PDES† × HPC*

12 Dec 2006

Kalyan S. Perumalla, Ph.D.

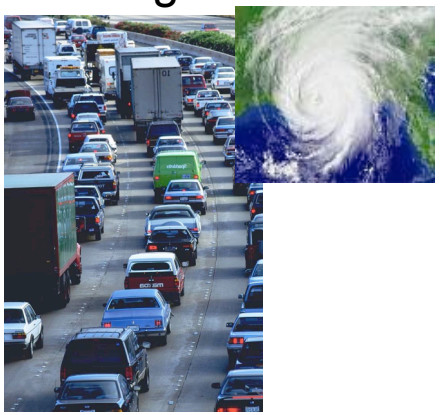
Senior Researcher, ORNL
Adjunct Faculty, Georgia Tech

† Parallel/Distributed Discrete Event Simulation

* High Performance Computing

PDES: Selected Application Areas

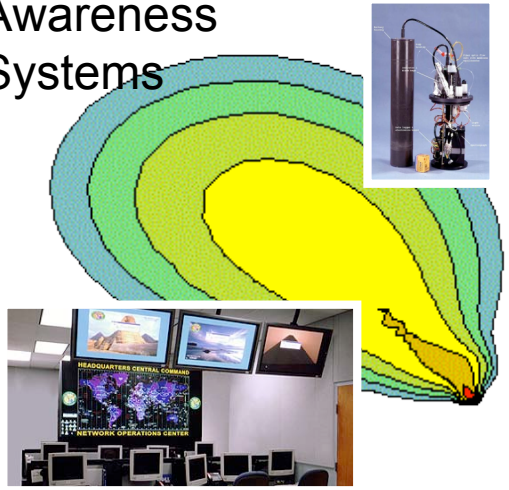
Emergencies



Global & Local Events

Current & Future Defense Systems

Protection & Awareness Systems

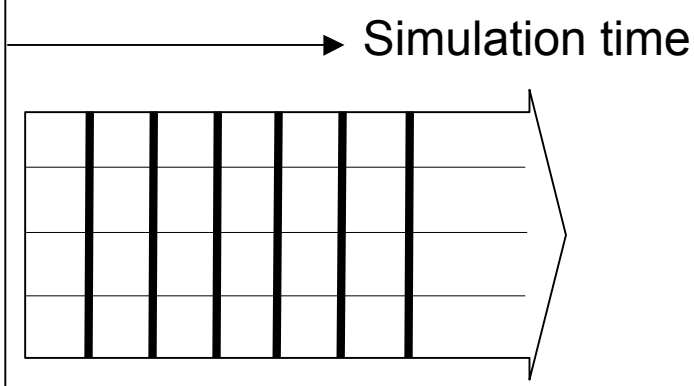


PDES: Selected Applications

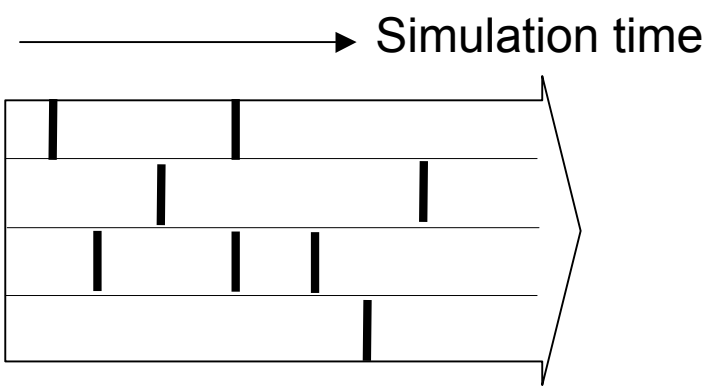
- Network simulation
  - Internet protocols, Security , P2P designs, …
- Traffic simulation
  - Emergency planning/response, Environmental policy analysis, Urban planning, …
- Social dynamics simulation
  - Operations planning, Foreign policy, Marketing, …
- Sensor simulations
  - Wide area monitoring, Situational awareness, Border surveillance, …
- Organization simulations
  - Command & control, Business processes, …

Time Stepped vs. Discrete Event Simulation

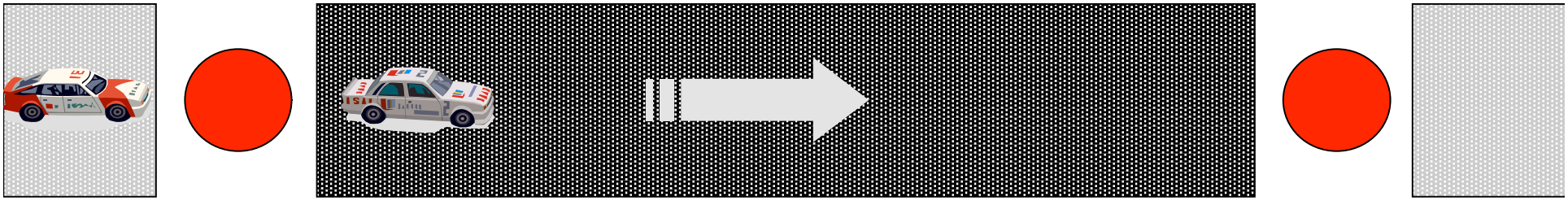
- **Time-Stepped (TS)**
- All entities are paced with time increment **dt**
- Entities exchange state updates via messages



- **Discrete-Event (DES)**
- State updates are scheduled at different times in the future
- Entities exchange **events** for state updates
- Events are executed in timestamp order



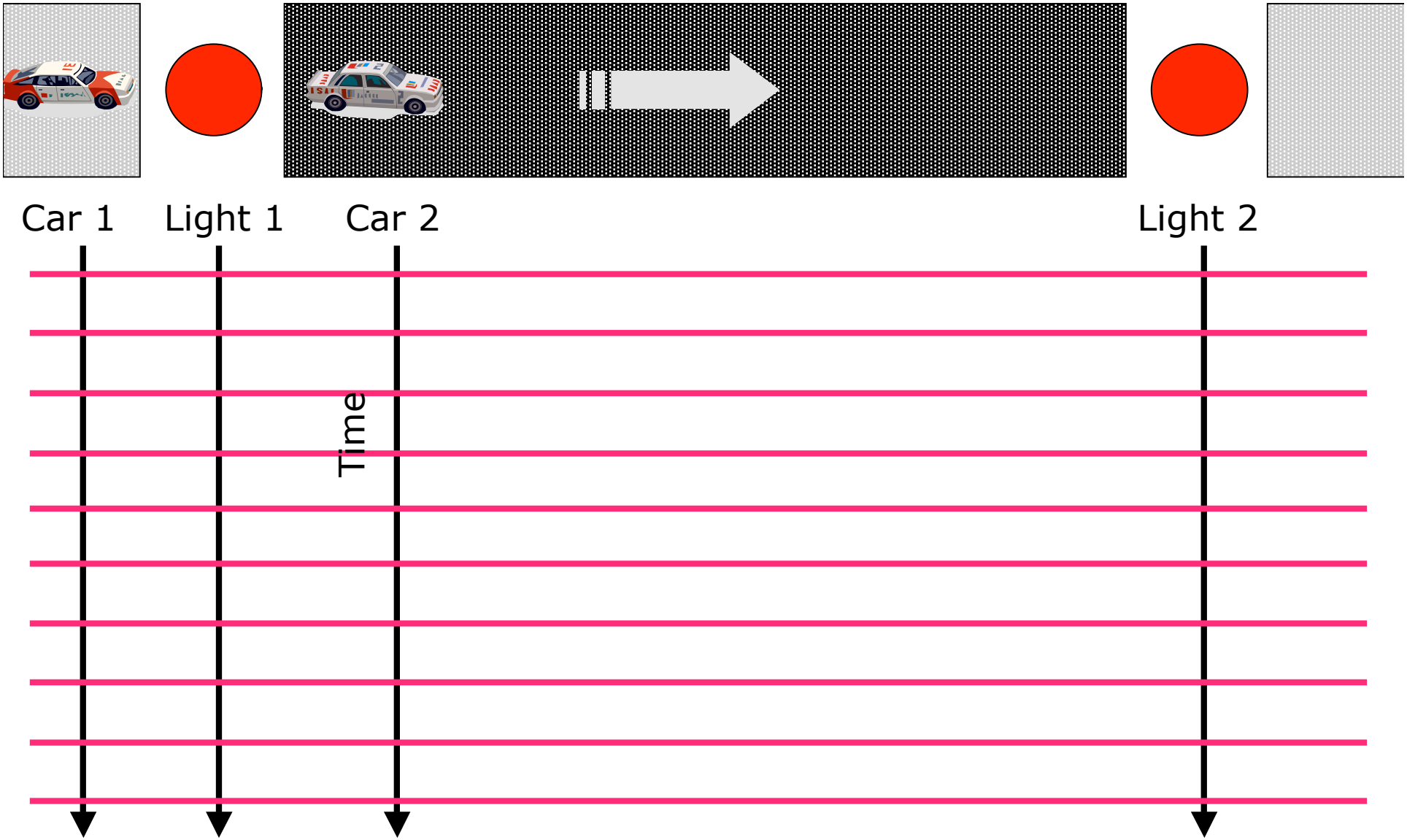
A Simple Example



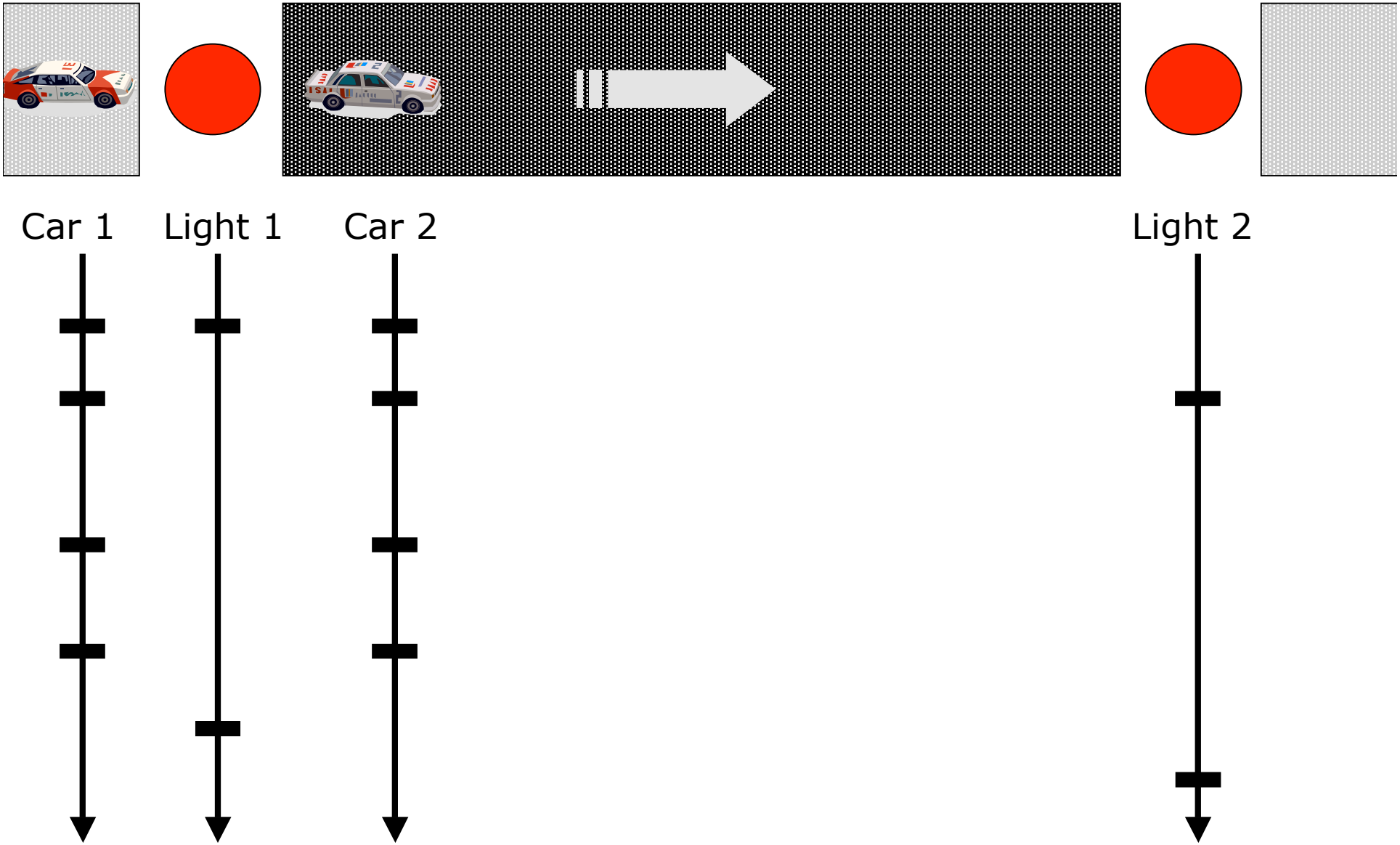
How is this executed in time-stepped models?

How is this executed in discrete event models?

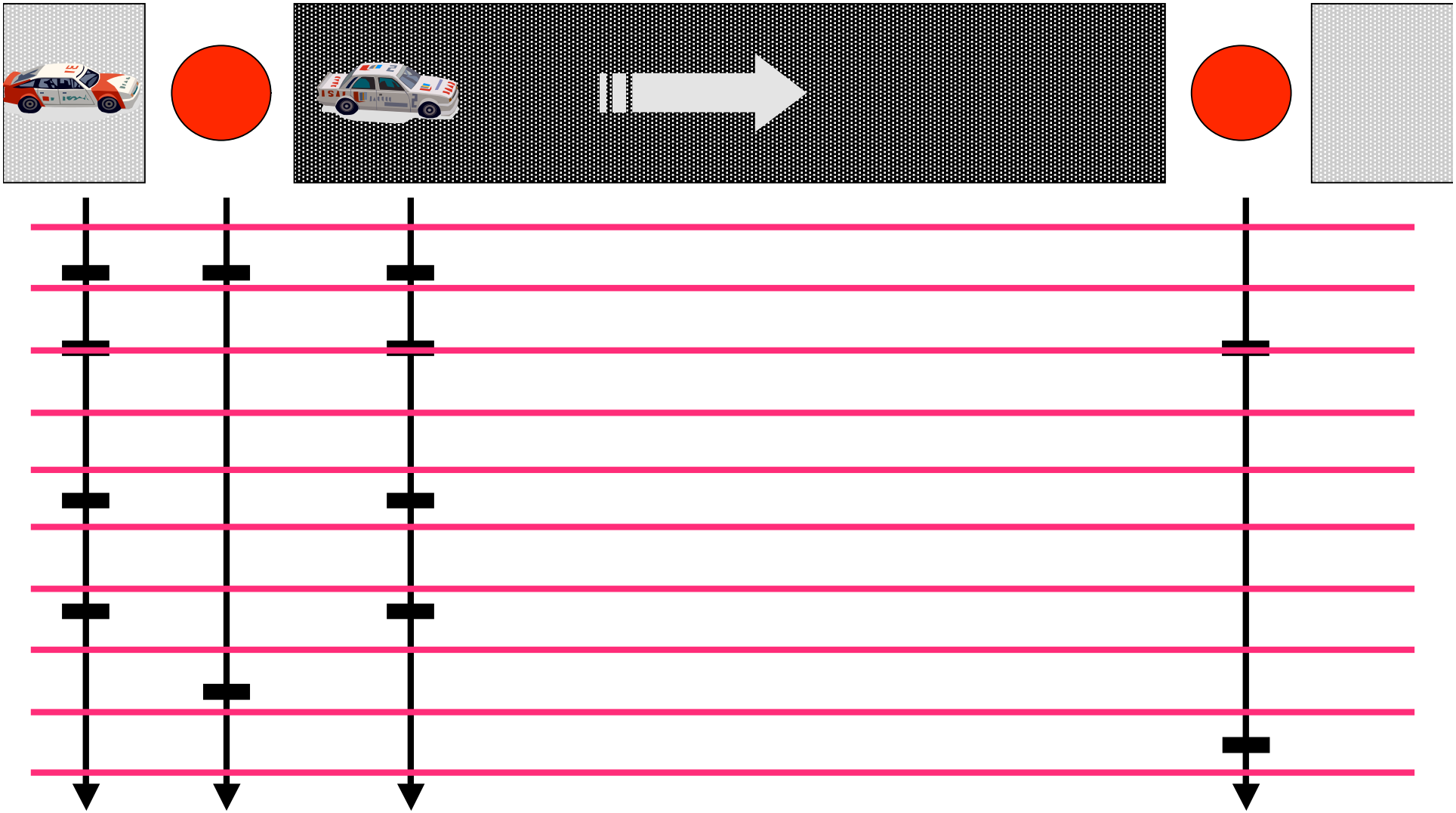
Time Stepped Model – Example



Discrete Event Model – Example



Time Stepped vs. Discrete

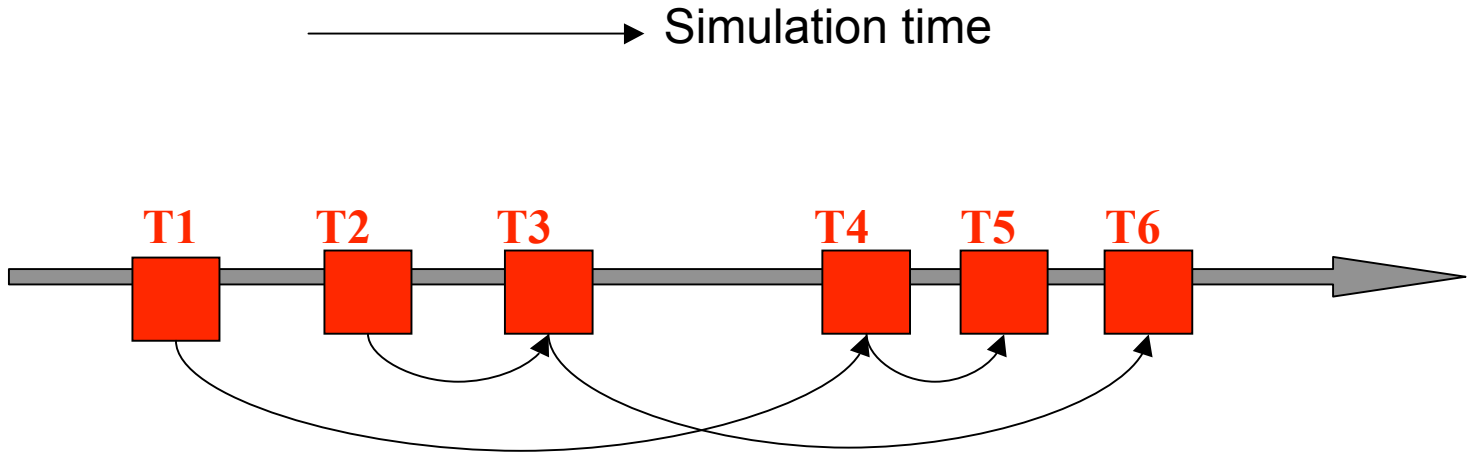


Sequential & Parallel DES

Sequential

1 processor

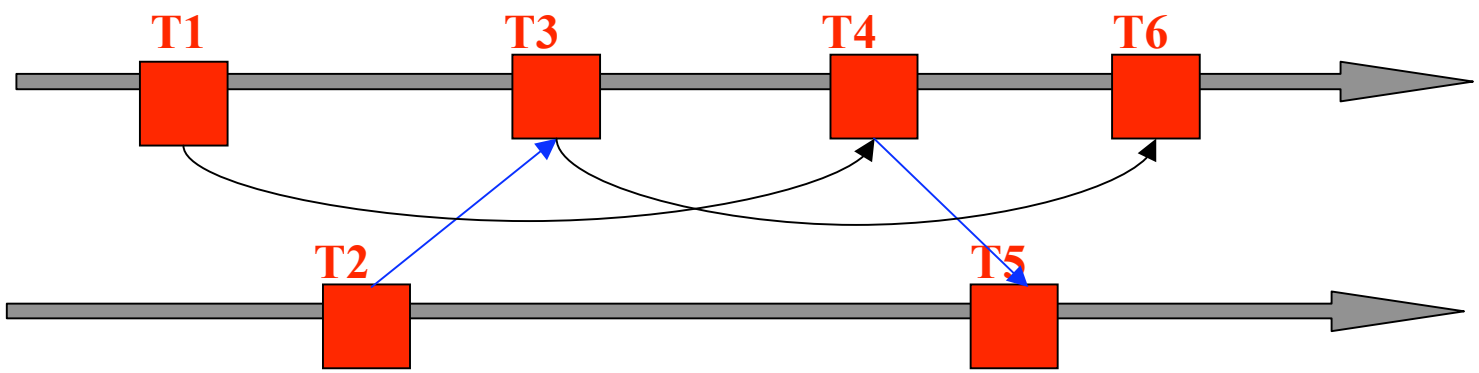
Example:



Parallel

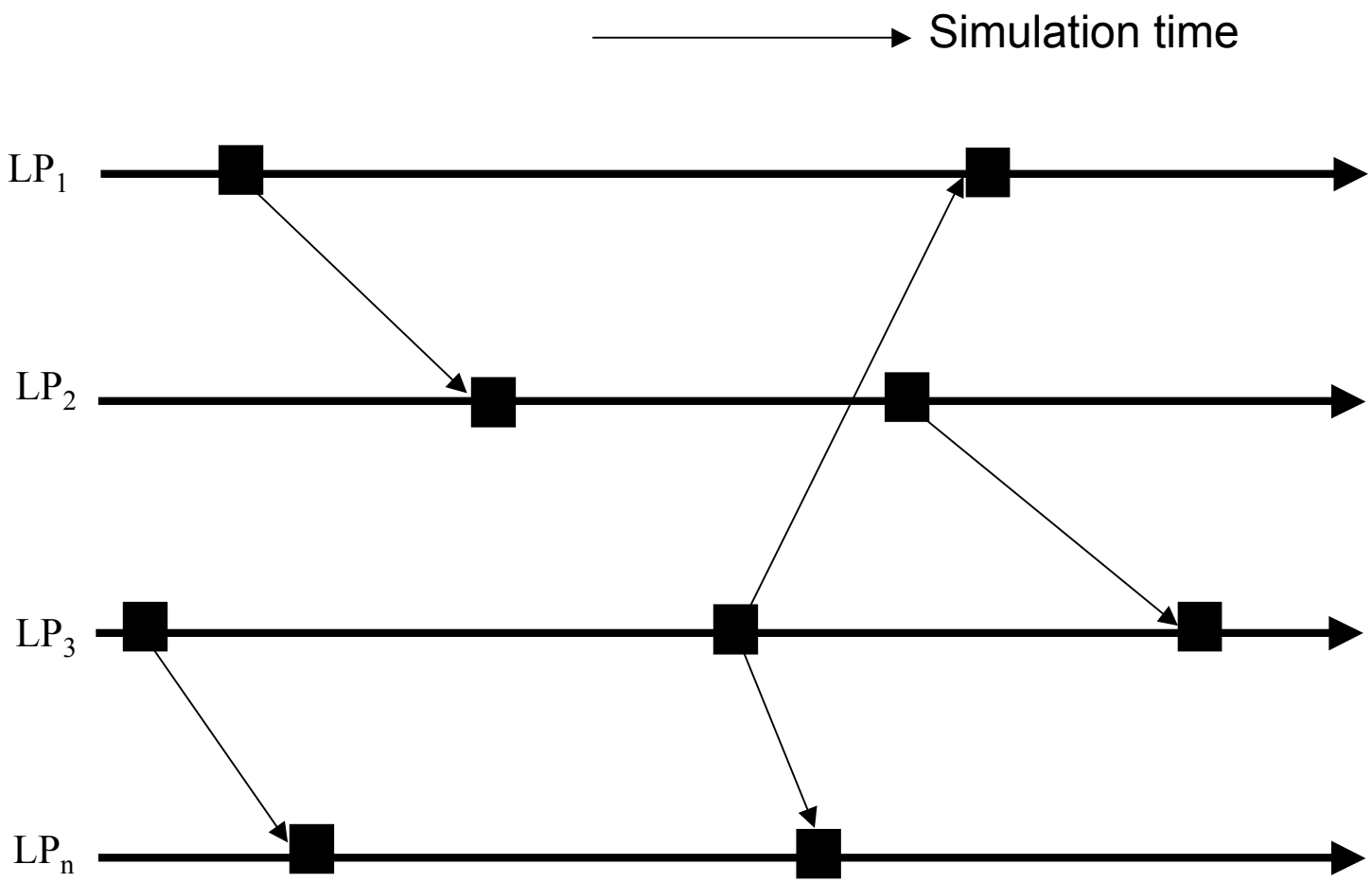
n>1 processors

Example:
2 processors



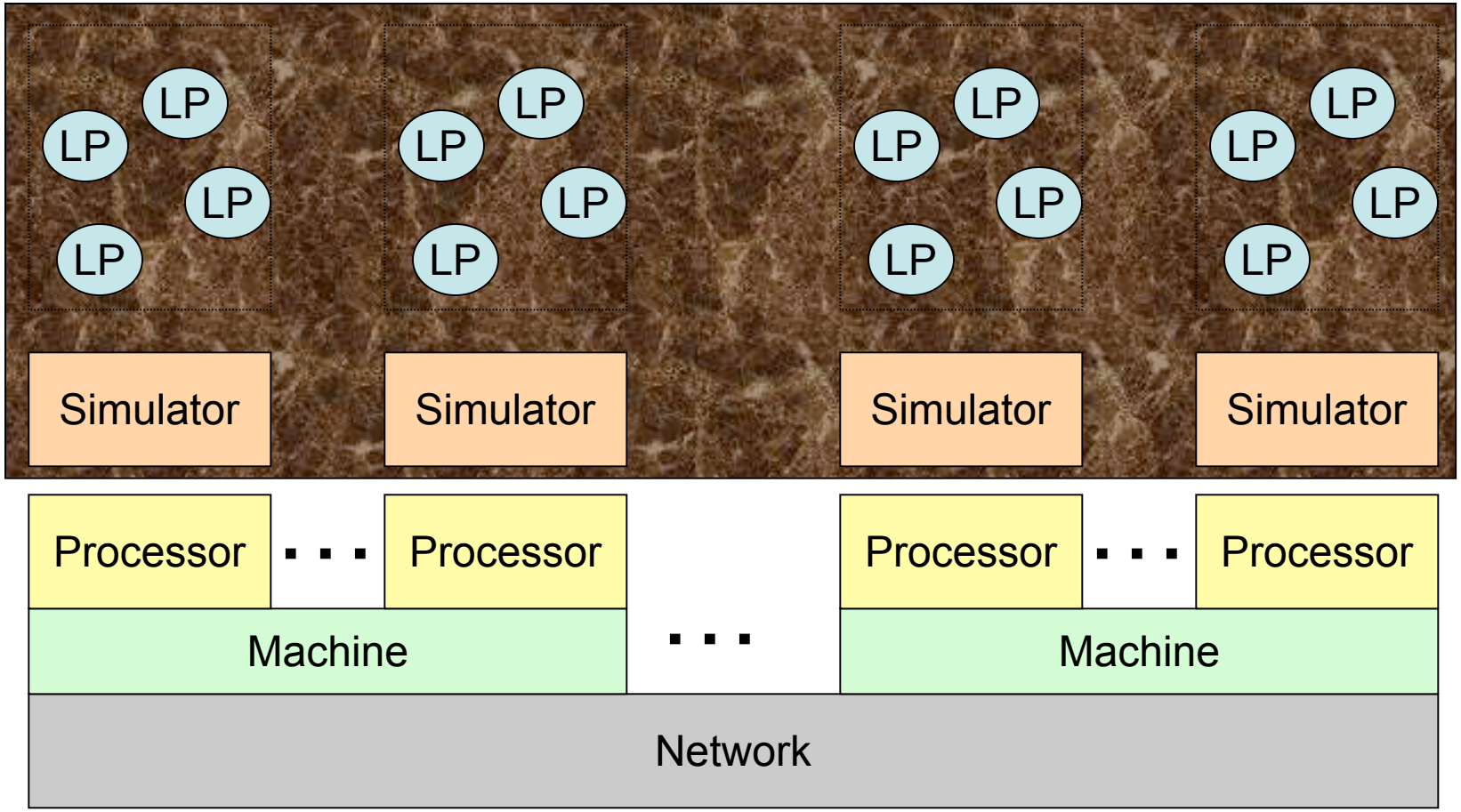
PDES Execution View

LP = Logical Process



PDES: Typical System Architecture

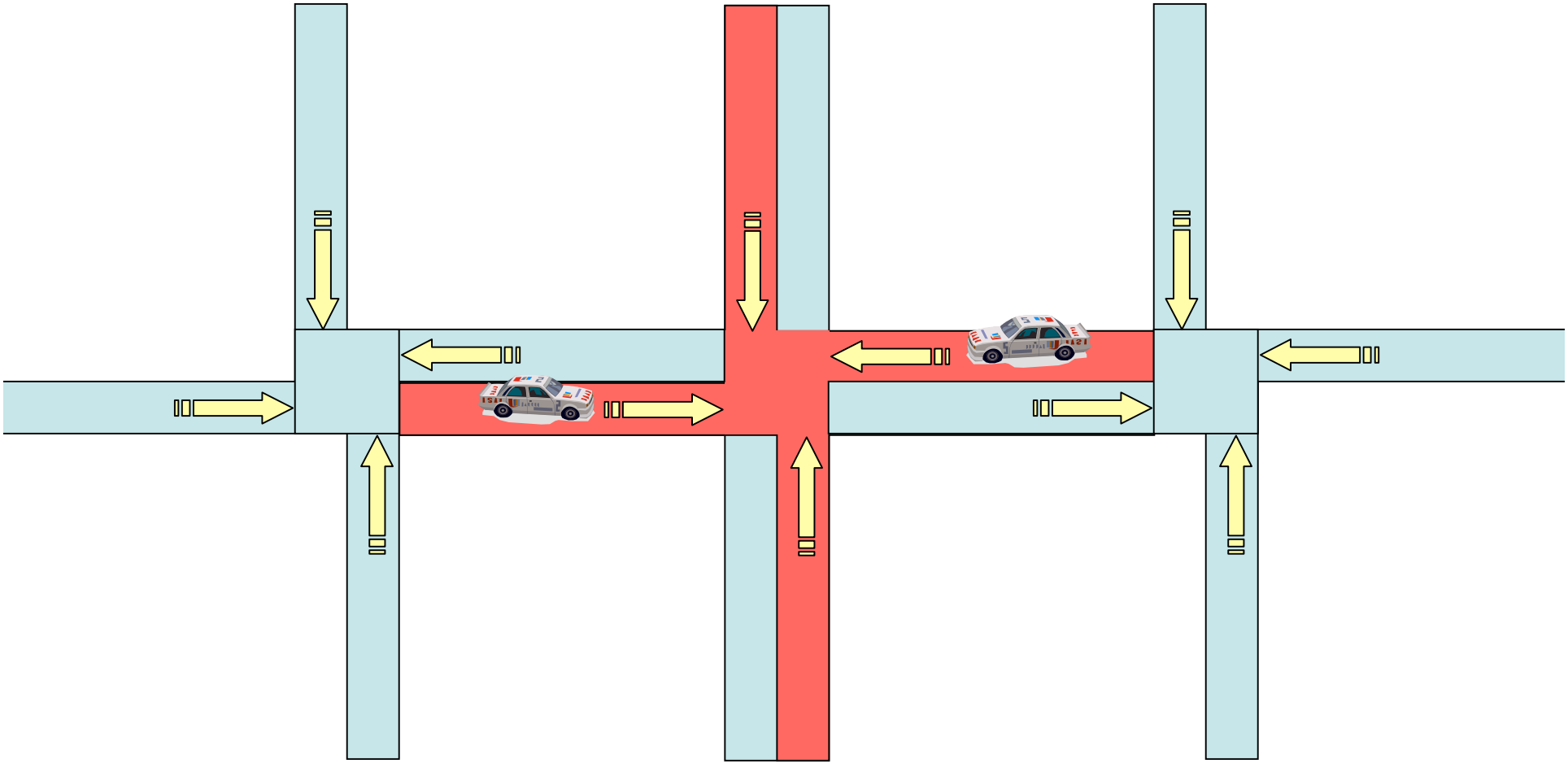
LP=Logical Process with its own timeline



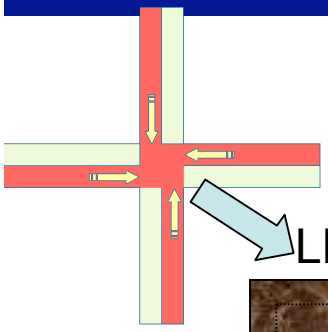
PDES: Typical Application Characteristics

- Global time synchronization
  - Total time-stamped ordering of events
  - Paramount for accuracy
- Fast synchronization
  - Scalable, application-independent, time-advance mechanisms
  - Critical for real-time and as-fast-as-possible execution
- Support for fine-grained events
  - Minimal overhead relative to event processing times
  - Application computation is typically only 5μs to 50μs per event
- Conservative, optimistic & mixed modes
  - Need support for the principal synchronization approaches
  - Useful to choose mode on per-entity basis at initialization
  - Desirable to vary mode dynamically during simulation
- General-purpose API
  - Reusable across multiple applications
  - Accommodate multiple techniques
    - lookahead, state saving, reverse computation, multicast, etc.

PDES Example: Transportation Simulations

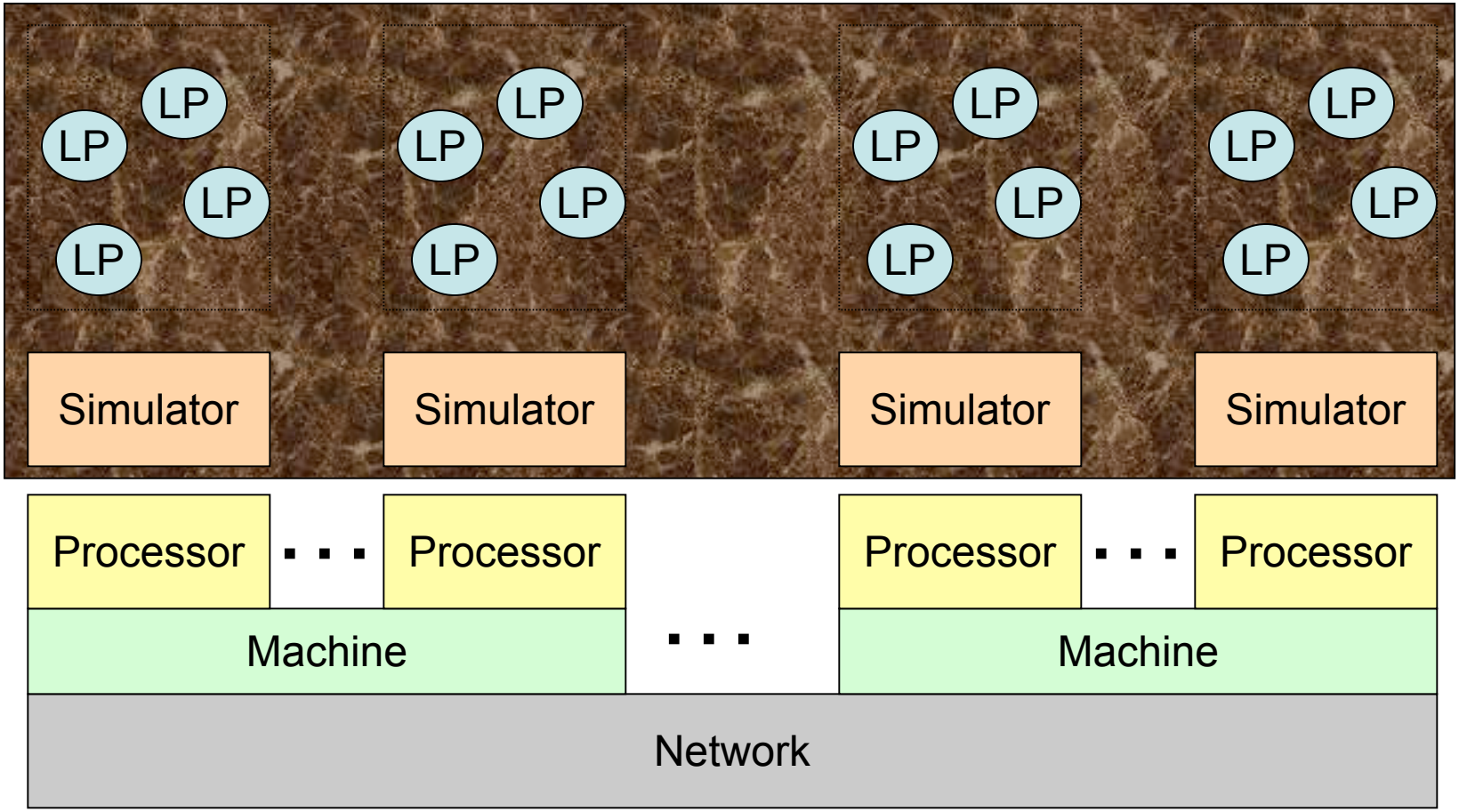


PDES Example (continued)



One intersection per LP

LP=Logical Process with its own timeline



Motivation

Government agencies' needs

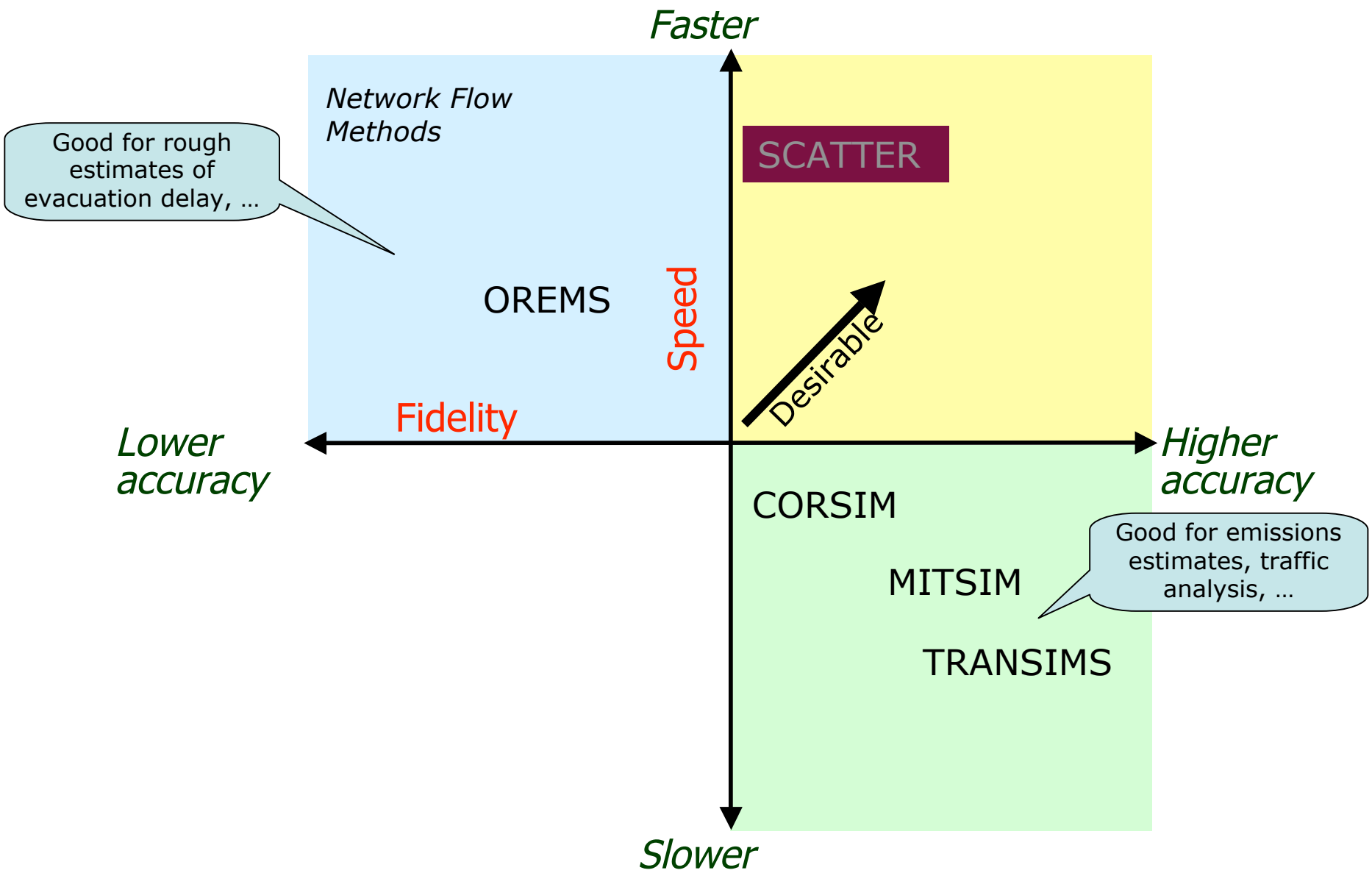
- Emergencies: e.g.
  - what's a good lower bound for evacuation time?
  - better to shelter in place or evacuate?
- Energy: e.g.
  - what's amount of fuel consumed?
  - how much fuel must be stocked and where?
- Policy: e.g.
  - Formulation, analysis, training
  - Pollution, emissions, environmental concerns
- …

Need fast, detailed simulations

Targets & Capabilities

- Target: Regional scale
  (multiple states; e.g. Southeast)
  - 10^6-10^7 intersections
- Current tool capabilities
  - At most 10^4 intersections
- Speed
  - Faster than real-time is very useful

Solution Space

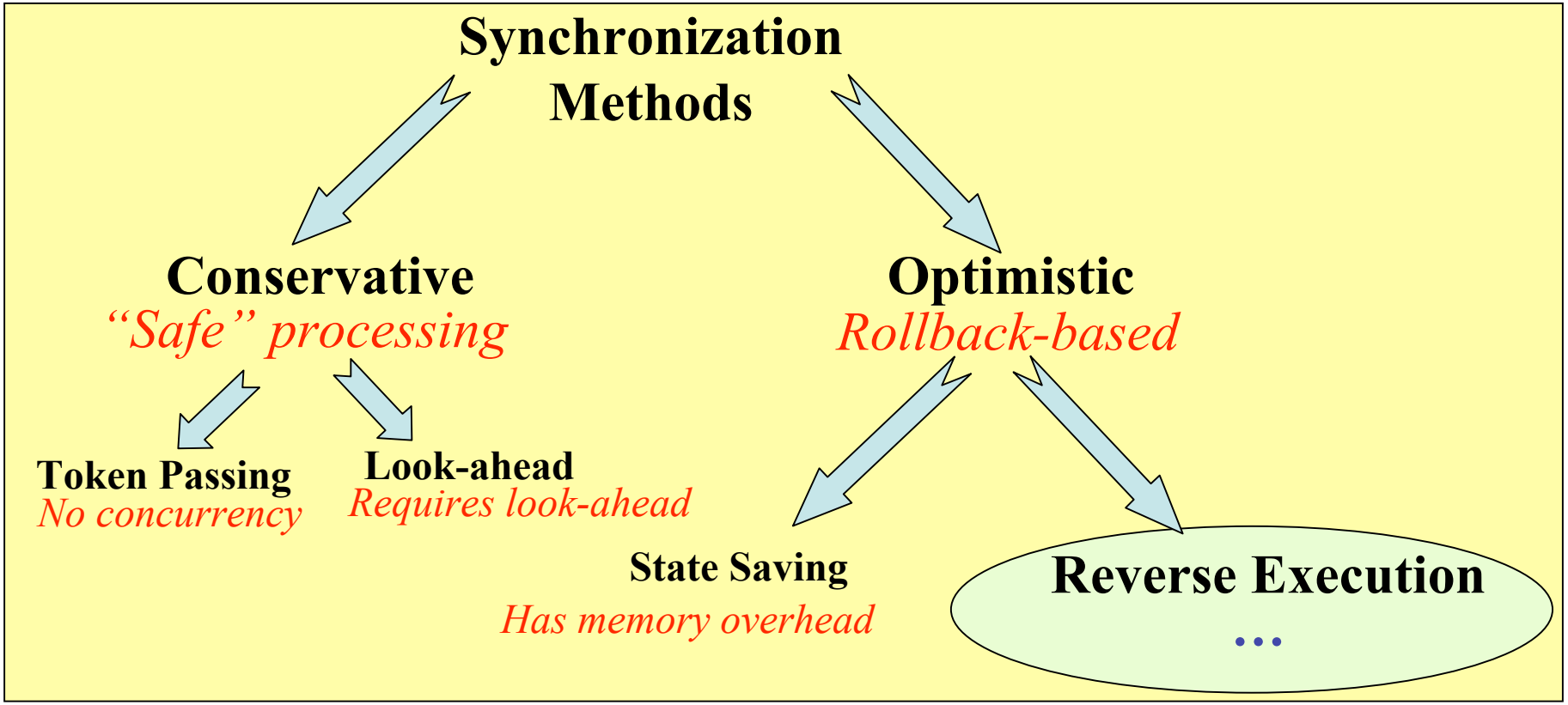


SCATTER Approach

- Parallel execution
  - vs. Sequential
- DES models
  - vs. Time-Stepped
- Important behaviors
  - Kinetics *plus* non-kinetics
- Scalability to HPC
  - 10^1-10^3 CPUs

Parallel Execution Techniques

Goal: Ensure global timestamp-ordered processing.
=> Synchronization among simulators required.



Reverse Computation (RC)

Problem: Efficient rollback for optimistic simulation

- Traditional Approach
- *State saving*
- Undo by saving and restoring *e.g*

```
{save(x);x=x+1} →
{restore(x)}    ←
```

- Disadvantages
- Large state memory size
- Memory copying overheads
- Poor match for large-scale, fine-grained applications.

- New Alternative
- *Reverse Computation*
- Undo by executing in reverse*e.g.*

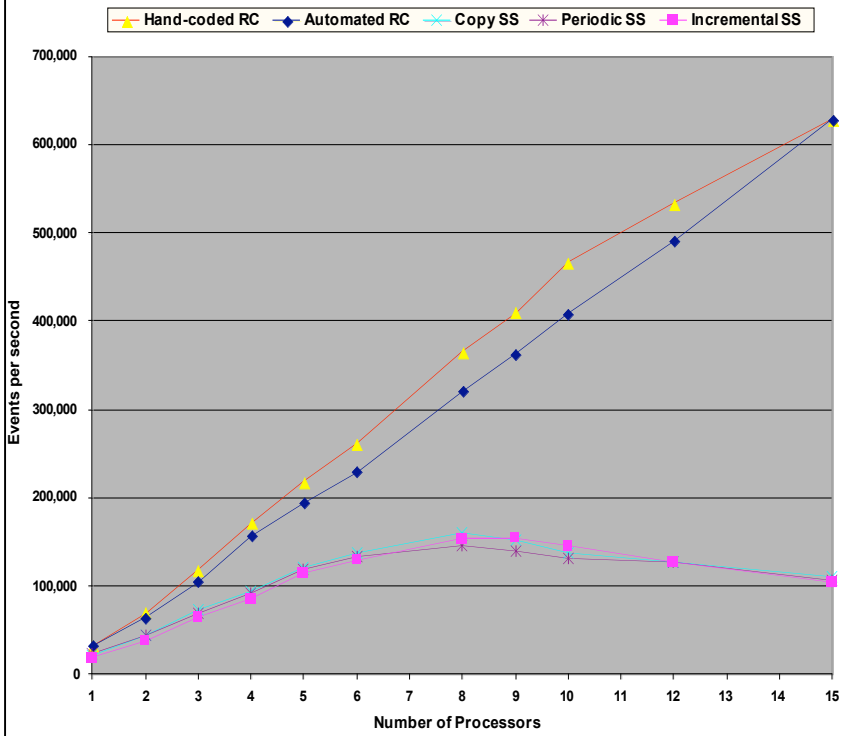
```
{x=x+1} →
{x=x-1} ←
```

- Advantages
- Reduced state memory size
- Reduced overheads; moved from forward to reverse
- Excellent match for large, high-performance simulations
- Can be automated.

RC Features & Performance

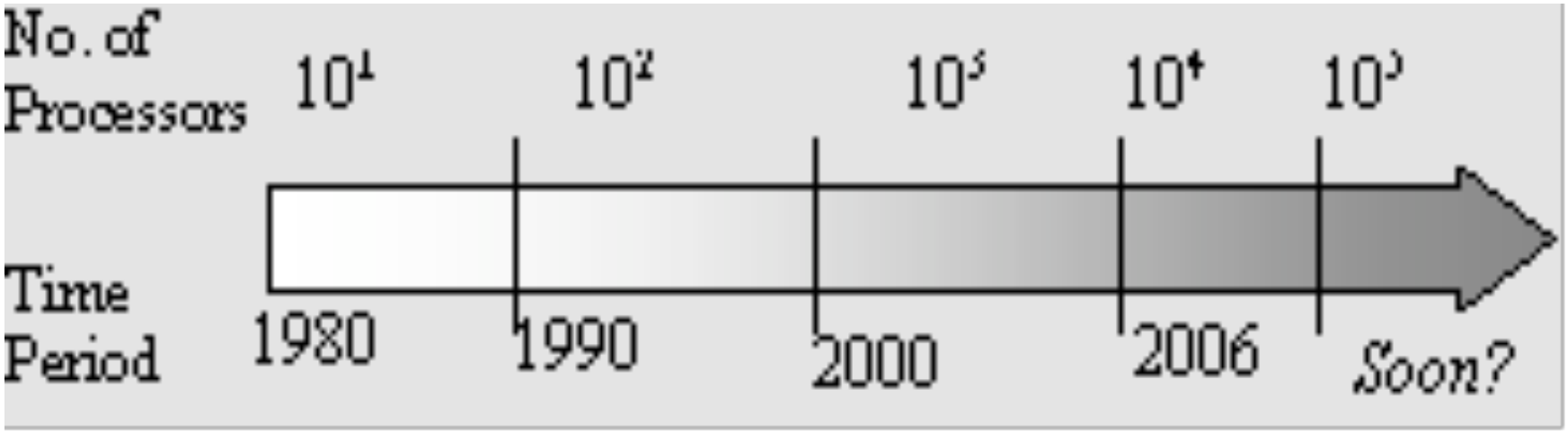
- Constructive operation =>
  zero state for reversibility (*e.g.* **x++**)
- Destructive operation =>
  state needs to be saved (*e.g.* **x=y**)
- Predominantly constructive operations =>
  reduced state size
- Queueing network models contain many constructive operations
  - random number generation (reversible RNGs)
  - queue handling (swap, shift, enqueue/dequeue, …)
  - statistics collection (increment, decrement, …)

Performance Gains

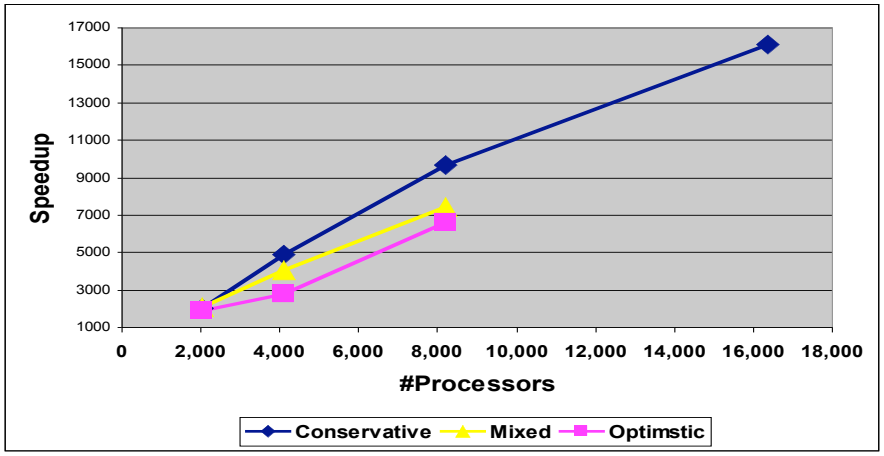


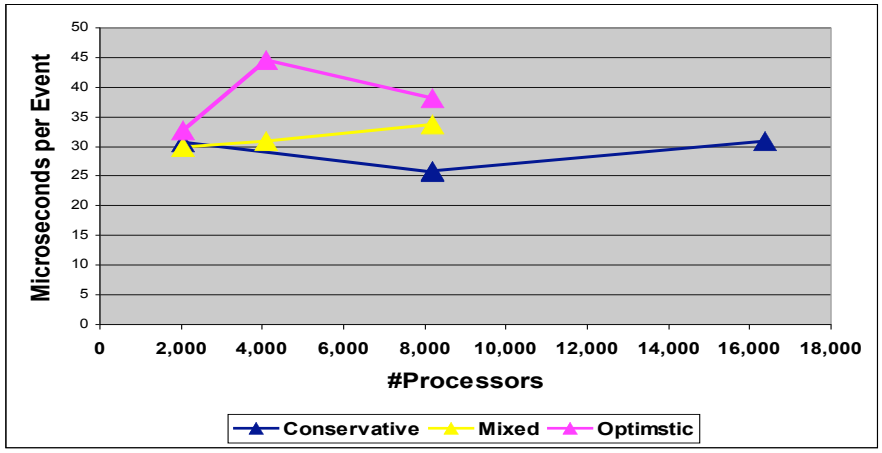
PADS Engine Scalability

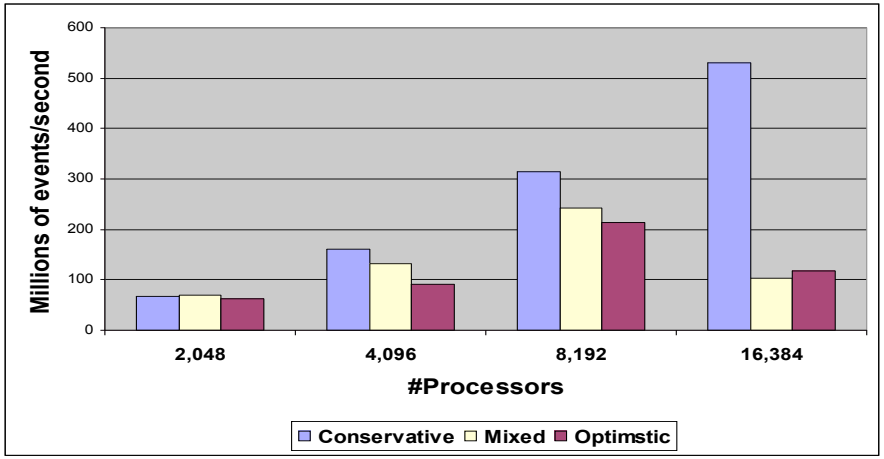
- Coming of age lately
- Time Warp & Mixed Mode possible now on 10^3-10^4 processors!
  - Very recent (10^3 in last 2 years, 10^4 last month)

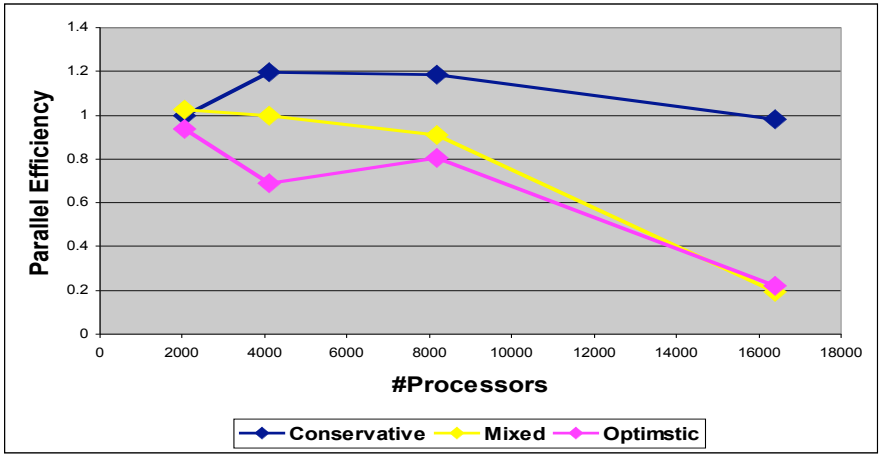


μsik Scaled to over 10^4 Processors



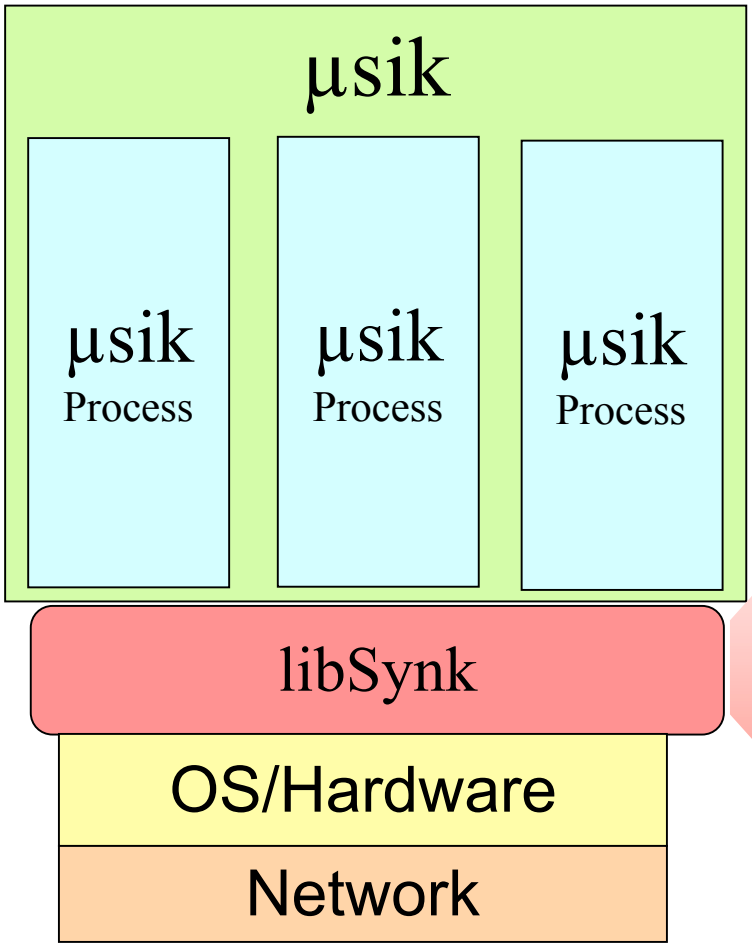


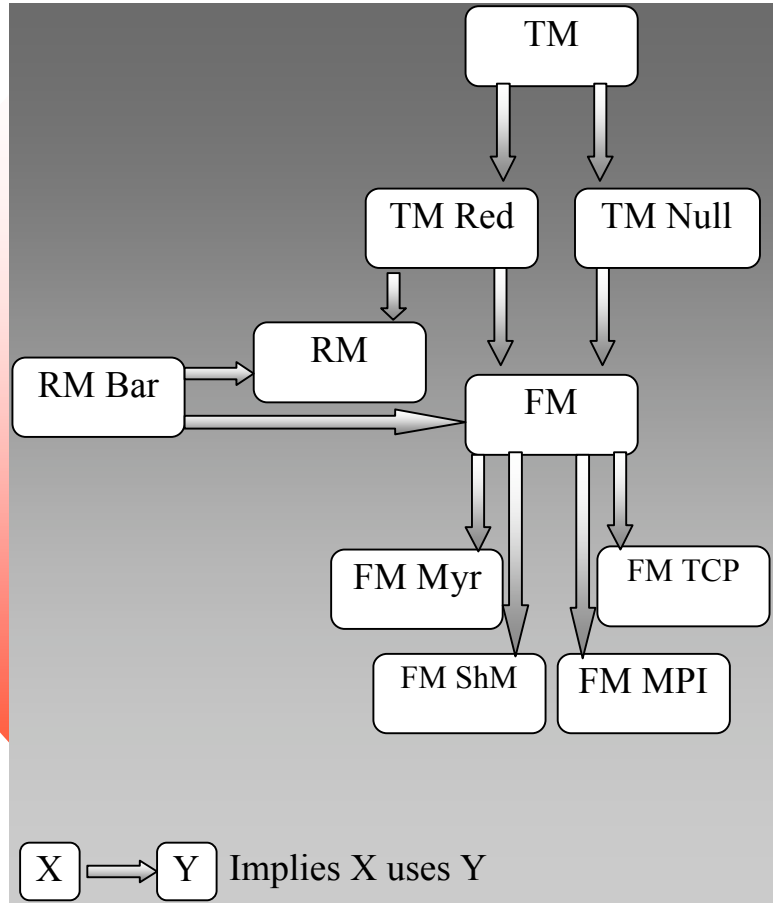




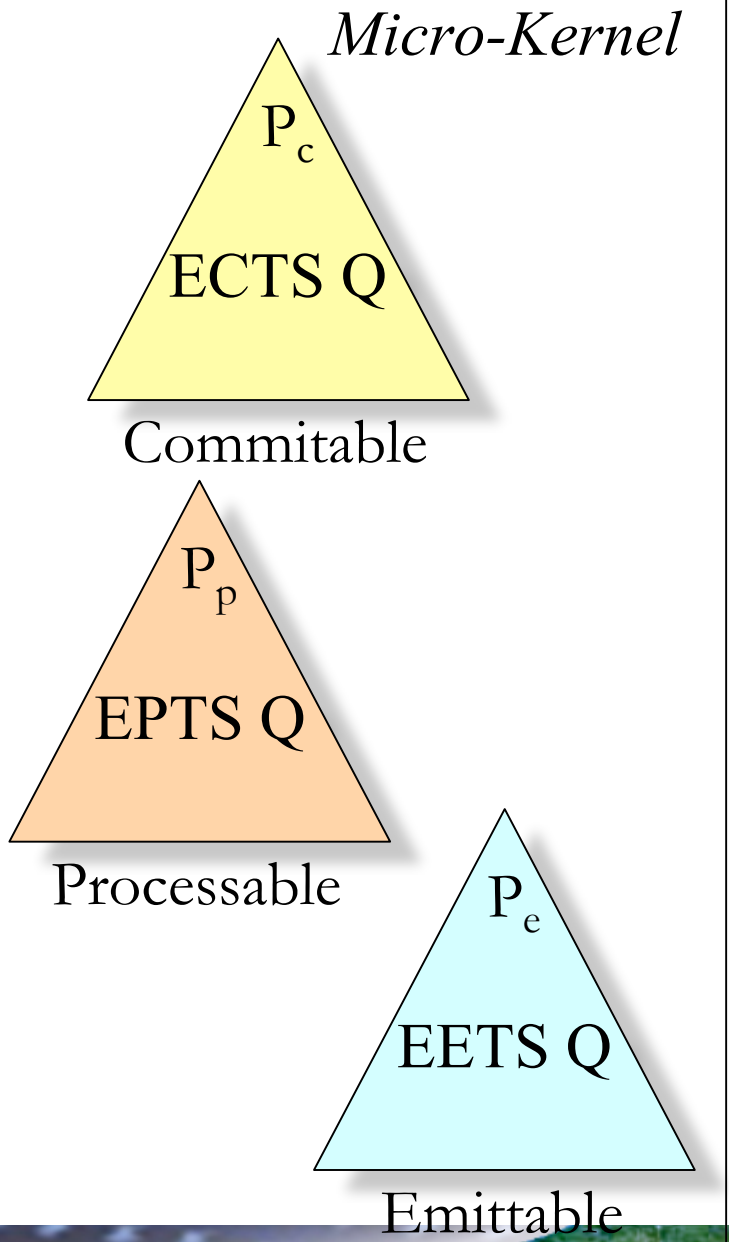
- Some recent results of **fine-grained** PDES benchmark
  - On Blue Gene Watson (BGW) at IBM TJ Watson Research Center
  - Well-known PHOLD benchmark, with 1 million logical processes, 10 million pucks
- The largest and fastest scalability results in PDES recorded to date

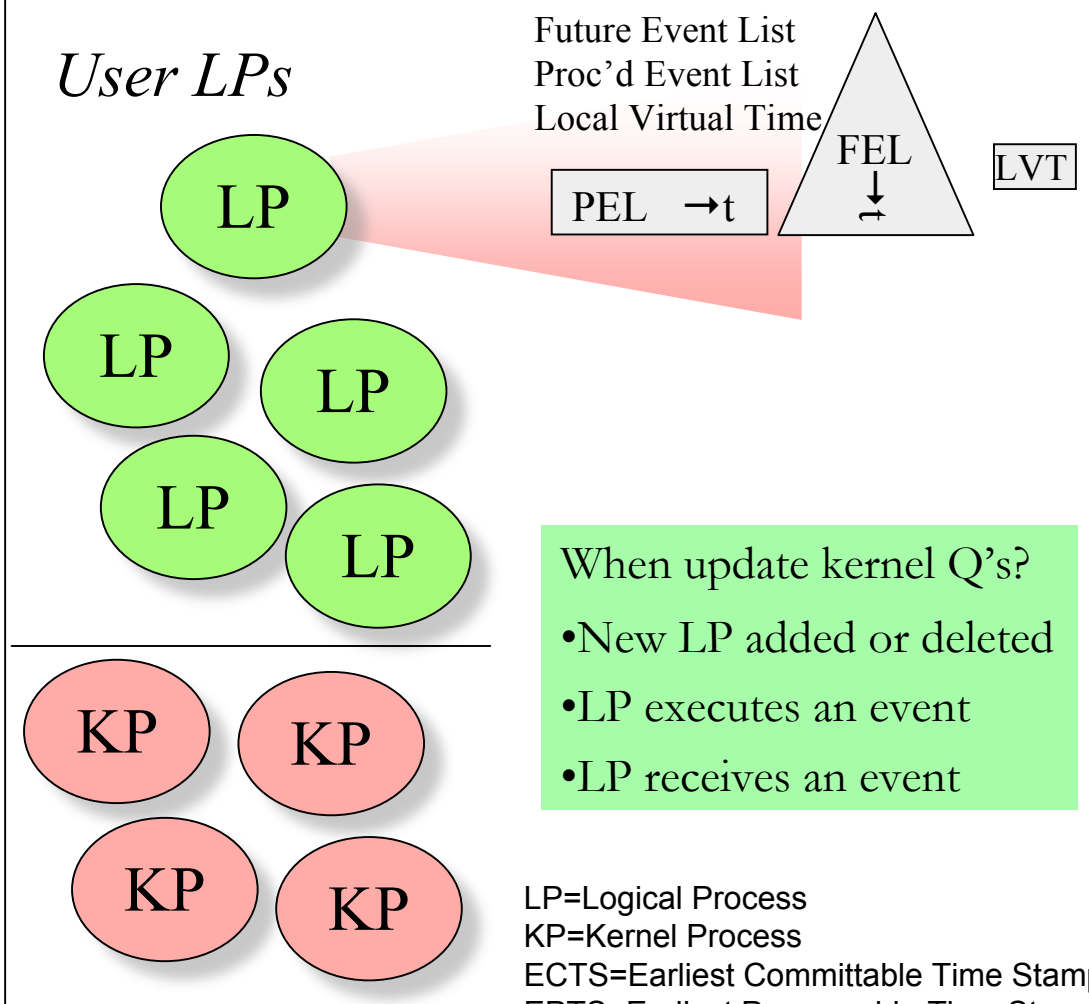
PDES Kernel Software to Build On





µsik Micro-Kernel Internals





When update kernel Q's?

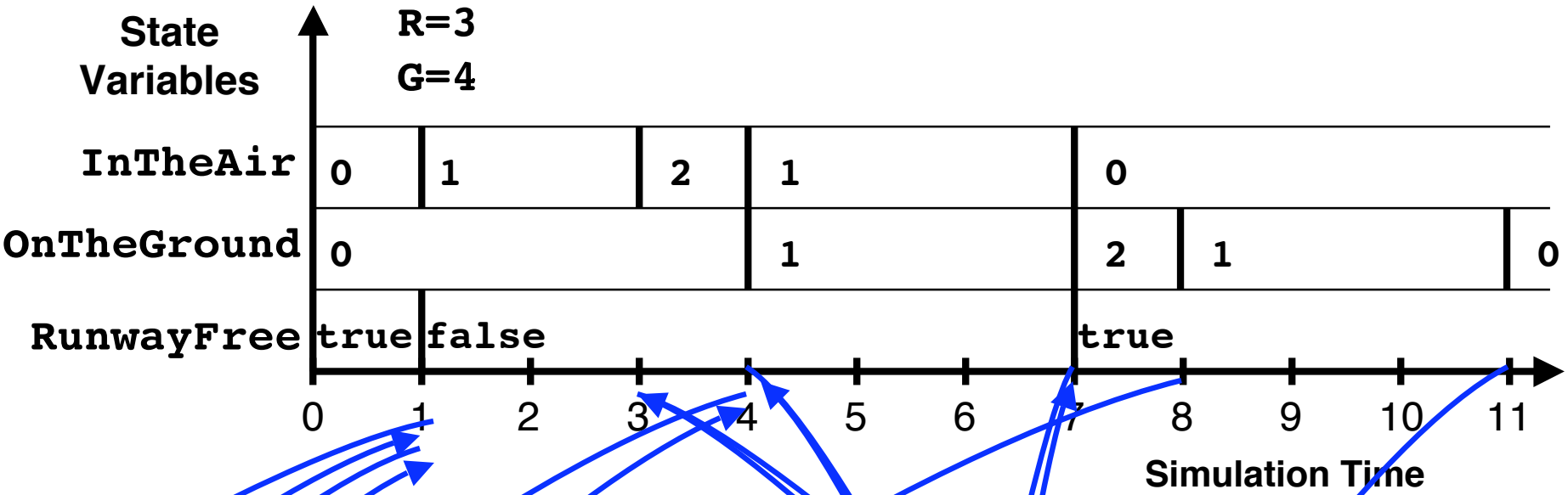
- New LP added or deleted
- LP executes an event
- LP receives an event

LP=Logical Process
KP=Kernel Process
ECTS=Earliest Committable Time Stamp
EPTS=Earliest Processable Time Stamp
EETS=Earliest Emittable Time Stamp
PEL=Processed Event List
FEL=Future Event List
LVT=Local Virtual Time

Alternative Kernels

- iSSF/DaSSF
  - Shared-memory only; distributed memory under development; mostly used for internet simulations only
- ROSS
  - Shared-memory; reverse-computation based; scalability $>10^2$ CPUs unknown
- DSIM
  - Recent kernel; distributed memory; Time Warp only
- Charm++
  - Raw event rate unknown; performance relative to other kernels unknown
- PARSEC
  - One of early PDES kernels; almost obsolete; no more development; scalability $>10^2$ CPUs unknown
- GTW
  - One of early PDES kernels; very well known, but obsolete; shared memory only
- FDK, HLA RTIs (Mak, Mitre, …)
  - DoD High Level Architecture runtime infrastructures; coarse-grained, relatively heavyweight; few scalable implementations for fine-grained parallelism
- Commercial sequential packages & shared-memory parallel simulation packages

Multi-Threading in PDES Models: Example




```
Flight 1
1 InTheAir := InTheAir+1;
2 WaitUntil (RunwayFree);
3 RunwayFree := FALSE;
4 AdvanceTime(R);
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround:=OnTheGround+1;
8 AdvanceTime(G);
9 OnTheGround:=OnTheGround-1;
```

```
Flight 2
1 InTheAir := InTheAir+1;
2 WaitUntil (RunwayFree);
3 RunwayFree := FALSE;
4 AdvanceTime(R);
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround:=OnTheGround+1;
8 AdvanceTime(G);
9 OnTheGround:=OnTheGround-1;
```