Communication Facilities for Database Transaction Processing

by
Prof. Bharat Bhargava
Dept. of Computer Science
Purdue University
West Lafayette, IN 47907
USA

317-494-6013
317-494-0739 (Fax)
bb@cs.purdue.edu

Table of Content

- Statement of problem
- Assumption
- Transaction processing benchmark
- RAID distributed database system
- Related projects
- LAN communication for DTP
- Evaluation of RAID communication
- WAN communication for DTP

Legend

DTP Distributed Transaction Processing

LAN Local Area Network

WAN Wide Area Network

IPC Inter-Process Communication

RPC Remote Procedure Call

Statement of the Problem

How to control and improve the performance of distributed transaction processing (DTP) system by improving the communication and by exploiting the knowledge about network characteristics, architecture, and dynamics?

Assumptions

- DTP is communication intensive.
- The conventional communication scheme cannot meet the performance requirement.
- Wide area networks are characterized by large number of sites, variable communication delay, and high message loss rate etc.
- Distributed transaction processing experiences a serious performance degradation during failures or changing of communication conditions.
- It's difficult to conduct experiments and testing in WAN environments.

LAN vs WAN

Issues	LAN	WAN
Scale	small (10-100 sites)	large (over 100 sites)
Geographic span	within a mile	over 100 miles
Topology	bus, ring	hierarchical interconnected irregular mesh
Routing	simple	multiple hops, dynamic
Speed	very high (10-1Gbps)	low (1Kbps-10Mbps)
Error rate	low	high
Variation*	small	large
Ownership	private	public
Access	controlled	no control shared

* variation of parameters such as delay, throughput, and losses.

Motivation

To improve the performance of DTP by efficient communication facility.

To provide structure as well as efficiency.

To control and improve the performance of DTP by surveillance that exploits the knowledge about network characteristics, architecture, and dynamics.

Transaction Processing Benchmark for Measurements

- Relation size: 100 tuples.
- Hot-spot size: 20% of the tuples.
- Hot-spot access: 80% of the actions access hot-spot tuples.
- Type of experiment: close (maintaining the concurrency level constant).
- Number of transactions: 250.
- Transaction length. Average: 6. Variance: 4
- Timeout: one second per action.
- Updates: 10% of the actions are updates.
- Restart policy: rolling restart backoff.
- Concurrency controller: two phase locking protocol.
- Atomicity controller: two phase commit protocol.
- Replication controller: ROWA.
- Concurrency level: one transaction at a time.

RAID Experimental Infrastructure

- RAID laboratory with 5 Sun3/50's and 4 Sparcstation1's, all with local disks, microsecond resolution timers connected by a 10Mb/s Ethernet
- Software to specify, create, and maintain replicated databases
- Extended DebitCredit benchmark (Anon 85):
  - Transaction length, and number of transactions to generate
  - Transaction arrival rate or maximum degree of multiprogramming
  - Ratio of small to large transactions
  - Fraction of actions that are updates
  - Hot spot size and access percent
- Action Driver Simulator:
  - Interprets commands written in our benchmark language

- Generates *open-* and *closed-* system transactions

- Automated overnight experimental procedure:
  - Reboot machines at 3:00 am
  - Setup environment
    * Unmount non-essential file systems
    * Start up the Oracle (the naming server)
    * Initialize database relations
  - Execute experiments found on the Benchmark directory. For every specified experiment:
    * Start a RAID instance
    * Start the AD simulator to run transactions
    * Terminate the RAID instance
    * Upon termination, RAID servers dump statistics data
  - Next day: digest statistics data into condensed tabular format

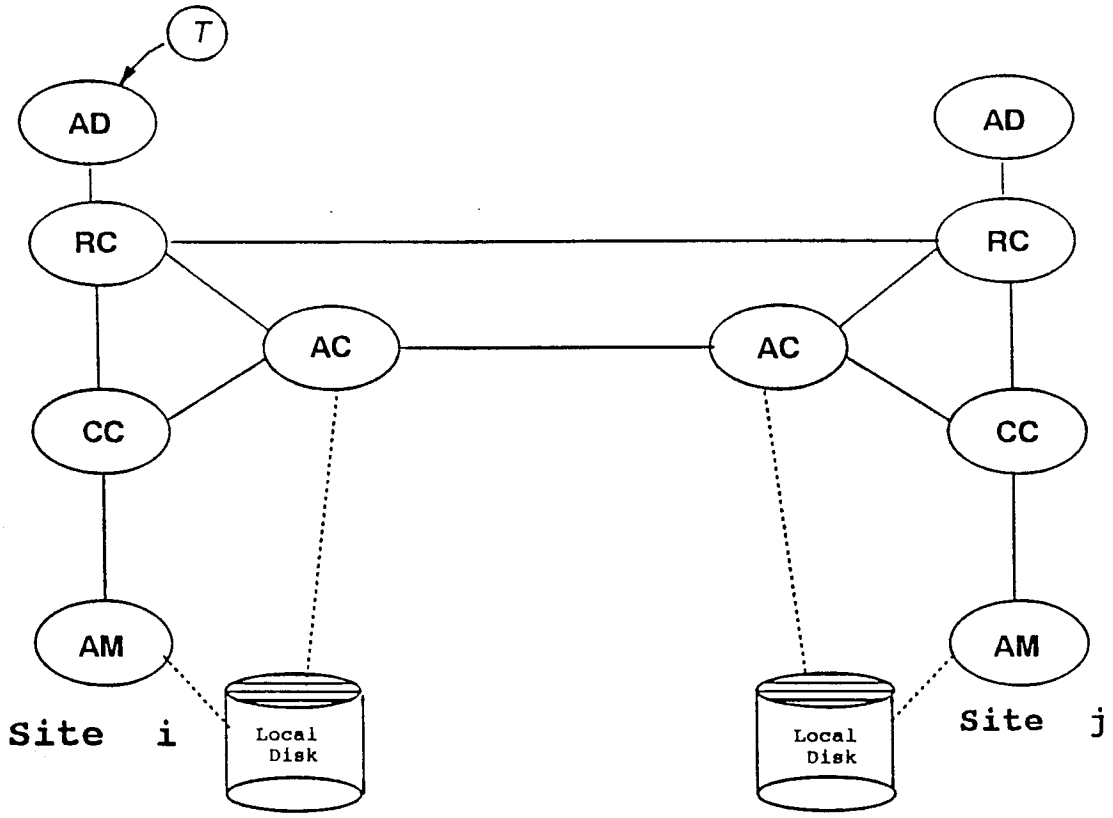
The RAID System

Robust and Adaptable Distributed Database System, Versions 1 and 2

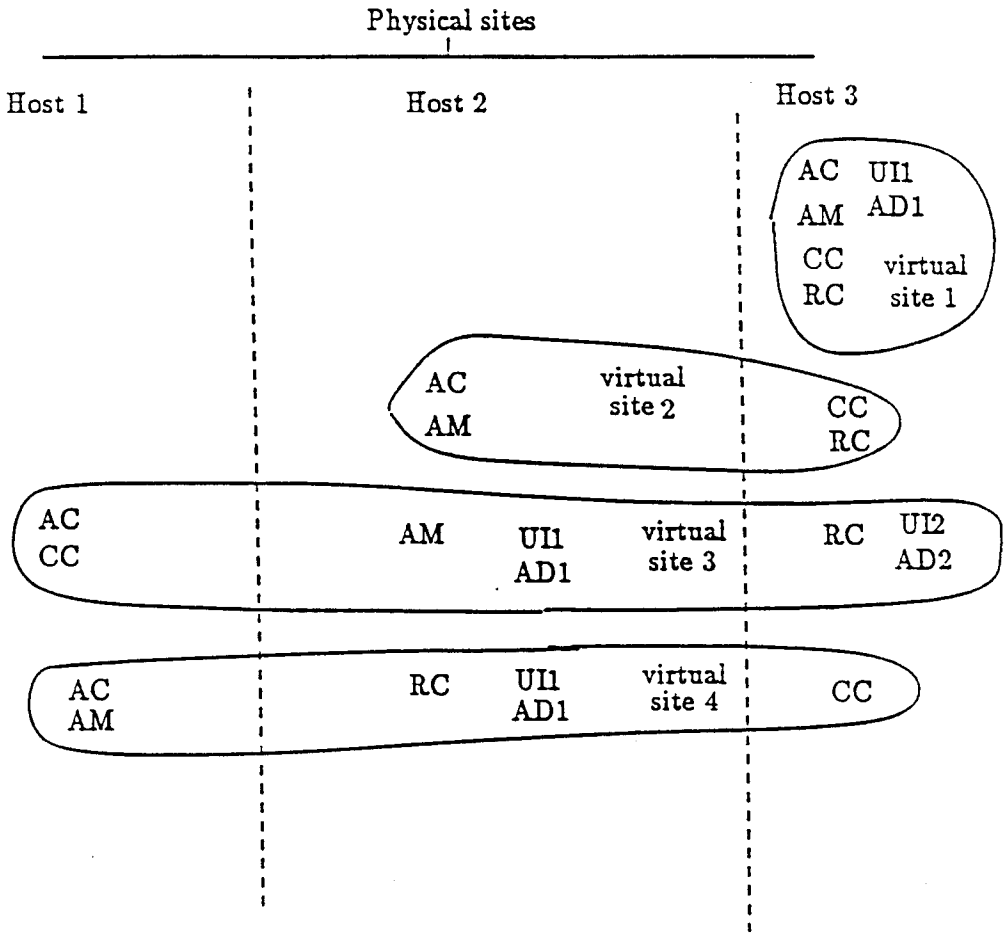
- functional distributed database system, 30k lines of C code, running on UNIX
- experimental system for investigating reliability and adaptability
- modular: independent servers connected by location-transparent UDP/IP-based communication package
- individual servers can adapt between algorithms

IEEE Transactions on Software Engineering, June 1989.

RAID Version 2



RAID Virtual Sites



Distinguishing Features of RAID-V2

- Improved flow of control for adaptability and fault-tolerance
- Directory-based partial replication
- Adaptable quorum-based replication control:
  - An adaptable version of the Quorum Consensus (Gifford 79), in which quorum assignments are determined by a quorum relation
- Dynamically adaptable concurrency control
- Use of XDR to support communication in a heterogeneous system

Adaptability in RAID

- Concurrency Control
- Replication Control
- Distributed Commit
- Network Paritioning
- Reconfiguration after Site Failure
- Inter-server Communication
- Server Relocation
- Parallel vs. Merged Servers (P-Raid)
- Object-relation database model (O-Raid)
- Kernel Extensions (PUSH)

IEEE Transactions on Knowledge and Data Engineering, December 1989.

Related Projects

- Networking/Communication
  - internetworking
  - transport protocols
- Distributed File Systems
  - Cedar, Sprite, Locus, etc
  - Andrew/Coda file systems
- Distributed Operating Systems
  - Amoeba, Athena, Mach, V, *x*-kernel
  - authentication server (Kerberos), group communication
- Database/Transaction Processing
  - Camelot, Argus, Eden, ISIS, Raid
  - Federated databases, multi-databases (R*)
  - Advanced transaction models

Networking/Communication

- IPC in Distributed Systems
  - Accent - IPC in kernel
  - Mach - hand-off scheduling
  - Camelot - a DTP build on top of Mach
  - VMTP - support intra-system model of DTP
  - DUNE - dynamic binding in RPC
- Local IPC
  - Lightweight RPC
  - User-level RPC
- Communication Protocols and LAN
  - VMTP
  - Virtual/layered protocol (x-kernel)
  - atomic broadcast, multicast
- Communication Support for DTP
  - Camelot, RPC

Distributed Operating System

Amoeba

(Vrije, Tanenbaum)

- Employs transaction model.

- Improves performance by using efficient RPC and local protocols.
  (1.10ms on a 3-MIPS machine)

- Uses "Amoeba gateway" to provide transparent communication in WANs without affecting the performance of RPCs in LANs.

- Fast Local Internet Protocol (FLIP).

Distributed File System

Andrew/Coda File System

(CMU, Satyanarayanan)

- Scalability as the design goal
- Improve performance by volume replication and client caching
- Improve performance by segregating files and access by their semantics
- Improve reliability by distinguishing "disconnected" and "connected" modes

LANs Communication for DTP

- Approaches
- Problems
- Solutions
- Our experiences

General Approaches for LANs

- Multi-threaded, light-weight process
  (Camelot)

- Multi-tasking
  (for multiprocessor machines)

- Merged servers
  (sacrifice the structure)

- Efficient RPC
  (Amoeba, Sprite, V, *x*-kernel)

- Primitives for transaction management
  (Mach)

Our Approaches for LANs

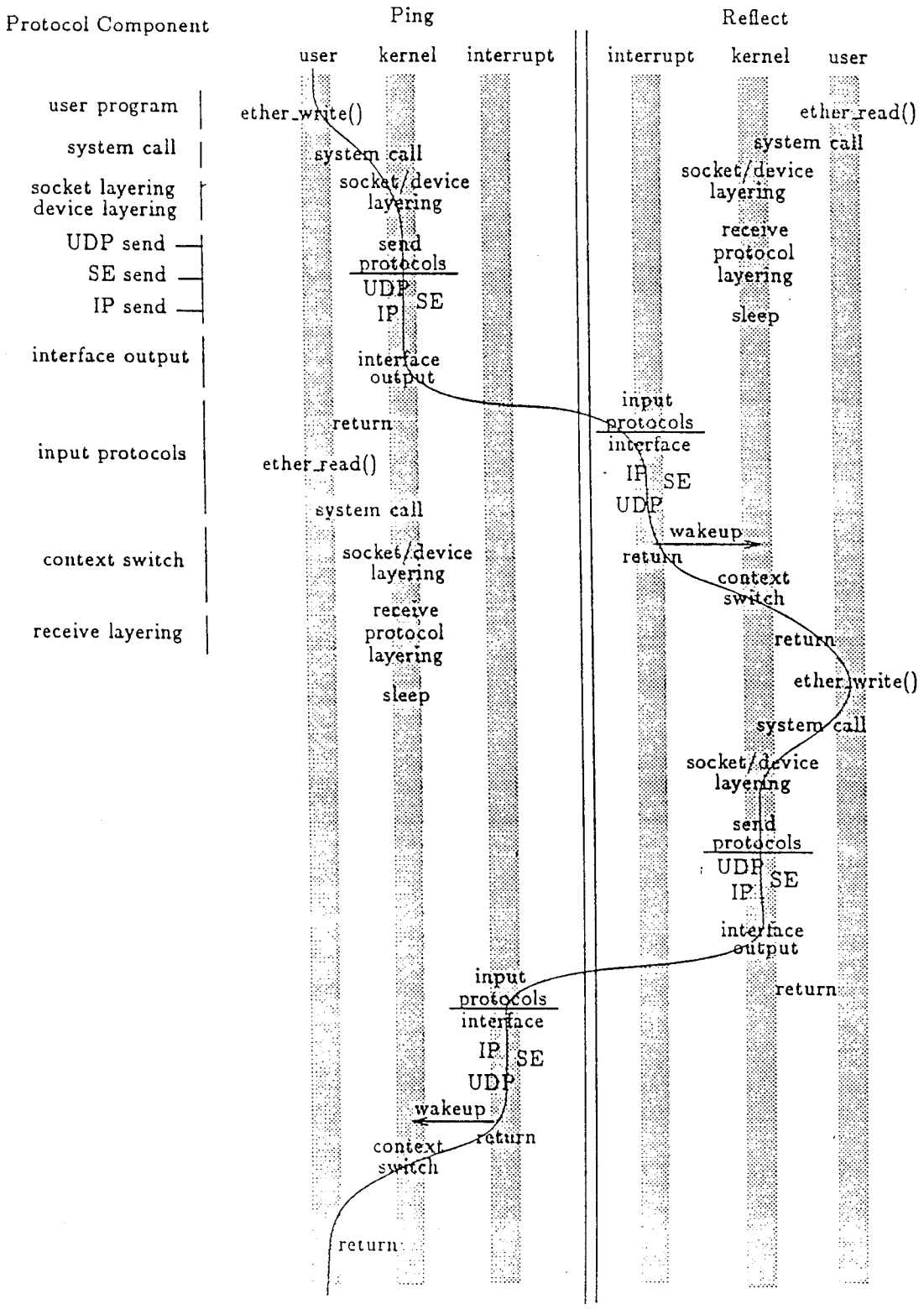
- Lightweight protocol
- Simple naming scheme
- Memory mapping and shared memory
- Physical multicasting
- Explicit control passing
- Adaptable to the communication environment

Experiments with Conventional Communication Schemes

- Socket-based Interprocess Communication
- Local Communication
- Multicast Communication
- Impact on Transaction Processing

Problem with Conventional Communication Schemes

- General purpose communication facilities are expensive
- DTP systems: communication intensive, most are local rather than remote
- Lack of communication support for DTP in some OS
- General purpose multicasting is unusable
- Name resolution is too expensive



Timing diagram for UDP and SE datagram services.

Local IPC Measurements

METHOD	MESSAGE SIZE 10 Bytes Time (ms)	1000 Bytes Time (ms)
2 Q Message Passing	2.0	2.9
1 Q Message Passing	2.0	2.9
Named Pipes	2.3	3.9
Shared Memory	5.1	5.5
UDP Communication	4.3	9.6

- message queues are 1/4 the cost of UDP
  - less top-heavy
  - less data copying

Ethernet Experiment

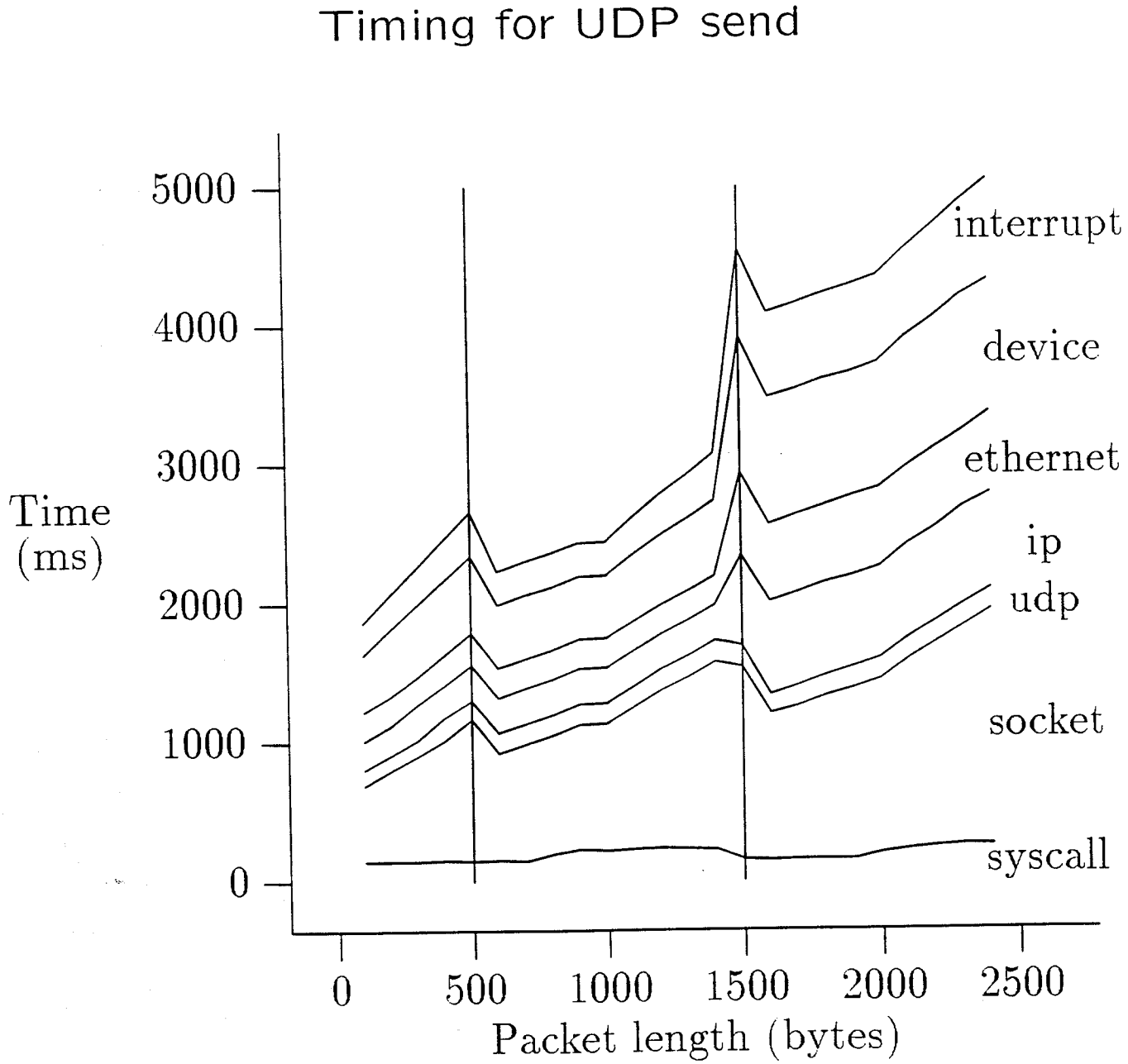
Timing of Components of UDP Send

Measurement Instant	Time (in ms)	Protocol Component	Duration (in ms)
start	0	-	0
after socket layers	1.74	socket layers	1.74
after connection	2.28	connection	0.54
end of UDP output	2.58	UDP send	0.30
before IP checksum	3.10	IP send	0.52
end of IP output	3.22	IP checksum	0.12
entire round-trip including receive	7.2		

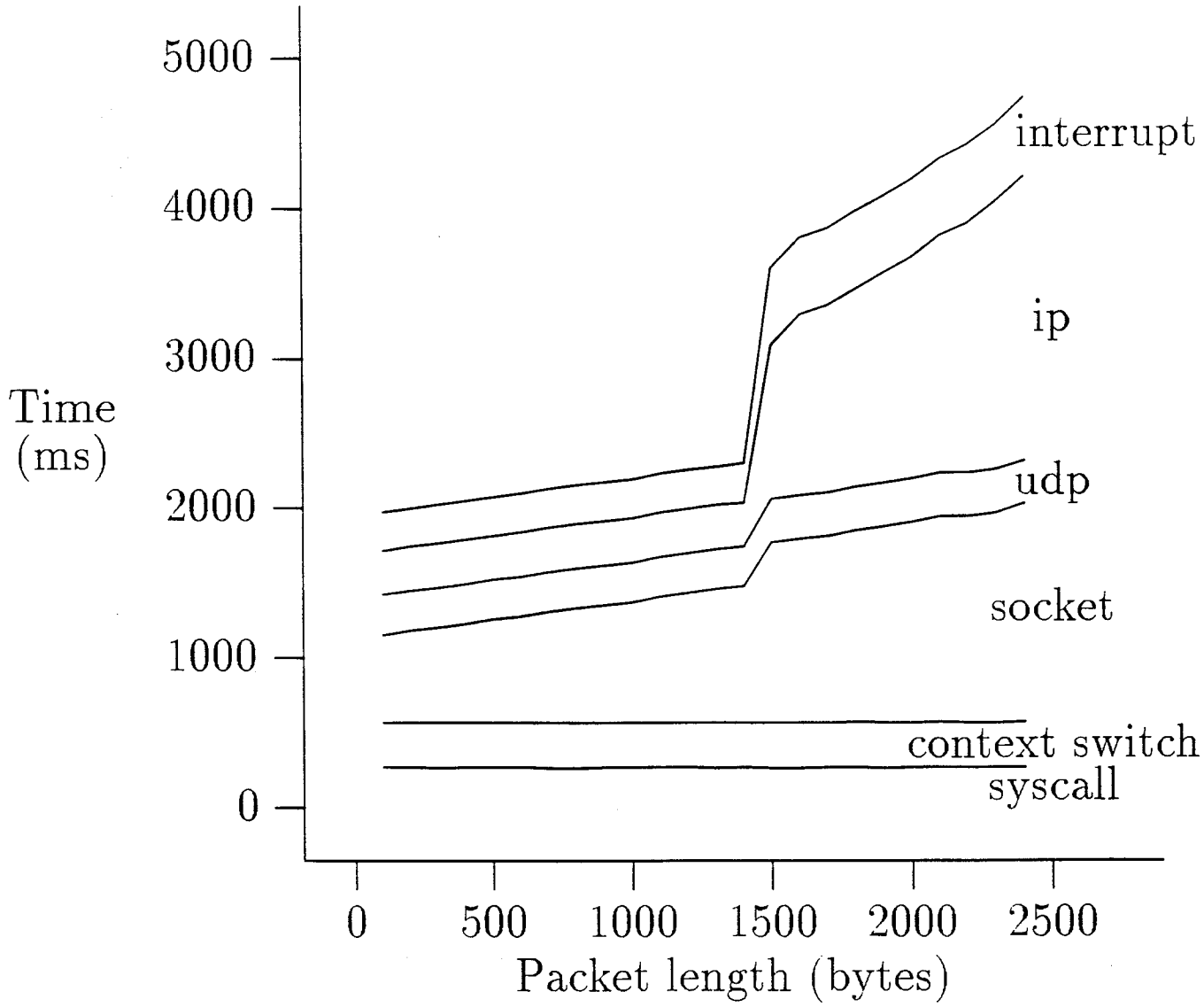
Timing of Simple Ethernet Components

Protocol Component	Time (in ms)	Measurement Tools
user-level code	0.08	user
system call	0.33	senull - user
device layering	0.61	null - senull
kernel transmit	1.36	kping
mbufs on read	0.45	`se_fast_return`
context switch	0.40	insomnia - senull
kernel-to-user copy	0.30	`se_null_read`

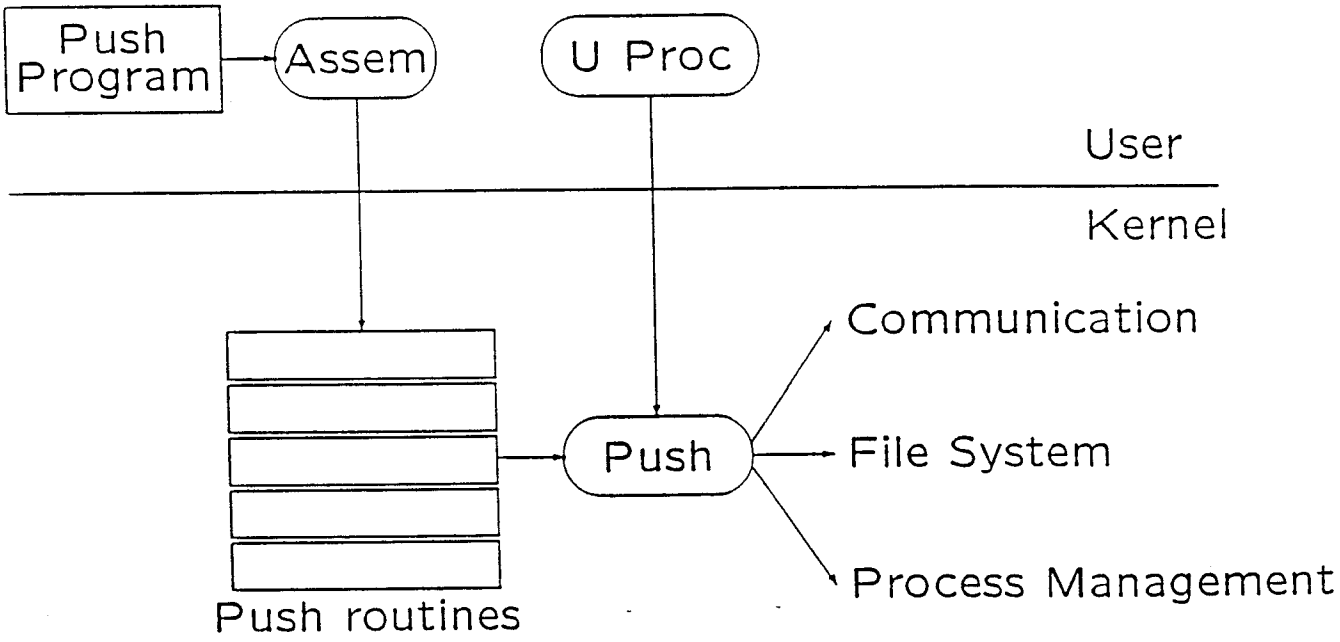
Evaluation of Unix IPC model (I)



Evaluation of Unix IPC model (II)



The PUSH system

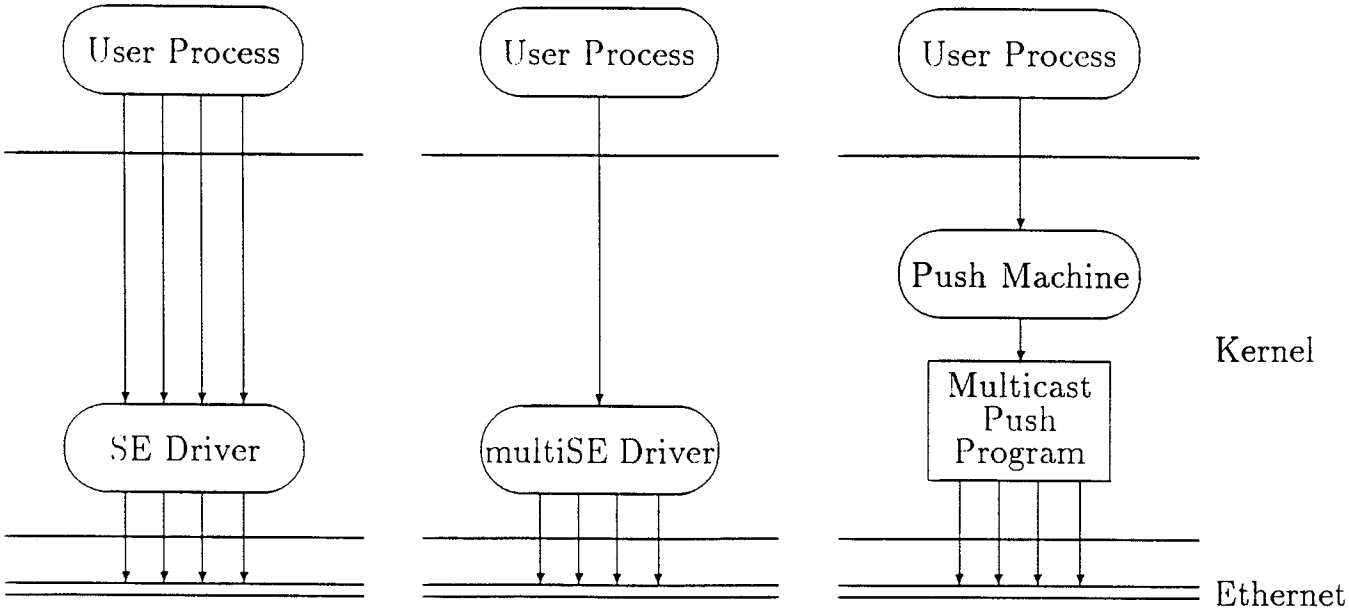


Multicasting Timing (in ms)

Number of destinations	kernel level SE	user level SE	Push
1	1.2	1.2	2.7
5	4.2	5.9	6.6
10	8.0	11.7	11.0
15	11.7	17.5	15.6
20	15.4	23.4	20.2

Evaluation of Multicasting (I)

Approaches for Multicasting

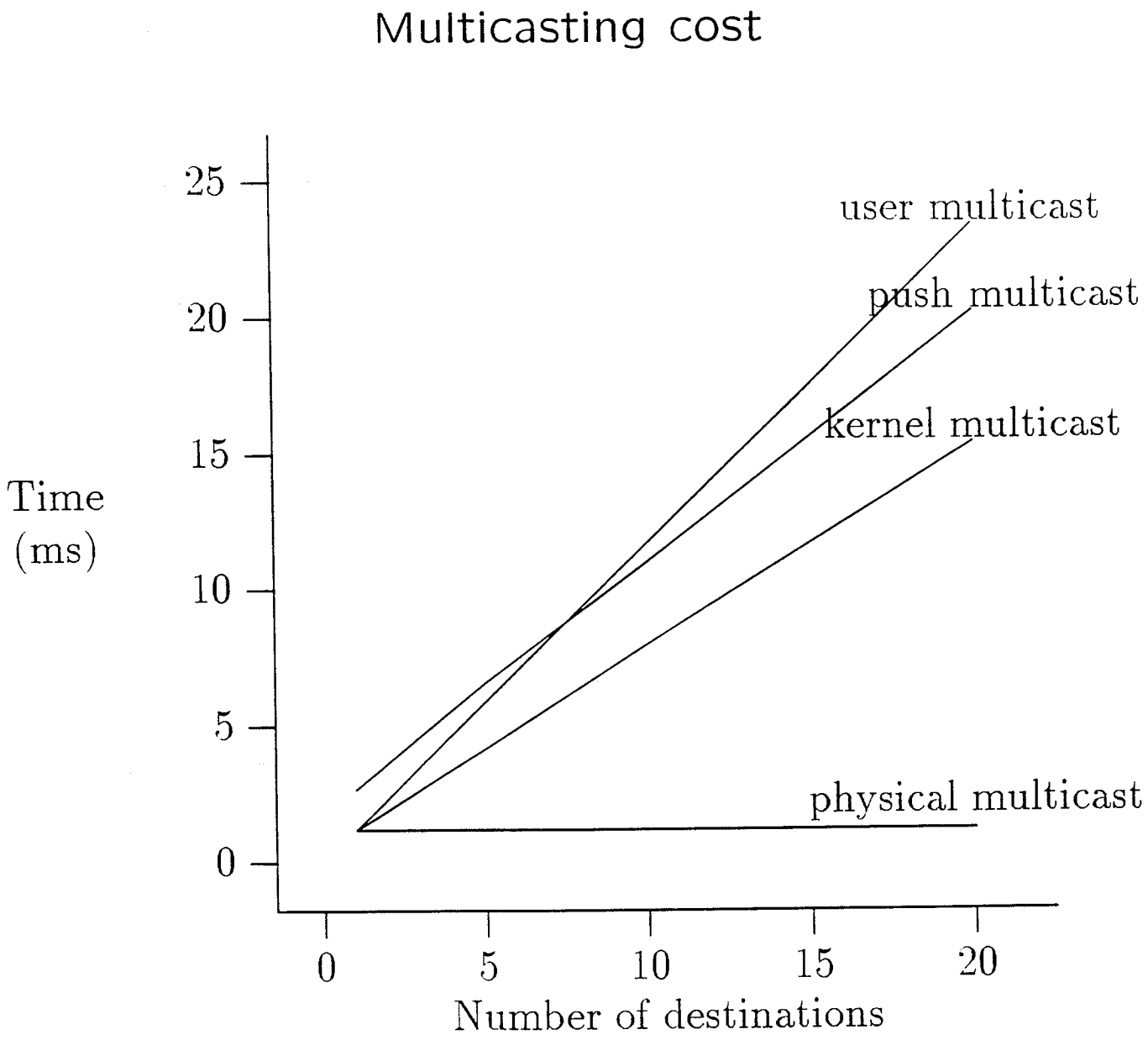


(a) User–level SE multicast

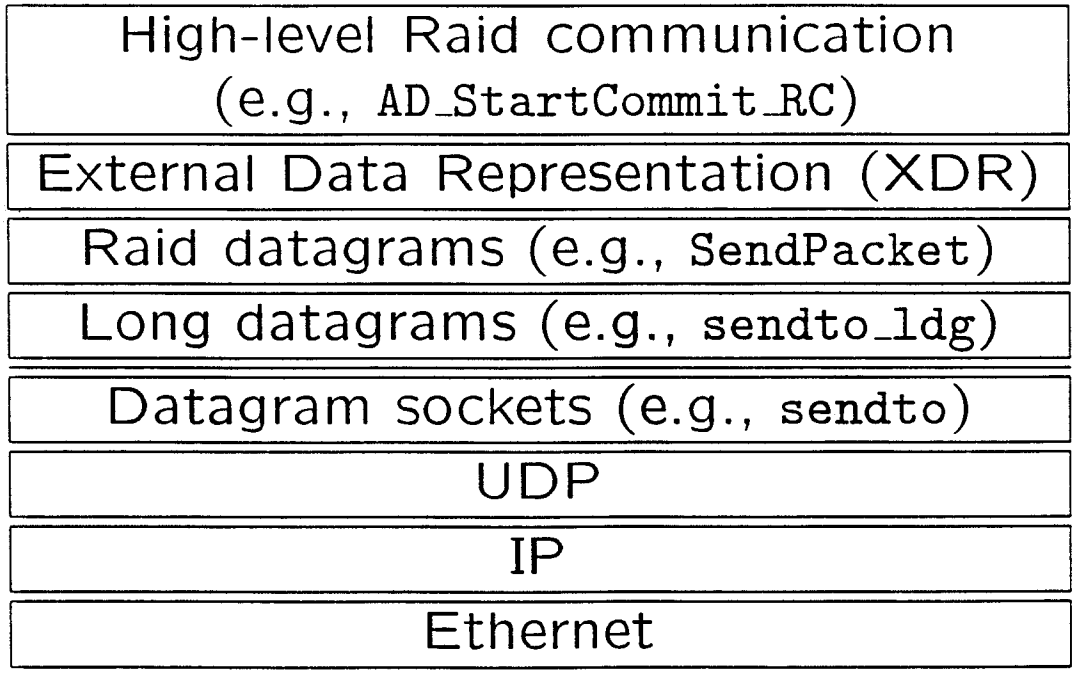
(b) Kernel–level SE multicast

(c) Push multicast

Evaluation of Multicasting (II)

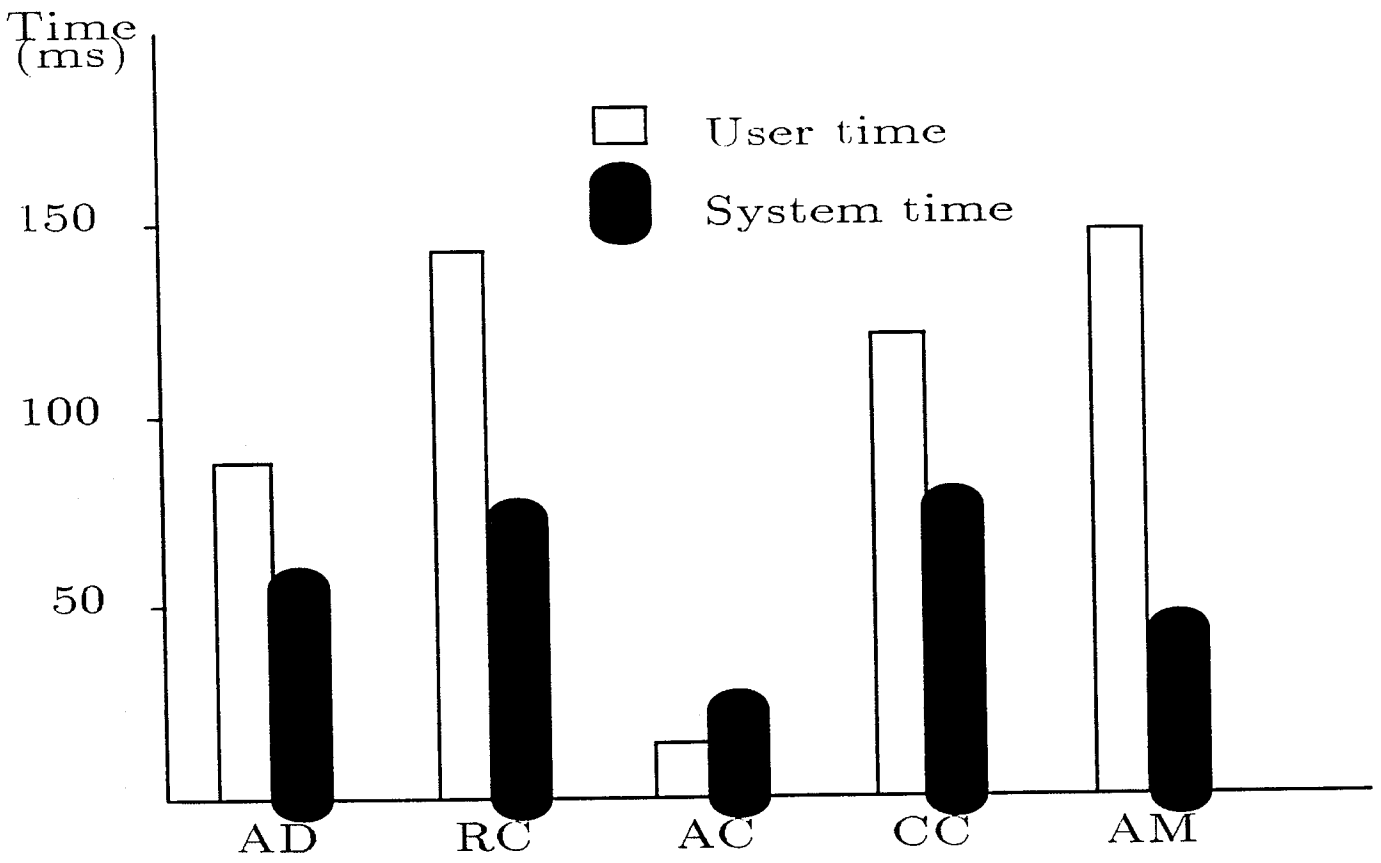


Raid Communication Subsystem, V.1



plus a separate process: name server `oracle`

Evaluation of Raid communication V.1



Legend:
AC = atomicity controller
CC = concurrency controller
AM = access manager
RC = replication controller
AD = action driver

RAID Elapsed Time for Transactions

transaction	1 site	2 sites	3 sites	4 sites
select one tuples	0.3	0.3	0.4	0.4
select eleven tuples	0.4	0.4	0.4	0.4
insert twenty tuples	0.6	0.6	0.8	0.8
update one tuple	0.4	0.4	0.4	0.4

RAID Atomicity Control CPU Time

	1 site		2 sites		3 sites		4 sites	
transaction	user	sys	user	sys	user	sys	user	sys
select one tuples	0.04	0.14	0.06	0.14	0.04	0.10	0.08	0.24
select eleven tuples	0.04	0.08	0.02	0.04	0.06	0.12	0.06	0.10
insert twenty tuples	0.20	0.16	0.08	0.14	0.10	0.12	0.08	0.10
update one tuple	0.04	0.10	0.06	0.12	0.06	0.16	0.06	0.16

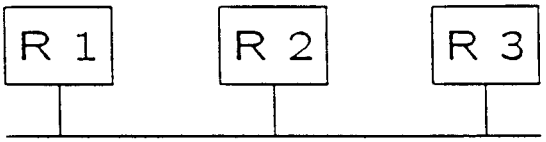
Times reported in seconds.

RAID Concurrency Control CPU Time

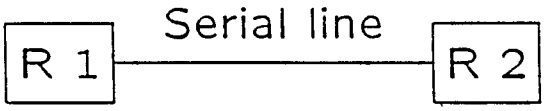
server	CC	
transaction	user	sys
select one tuple	0.04	0.06
select eleven tuples	0.02	0.02
insert twenty tuples	0.12	0.13
update one tuple	0.02	0.02

Times reported in seconds.

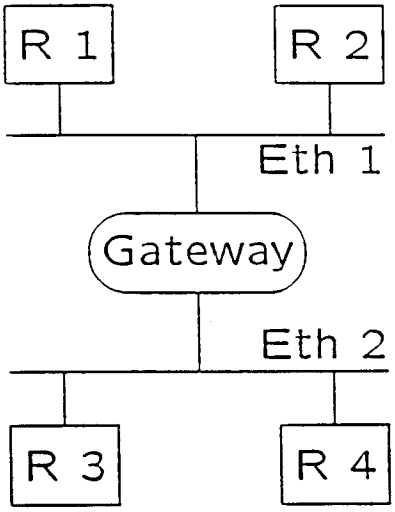
Network Configurations for Raid



(a) LAN



(b) 19200 bps serial line



(c) Internetwork of LANs

Transaction Execution Time on Different Communication Topologies (in ms)

Transaction	a-1	a-2	b	c
select one tuple	100	100	240	120
insert twenty tuples	320	380	520	400
update one tuple	100	120	260	120

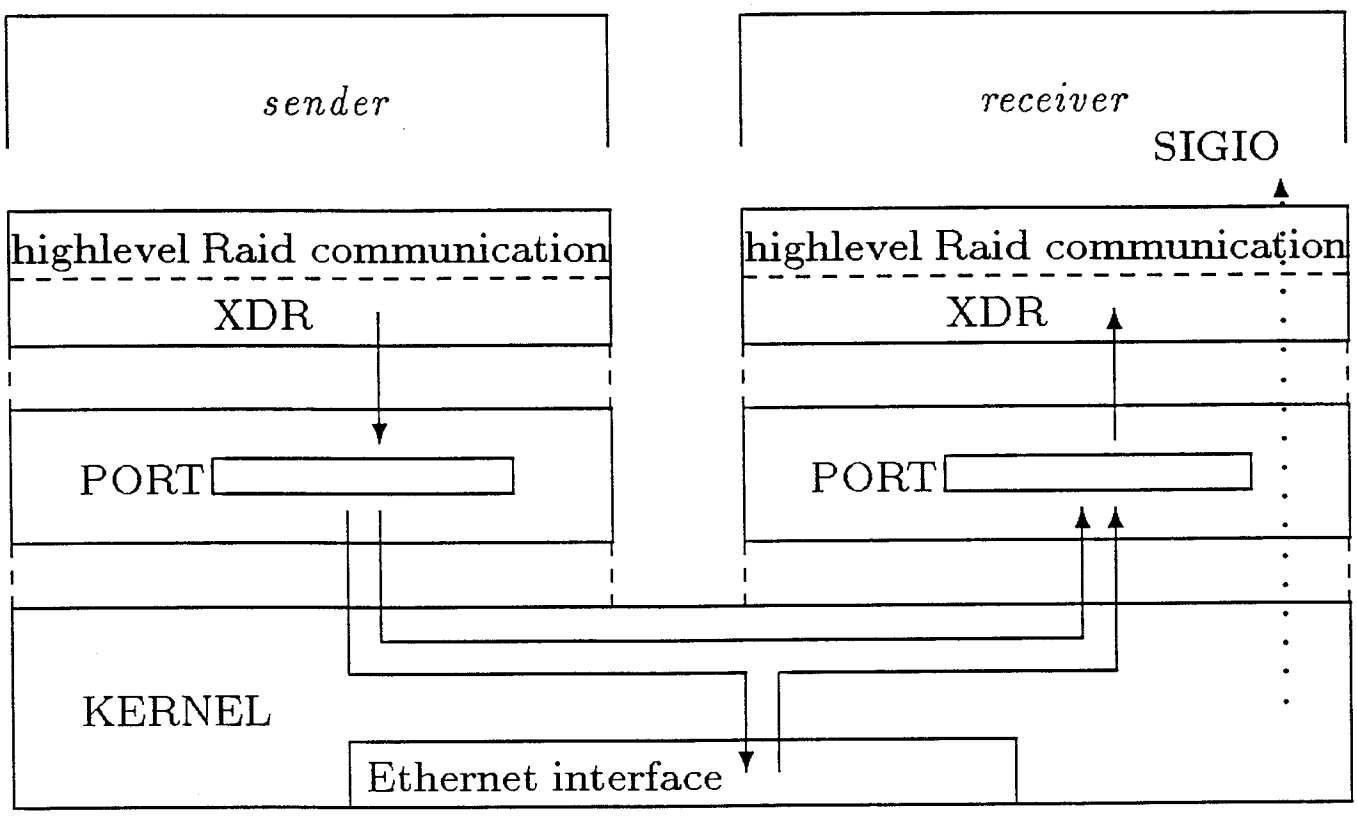
Improvement Emphasis for Raid Communication subsystem Version 2

- Avoid top-heavy communication abstractions
- Reduce kernel interaction
- Minimize message copying
- Use a simple IPC memory management
- Adopt same mechanism for local and remote
- Exploit the nature of DTP

Raid Communication Subsystem V.2

- Port: shared by process and kernel
- Protocol: SE for LAN
- Naming: simple naming scheme:
  ⟨Raid instance, server, server instance⟩ ⇔ port number
- Multicasting: each site sets a multicasting address using the transaction ID.
- Communication Primitives

Structure of Raidcomm V.2



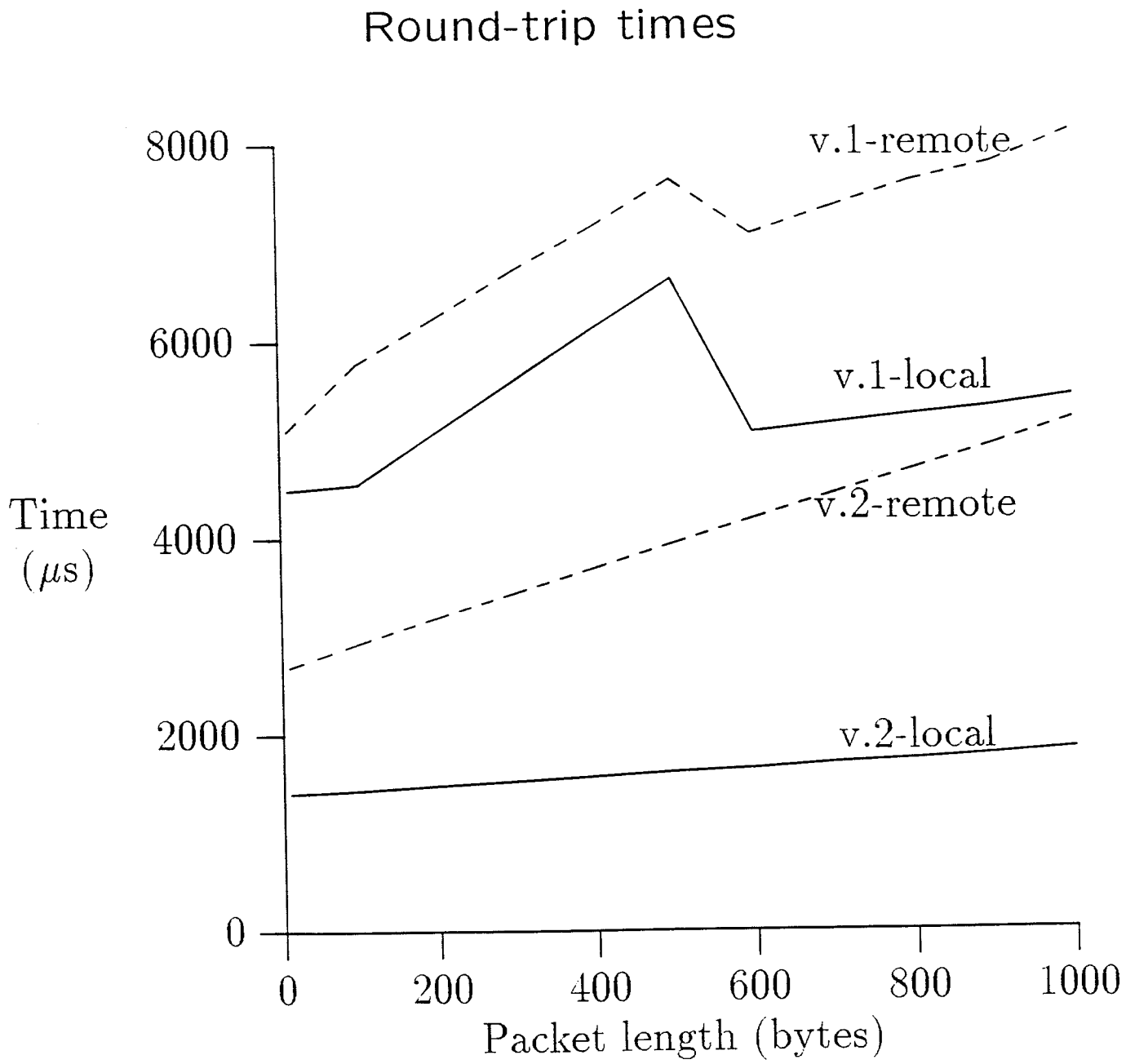
Structure of a Communication Port

trmlen	Transmission Buffer	Active Buffers
len1	Receive Buffer 1	
len2	Receive Buffer 2	
len3	Receive Buffer 3	
	. . . . .	
lenN	Receive Buffer N	

Performance: Communication Primitives

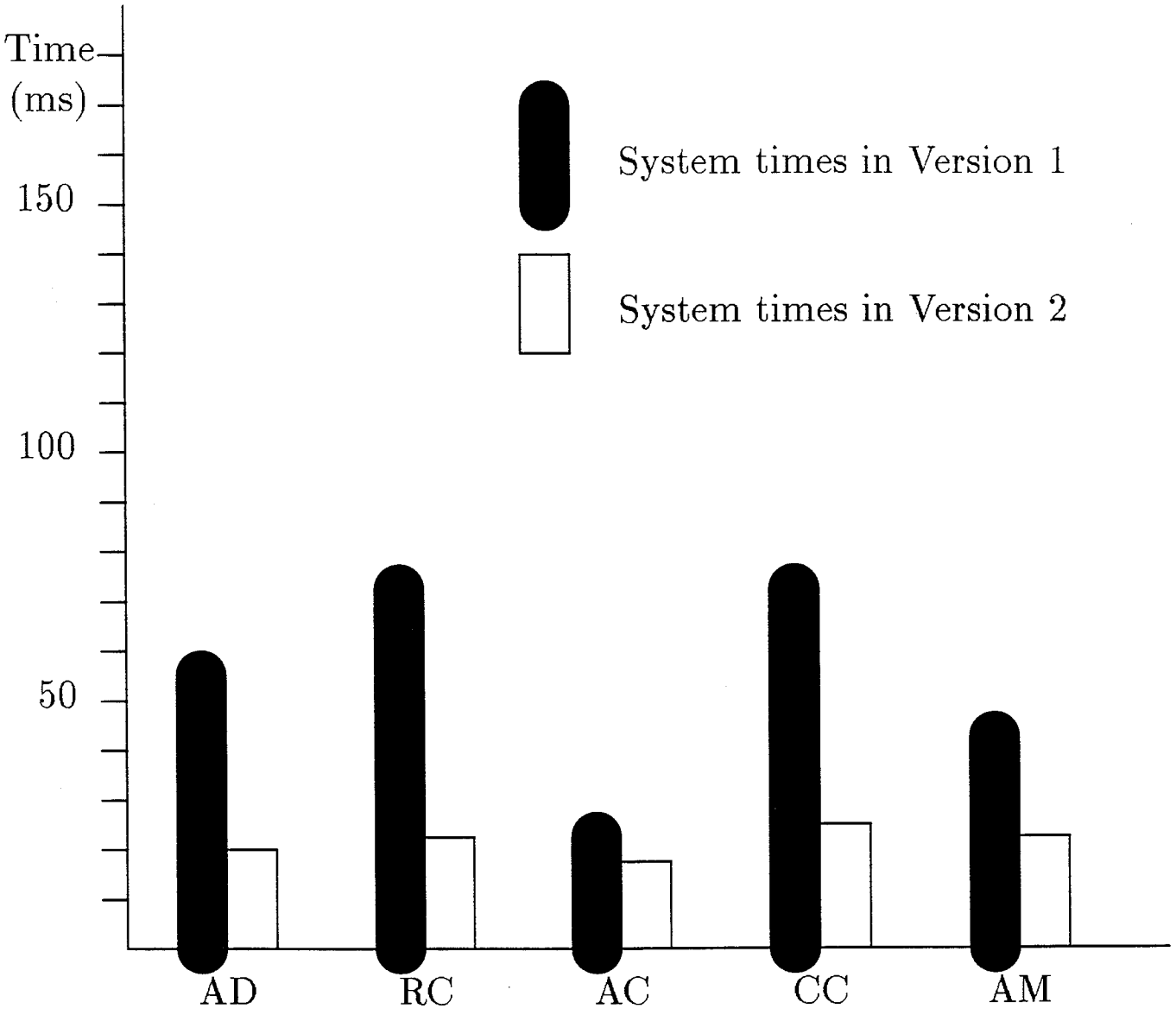
- Goal: to evaluate the performance of Raidcomm Version 2
- Experiments: measure the local and remote round trip times
  - Add socket-based IPC and two SYS V local IPC methods for comparison
- Conclusions:
  - Our protocol is extremely lightweight
  - Most of the local round trip time is due to context switch overhead

Evaluation of Raidcomm V.2



Evaluation of Raidcomm V.2

Round-trip times

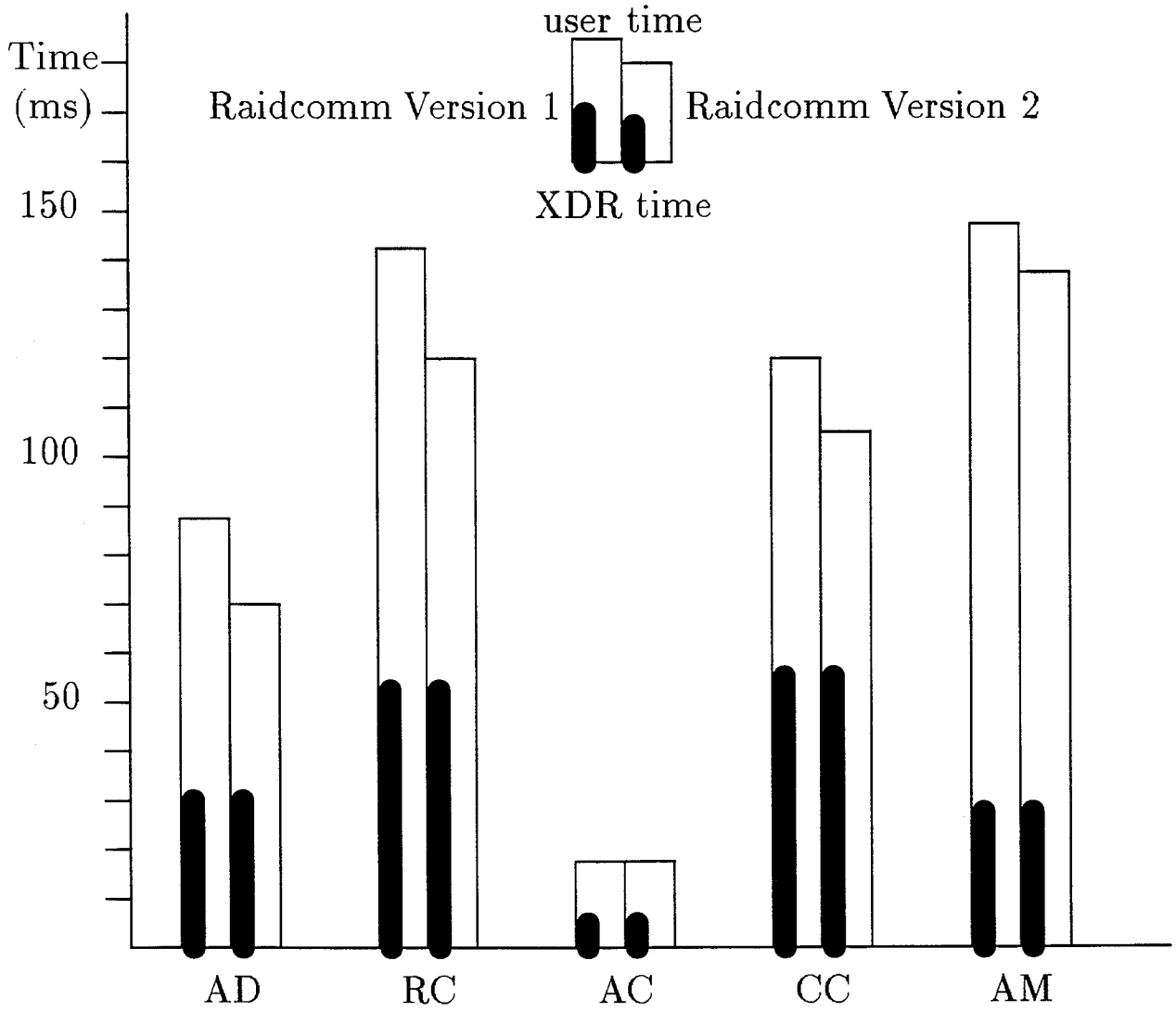


Problem with XDR

- A data representation standard
- Doesn't take into account the high-level demands
- Encoding and decoding are expensive
- Unnecessary in most cases

Evaluation of XDR

Average user times for a transaction in a single machine



Problem with context switch and scheduling

- OS scheduling policies do not consider high level relationships in a group of processes.

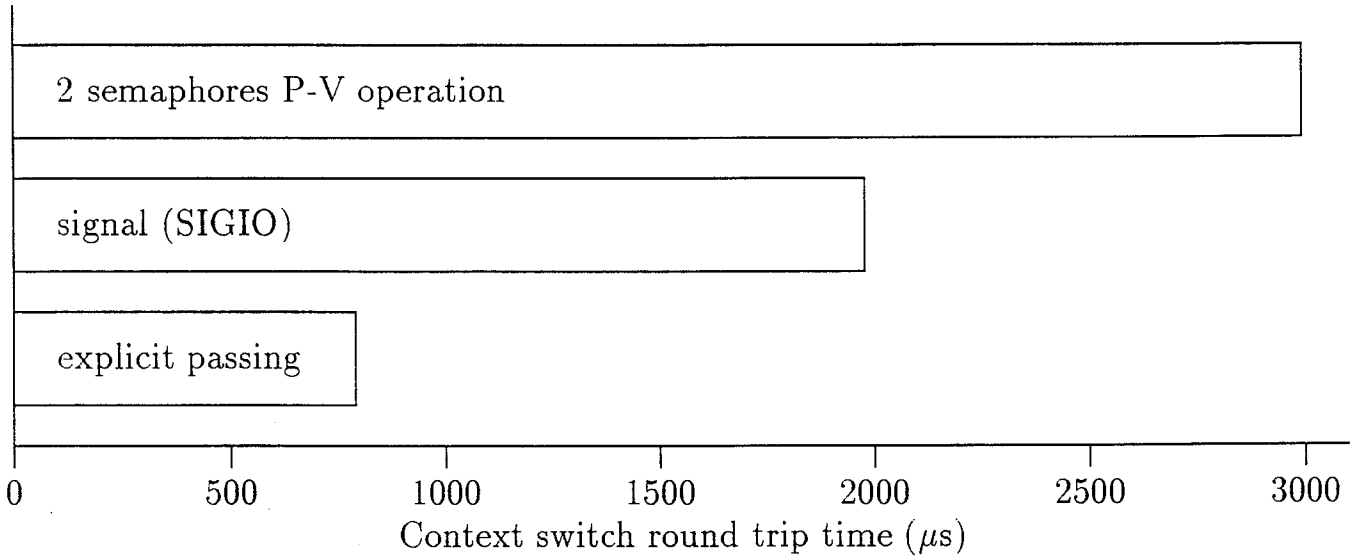
- Raid scenario:

  CC is CPU intensive and its priority is decreasing after some times. This forces CC is give up CPU after processing only one message, even though its time slice has not yet expired.

- context switches caused by synchronization are expensive.

Evaluation of Context Switch

The performance of context switch



Improvement Emphasis for Raid Communication subsystem Version 3

- Avoid unnecessary data formatting
- Provide fast complex data transfer
- Eliminate kernel's involvement in communication activities
- Use shared-memory as communication channel
- Reduce context switch overhead
- Use explicit control passing scheme