Interrupt function with inputs sent from influencer components.

Outstanding Issues

- Deadlock prevention at GVT calculation stage.
- Multithreading within PYRO.
- Initialization Issues. (startup)
- Simulation Trace.

Distributed Middleware

- Distributed Simulator can be run on a single multi core machine, but also on a cluster.
- The Distribution involves:
  - Create and deploy root coordinators on specified machines.
  - A middle ware to support messaging between components.
  - Dealing with fault tolerance issues.

Fault Tolerance

- Fault tolerance:
  - Fault model is machine crashes.
  - Similar to handling "roll backs" in time-warp.
  - State has to be persisted, or replicated.
  - An overall coordinator is needed.

Conclusion

- Distributed DEVS simulator highly desirable.
- Several techniques for distributing simulation protocol.
- Time warp mechanism increases performance, when the partition is well chosen.
- Extending PYDEVS simulator to run in time-warp over a cluster.

References

- by Bernard P. Zeigler, Tag Gon Kim, Herbert Praehofer.
- Eugene Syriani, Amr Al Mallah. modeling, simulation and implementation of a distributed DEVS simulator

C'EST TOUT!