

**A Unified Modeling Methodology
for Performance Evaluation Of
Distributed Discrete Event Simulation Mechanisms**

Bruno R. Preiss

Wayne M. Loucks

Department of Electrical Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

V. Carl Hamacher

Departments of Electrical Engineering and Computer Science
University of Toronto
Toronto, Ontario, Canada, M5S 1A4

This research was funded by the Information Technology Research Centre of the Province of Ontario (Canada) and by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grants A5192, A6840 and OGP0036635.

Introduction and Motivation

- a single specification method for discrete event simulation (DES)
- independent of the underlying execution mechanism
- change execution mechanisms without altering simulation models
- compare performance of different execution mechanisms

Execution Mechanisms

- traditional DES
(event-list-driven)
- distributed DES mechanism using multiple, synchronized event lists
- Chandy-Misra distributed DES
(without deadlock detection and recovery)
- Virtual-time-based distributed DES
(using the time warp mechanism)

Modeling Method

- real world system

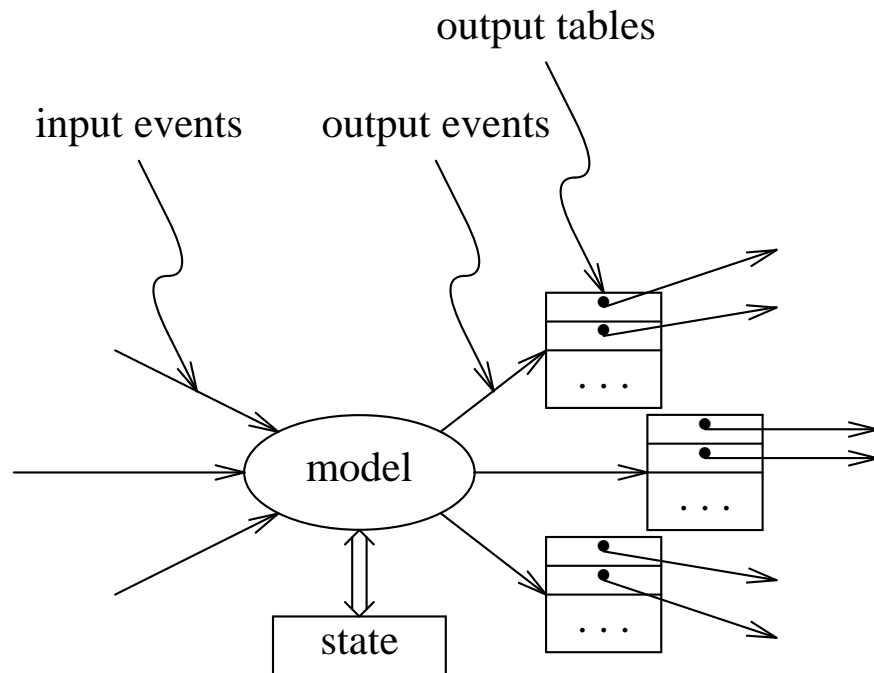
modeled as:

- static network of physical processes (PPs)
- periodic exchange of information (events)

simulated by:

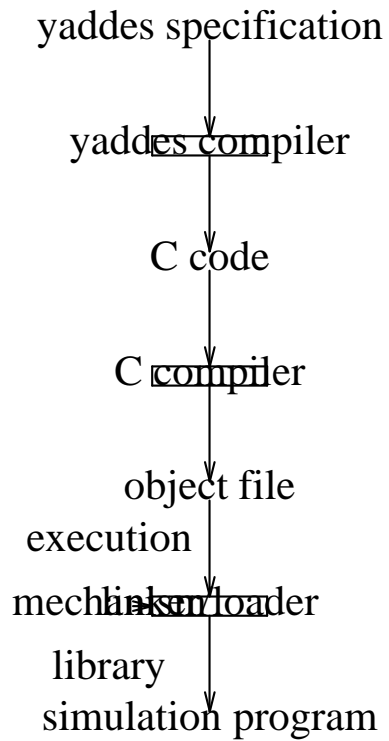
- collection of logical processes (LPs)
- exchange of time-stamped messages
- LPs are “general state machines”
- specification of LP is called a *model*
 - state transition table *and*
 - output event specifications *for each*
 - input event combination

A Yaddes Logical Process



The Yaddes Language

- preprocessor for the C programming language



Language components

- model specifications — describe generic LPs
- process specifications — instantiate LPs
- connection specifications — link LPs

Model Specification Example

```
model TwoInputNand
  inputs in0, in1
  outputs out
  state { . . . }
  initial state { . . . }
  action initial { . . . }
  action in0 { . . . }
  action in1 { . . . }
  action in0, in1 { . . . }
end model
```

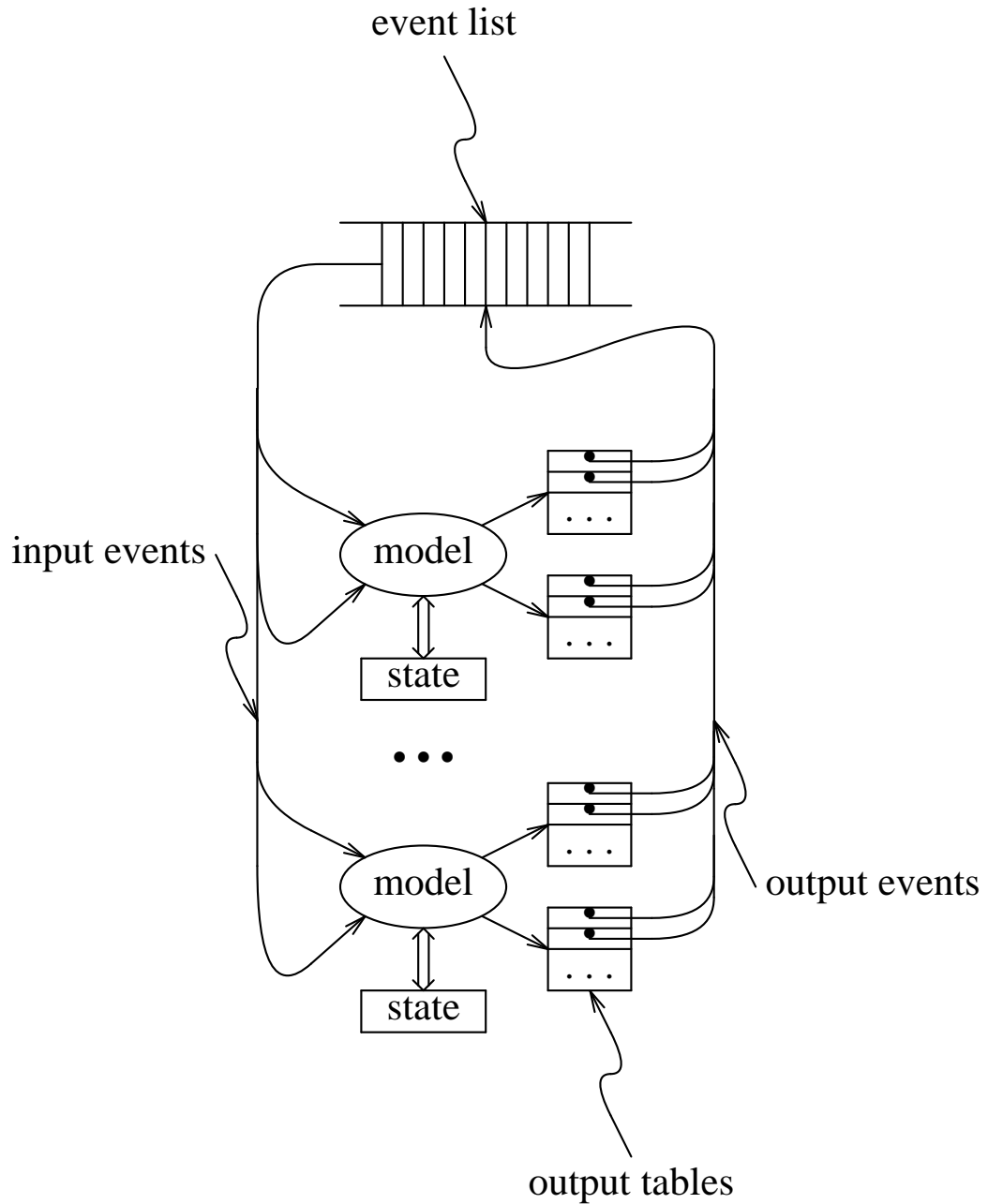
Process Specification Example

```
process X on 0 : ReadFromFile  
process Y on 0 : ReadFromFile  
process Gate0 on 0 : TwoInputNand  
process Gate1 on 0 : TwoInputNand  
process Z on 0 : WriteToFile
```

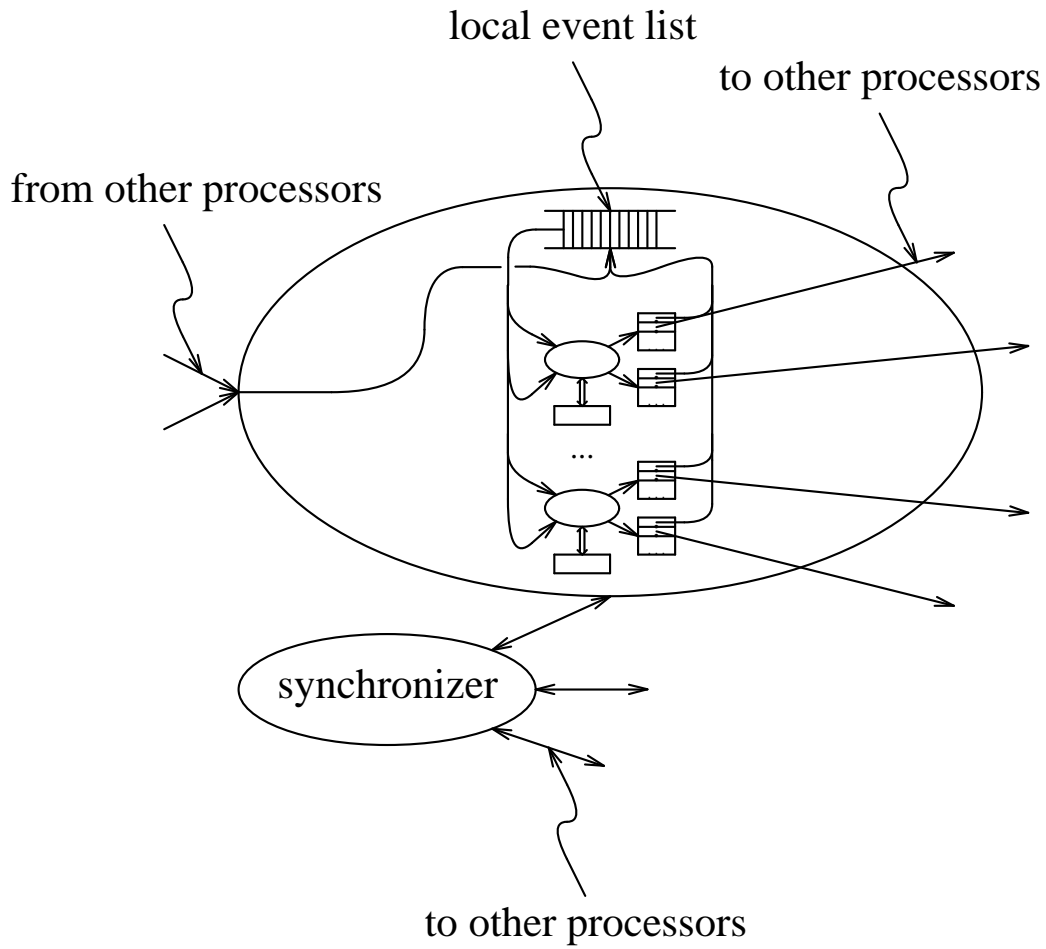
Connection Specification Example

```
connect X.out to Gate0.in0  
connect Y.out to Gate1.in1  
connect Gate0.out to Gate1.in0, Z.in  
connect Gate1.out to Gate0.in1
```

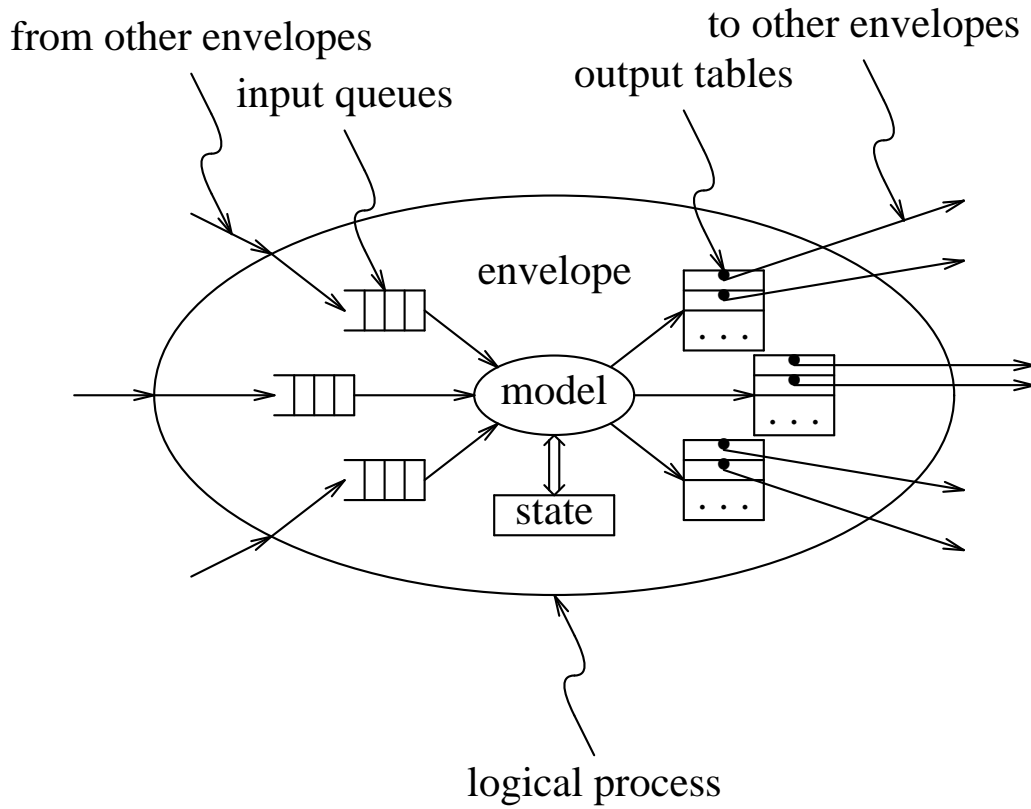
The Event-List-Driven Simulation Environment



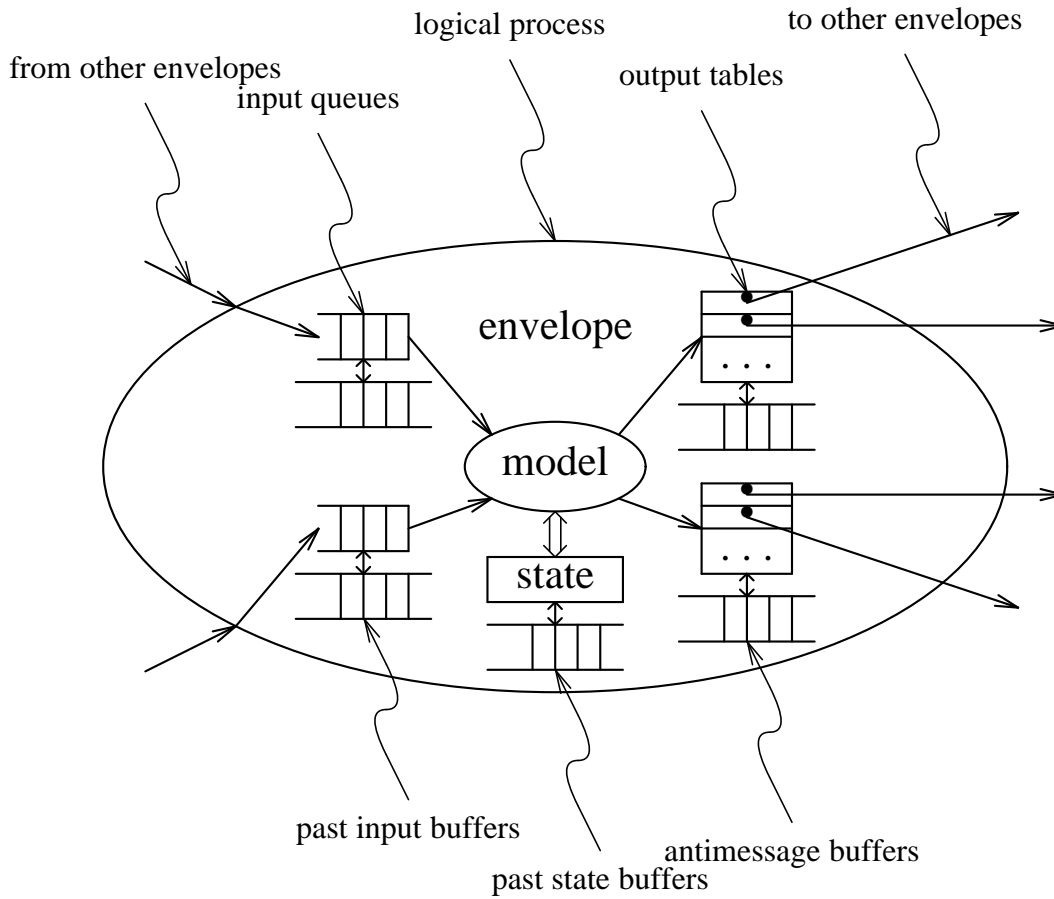
The Multiple, Synchronized Event List Environment



The Chandy-Misra Environment



The Virtual-Time Environment



Benchmark Simulations

- Acyclic logic simulation
 - 4-bit binary adder
- Cyclic logic simulation
 - 4-bit synchronous counter
- Closed Queueing System Simulation
 - multiprocessor communication structure (P-bus)

	Acyclic Logic	Cyclic Logic	Queueing System
No. of model specs.	15	12	7
No. of process specs.	50	58	49
No. of connect specs.	45	57	85
No. of External Inputs.	9	9	0
Execution Time (s)	6-119	3-91	77-1200

Acyclic logic simulation example

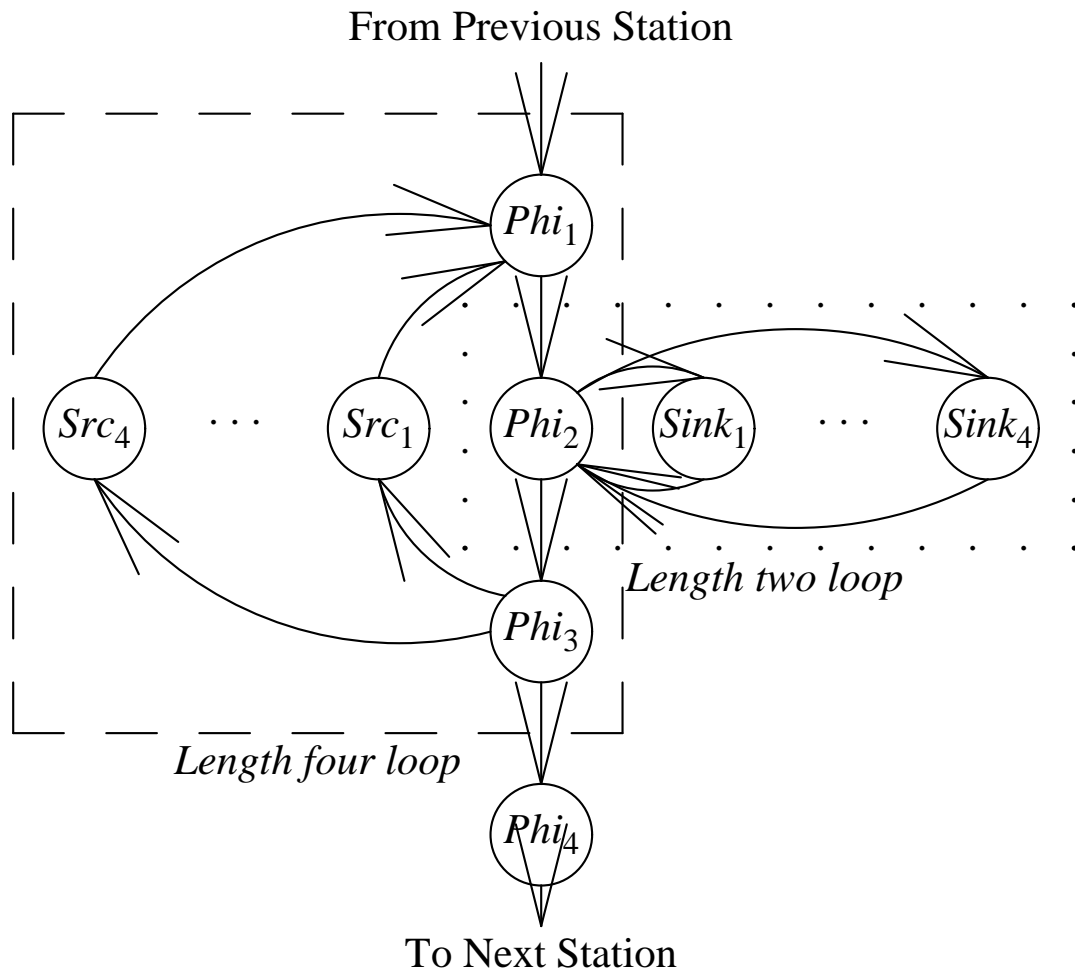
- 4-bit binary adder

Cyclic logic simulation example

- 4-bit synchronous counter

Closed queueing system simulation example

- multiprocessor communication structure (P-bus)



Deadlock Avoidance Techniques

- null messages — NULL_OUTPUT and OUTPUT
- ignore input x until time t — IGNORE
- deactivate input x — DEACTIVATE and ACTIVATE

Method	NULL MSGS	IGNORE	ACT/ DEACT	ACT/ DEACT
Loops Affected	—	2	2	2 and 4
No. of Messages	377,849	241,312	241,312	115,433
No. of Null Msgs	262,464	125,925	125,925	48
Sim. Time (ticks)	2,000	2,000	2,000	2,000
Exec. Time (s)	308	207	202	108

Performance Metrics

- additional overhead required by each simulation mechanism
- parallelism needed to provide any improvement over “best” sequential code
- performance requirements of the communication structure

Performance Measures

- T_{model}/T_{ex} — the fraction of time spent doing useful work
- T_{ex}/T_{Single} — a measure of the parallelism that would be needed to match the speed of the single event list
- N_{MC} — the total number of model calls needed for the simulation
- $(N_{mess}+N_{oh})/N_{MC}$ — the average number of messages sent per model call

Performance Results

	Single Event List	Four Event Lists	Chandy- Misra	Virtual Time
<hr/> Acyclic logic simulation <hr/>				
T_{model}/T_{ex}	0.32	0.15	0.16	.09
T_{ex}/T_{Single}	1	2.4	2.6	4.9
$N_{MC}/1000$	9.0	9.0	13.2	11.2
$(N_{mess}+N_{oh})/N_{MC}$	—	3.19	1.4	1.09
<hr/> Cyclic logic simulation <hr/>				
T_{model}/T_{ex}	0.28	0.12	0.08	.07
T_{ex}/T_{Single}	1	2.5	31.	6.4
$N_{MC}/1000$	3.8	3.8	76.6	9.4
$(N_{mess}+N_{oh})/N_{MC}$	—	4.01	1.61	1.07
<hr/> Closed queueing system simulation <hr/>				
T_{model}/T_{ex}	0.24	0.12	0.16	.1
T_{ex}/T_{Single}	1	1.9	1.5	3.6
$N_{MC}/1000$	94.	94.	94.	178.
$(N_{mess}+N_{oh})/N_{MC}$	—	3.18	1.22	0.79

Communication Requirements

Assumptions:

- communication and computation are done in parallel
- communication and computation are evenly distributed
- interprocessor communication structure is serial (i.e., a bus)
- parallelism is average number of events at head of event list (4.8 for adder, 4.7 for counter and P-bus)

$$Time_{for\ a\ single\ message\ transfer} = \frac{T_{ex}/Parallelism}{N_{mess} + N_{oh}}$$

- required message passing time: 95-420 μ s

Current Project Status

- four execution mechanisms

three implementations:

- portable
(uniprocessor version)

o/s:

- BSD 4.3 Unix
- Apollo DOMAIN/IX
- MS-DOS

processor:

- μ VAX II
- Apollo DN3010
- IBM-PC compatible
- VAX-only
(simulated multiprocessor version)
- fully distributed
(network of Apollo DN3010 workstations)
o/s: DOMAIN/IX with NCS and NIDL