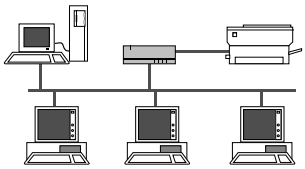


Distributed Systems
600.437

Group Communication

Department of Computer Science
The Johns Hopkins University

Programming Exercise 1

Please add your HOPS/UGRAD account names to the top of submission

Make sure the directory:

~username/cs437/ex1/

Is world readable and contains the entire project including source code, a make file or build instructions, result files, and the project report.

Group Communication

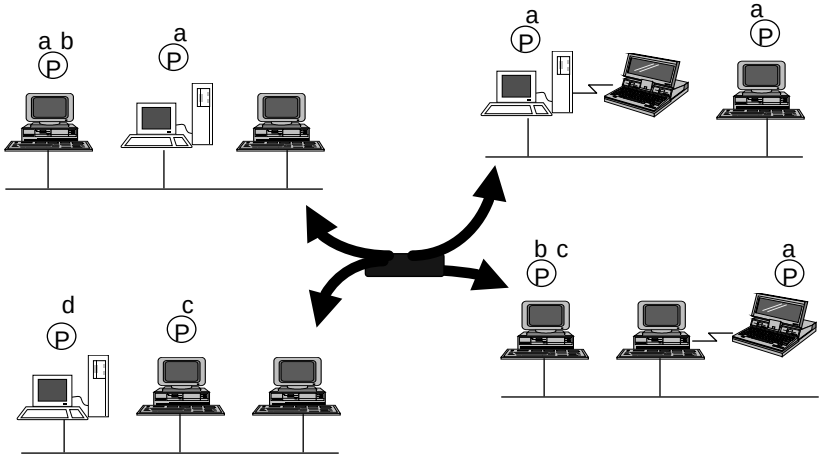
Lecture 5

Further reading:

Building Secure and Reliable Network Applications

The course web page http://www.cnds.jhu.edu/courses/cs437/

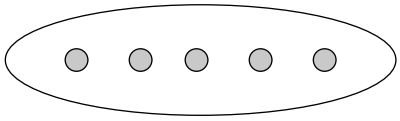
Group Communication



Who Needs Group Communication ?

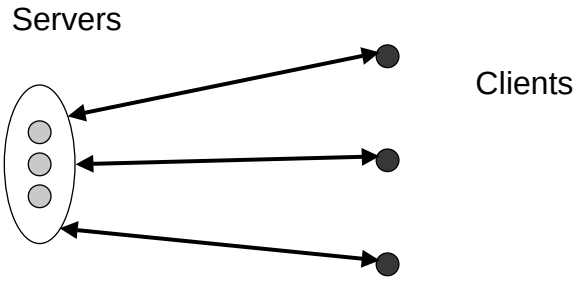
- Highly available servers (client-server).
- Conferencing
- Database replication
- Cluster management
- Distributed logging
- ...

Communication Pattern 1
Peer Group



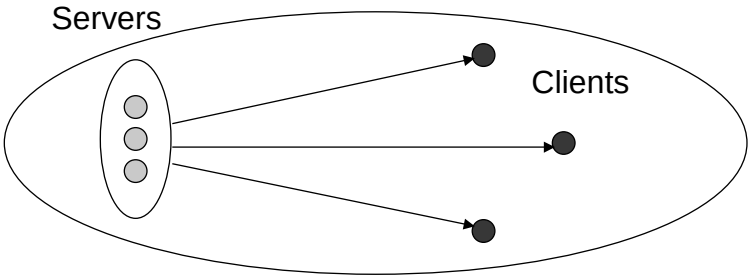
- All the members are equal.
- All the members send messages to the group.
- All the members receive all the messages.

Communication Pattern 2
Client-Server Group



- Common communication pattern--replicated servers
- Client does not care which server answers

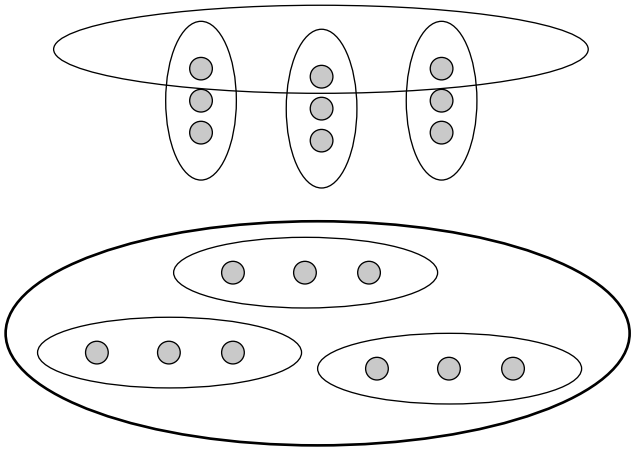
Communication Pattern 3
Diffusion Group



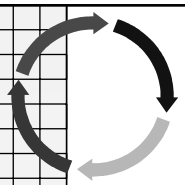
- Servers send to Servers and to Clients.
- Clients usually do not send to the group.

(Stock price updates).

Communication Pattern 4
Hierarchical Group



- Ease scaling problems.



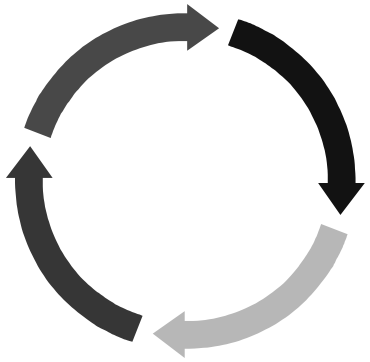
The Single Ring Protocol

- UCSB 1992-3
- Open groups.
- The communication is broadcast (UDP/IP).
- Services: Agreed (which is also FIFO and Causal), Safe.
- supports message omissions, network partitions, crashes and recoveries.

The Ring Ordering Scheme

Token fields

- type - {regular, form}.
- seq - of last message.
- **aru** - replaces acks.
- rtr - retrans. requests
- fcc - flow control.



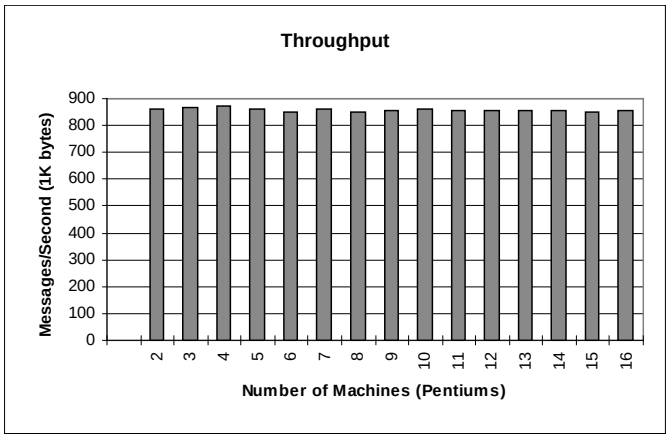
The Ring Ordering Scheme (cont)

How to update the token aru?

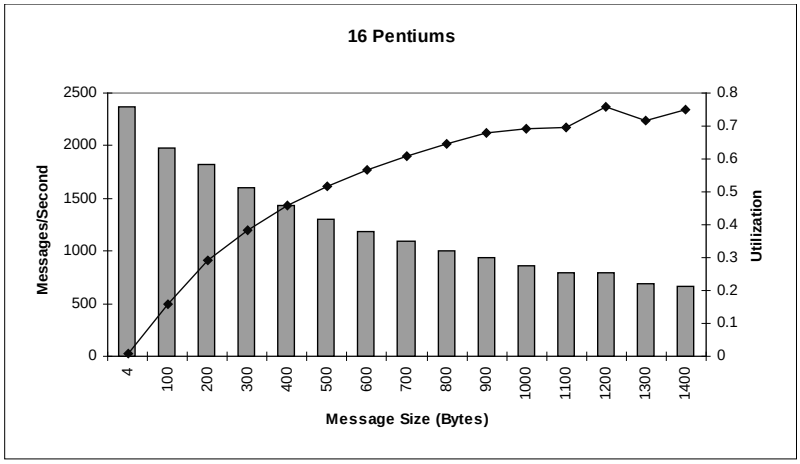
- If token.aru = token.seq and have all the messages then should higher aru together with the seq (when send new messages).
- If missed a message with m.seq smaller than token.aru then should set token.aru = m.seq
- If is the one that lower the aru, and the token.aru is still the same, should set token.aru to its local aru.

The trick: Everyone has all the messages up to:
min(token.aru, previous token.aru)

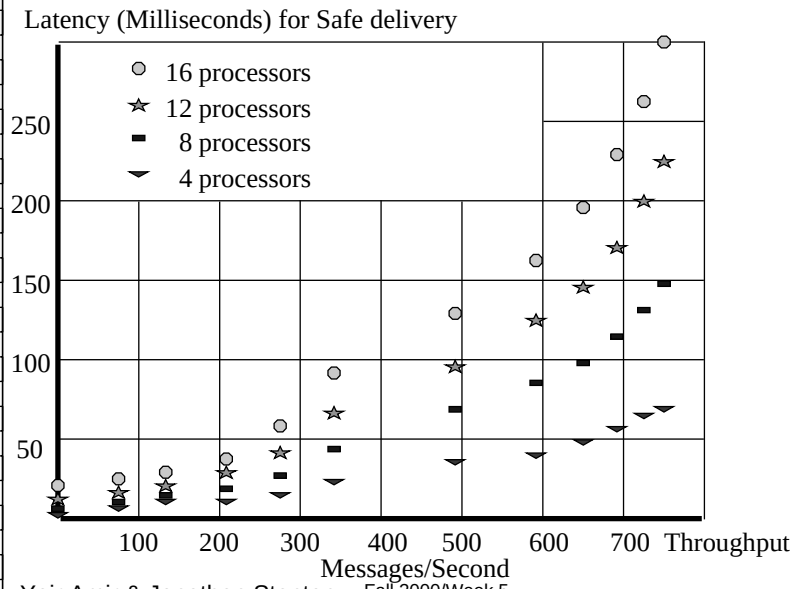
Performance



Performance (cont)



Performance (cont..)



Quick Design Problem!

- Dot com content site has 50 servers serving content. The company's revenue depends on exact logs of how many pages are viewed and by what people.
- Each server produces about 500 log entries per second. Each log entry is about 100 bytes. All logs need to end up in at least one location. All 50 servers are in one geographic location. A fast network with several segments connects them.
- Design a system to achieve this task.
  - **GOAL:** correctness, low cost, scalability, simplicity, ease of setup.

Failure Models

Possible faults:

- Message omissions and delays
- Processor crashes and recoveries
- Network partitions and re-merges

Most of the time it is assumed that:

- Message corruption is detected
- There are no malicious faults

Transis:Membership Algorithm

- Utilizes hardware broadcast
- Ordering and Reliability optimized by DAG
- Handles crashes and recoveries.
- Handles network partitions and merges.
- Terminates in a bounded time (to do that, it allows the extraction of live but “inactive” processors).
- Guarantees extended virtual synchrony (relationship between messages and membership events).

Membership (cont.)

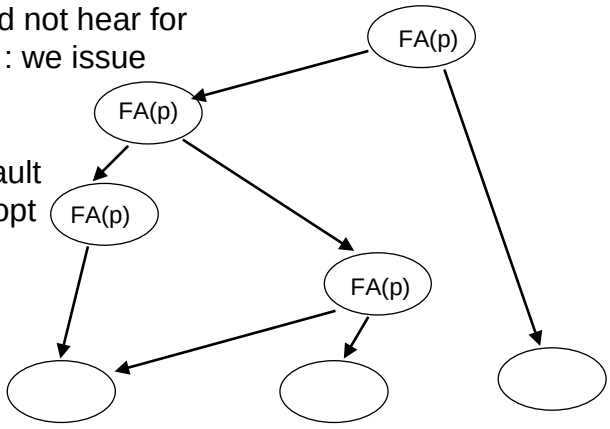
- Partitioning / crashes detection
  - Timeout: invoked by timeout.
- Merging
  - Symmetric: no joining-side, accepting-side
  - Spontaneous: invoked after receiving Join messages or “foreign” messages.
- Faults may occur at any time (even while merging).

Faults & Partitions

When Detecting a processor from which we did not hear for a certain timeout : we issue a fault message.

When we get a fault message, we adopt it (and issue our copy).

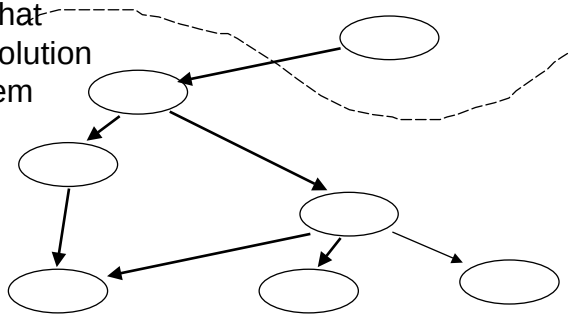
Problem: maybe p is only slow.



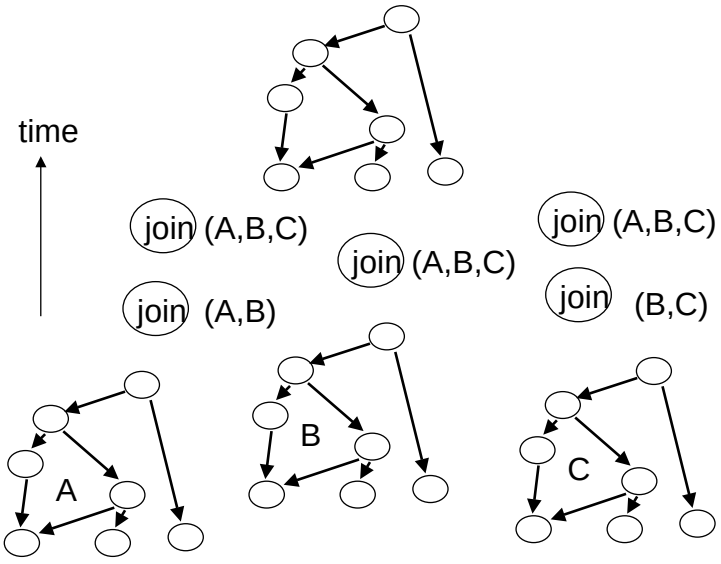
The Problem

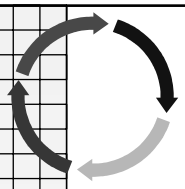
When a partition occurs, we cannot always completely determine who received which messages.

It is proven that there is no solution to this problem (no common knowledge)



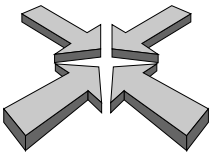
Merges





The Single Ring Protocol

- Membership has several stages:
  - Detect that old membership is lost.
  - Gather together all alive members.
  - Form a new ring and send old state.
  - Transfer missing messages.
  - Install new membership.
- Supports message omissions, network partitions, crashes and recoveries.



Membership

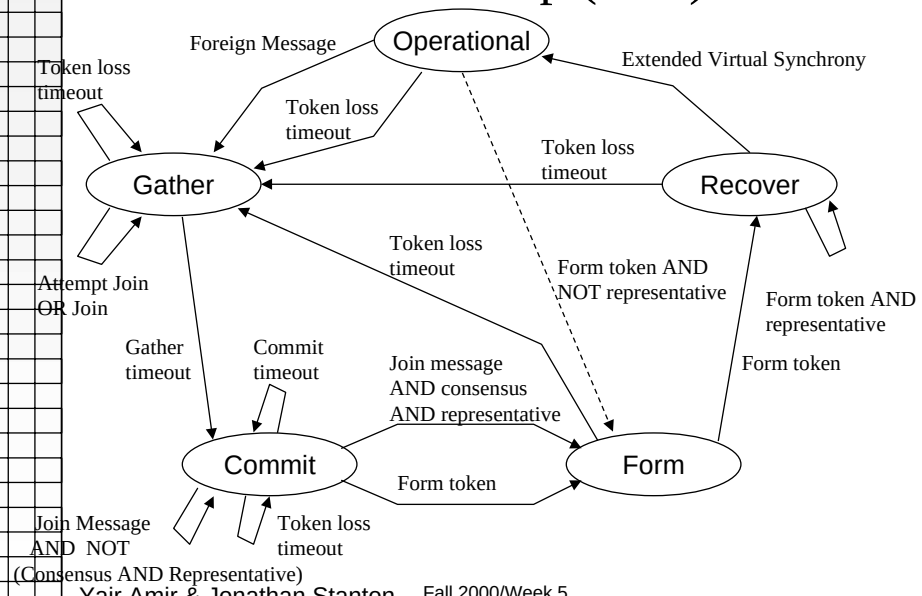
Events

- Foreign Message.
- Attempt join/ Join.
- Gather timeout.
- Commit timeout.
- Form token.
- Token loss timeout.

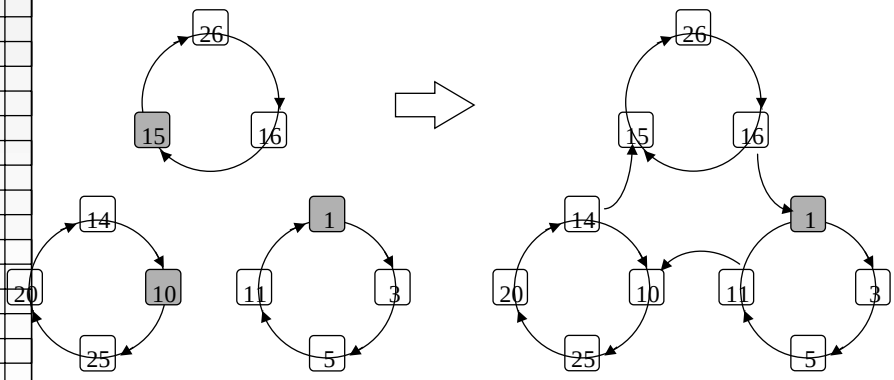
States

- Operational state.
- Gather state.
- Commit state.
- Form state.
- Recover state.

Membership (cont)



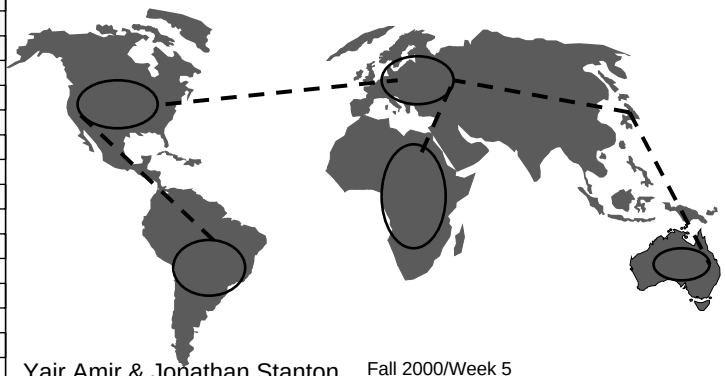
Membership (cont..)



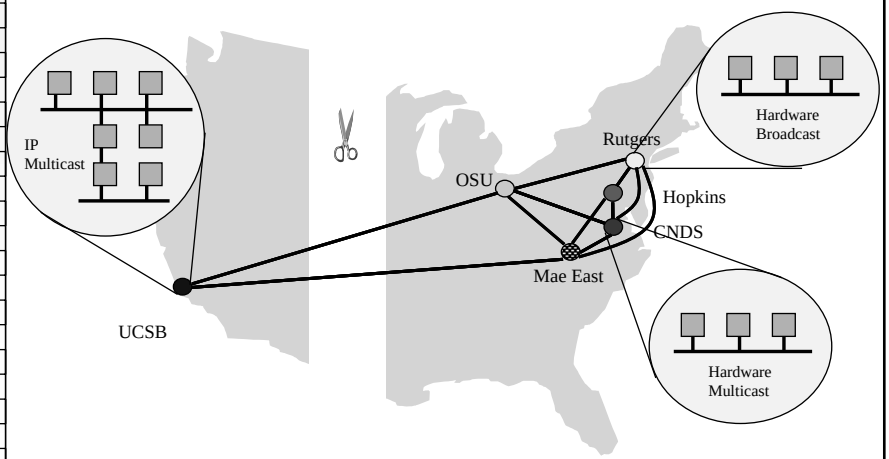
Representatives are shown shaded

Spread Group Communication

- Wide-area membership & distribution
- Open groups (daemon-client model)
- Virtual overlay networks



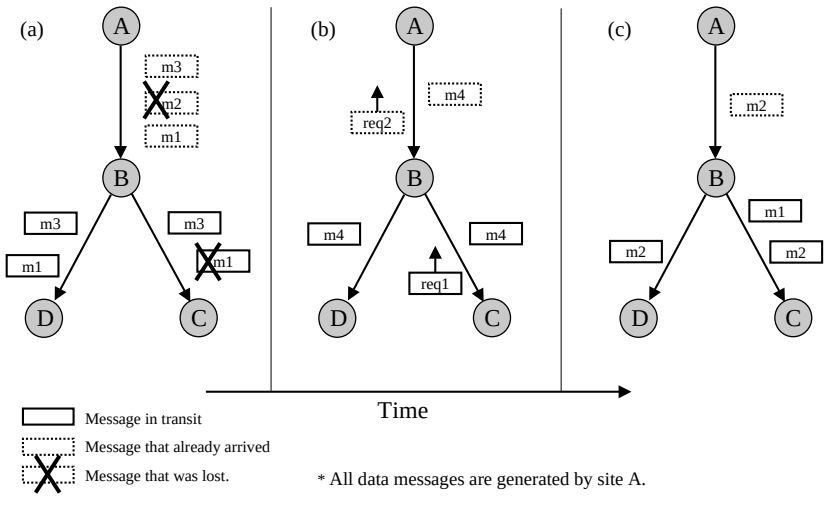
Overlay Networks in Spread



Hop Protocol

- Requirements
  - Eventually Reliable transport
  - Link-level Flow/Congestion Control
- Not Needed
  - FIFO ordering (higher layers take care of that)
- Approach
  - Non-Blocking
  - Lazy-Selective-Retransmits
  - Rate-based flow control
  - Eventually reliable links

Non-Blocking Example



End-to-End Reliability

- Group Communication systems already have strong membership services.
- Coupling membership with link-level eventual reliability you get end-to-end reliability.
- Three possible cases:
  - No failures, no slow receivers.
  - No failures, slow receivers.
  - Failures occur.

Spread Ordering Algorithm

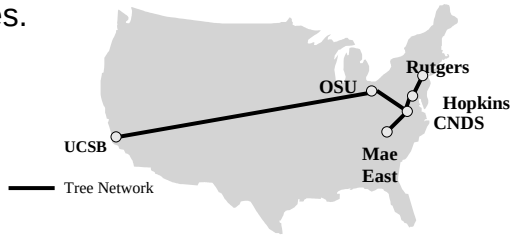
- Messages are stamped with several sequence values when created:
  - Site Sequence number (one sequence per site)
  - LTS (Lamport Time Stamp) (one sequence overall)
  - FIFO Sequence number (one sequence per client)
- Complete received messages are stored in a queue ordered by LTS and SiteID
- The highest FIFO sequence value that is deliverable in each session is stored
- Each daemon stores a My_Site_ARU value (my site has all messages with an LTS less then that) and an ARU value (everyone has all messages with an LTS less then that).

Spread Ordering (cont.)

- Delivery consists of applying 4 ordered tests to the Message List and delivering any messages which pass the tests to the session layer
  - Any message <=“SAFE” (LTS< ARU) can be delivered and deleted
  - Then any message <= “AGREED” (LTS < My_Site_ARU) upto first SAFE message
  - Then any message <= “FIFO” for which (session_seq < Max FIFO deliverable seq) or is RELIABLE
  - Unreliable messages can be delivered anytime and might be delivered faster then identical RELIABLE messages

Wide Area Throughput

- Source sends 10,000 reliable messages of 1024 bytes.

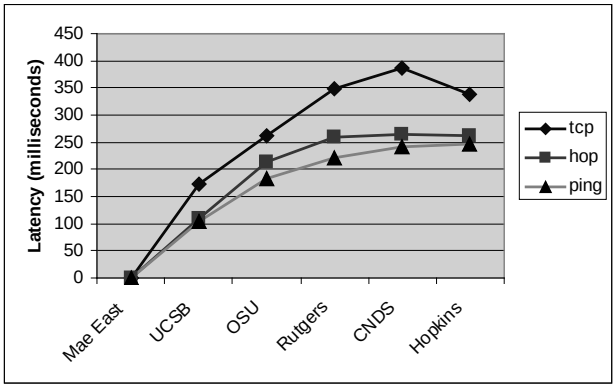


Sending Site	Fanout	Tree
Mae East	487.32	559.79
CNDS	666.94	560.71
Rutgers	184.99	568.34
UCSB	328.81	578.84

Rates in Kbits/second

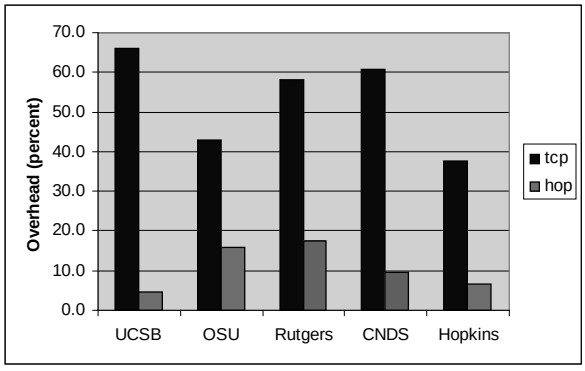
Multiple Link Latency (Mae East)

- The 1024 byte messages were sent by Mae East and bounced back by each of the other sites in the Chain network.



Multiple Link Latency (Mae East)

- Overhead in latency, in comparison with ping, that is added by each protocol.



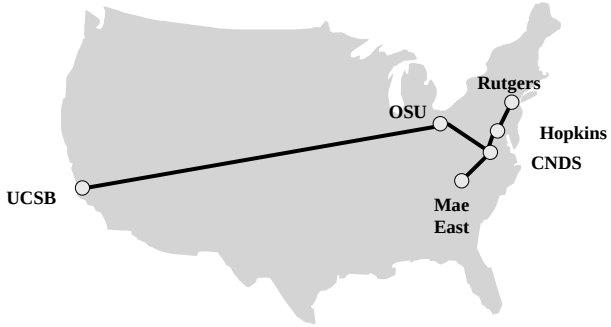
Single Link Throughput of Hop under Loss

- The single link was Mae East to UCSB.
- Packets randomly dropped on each side of the link with uniform probability.
- The generated losses were in addition to any real losses on the network.
- Each test sent 10,000 messages of 1024 bytes.

Additional Loss Rate	0%	5%	10%	20%
Throughput (Kbits/sec)	1378.807	1269.043	1113.887	995.000
Throughput (Percentage)	100.0%	92.0%	80.8%	72.2%

Multiple Link Throughput of Hop under Loss

- Full Tree network



Additional Loss Rate	0%	5%	10%	20%
Throughput (Kbits/sec)	2202.78	2090.45	1807.49	1388.51
Throughput (Percentage)	100.0%	94.9%	82.1%	63.0%