High Speed Total Order for SAN infrastructure

Tal Anker, Danny Dolev, Gregory Greenman, Ilya Shnaiderman

School of Engineering and Computer Science

The Hebrew University of Jerusalem

SAN as a distributed system

- SAN is a distributed system that provides many services:
  - Archival and retrieval of archived data
  - Backup and restore
  - …
- This work focuses on efficient way to implement:
  - *Fault tolerance of internal elements*
  - *Data sharing by different servers*

Fault Tolerance by State Machine Replication

- An internal server is “cloned” (replicated)
- Each replica maintains its own state
- The states of replicas are kept consistent by applying transactions to all the replicas in the same order
- The order of transactions is established by a total order algorithm

Types of Total Order Algorithms

- Symmetric (Lamport 78)
  - Messages are ordered based on a logical timestamp and sender's id
- Leader-based (Sequencer)
  - All messages are ordered by a special process called sequencer

Data Sharing and Locking

- In order to keep shared files consistent, a lock mechanism is to be implemented taking into account that:
  - locking is a frequent event in SAN
  - SAN is expected to provide high throughput
  - lock requests should be served promptly (low latency)

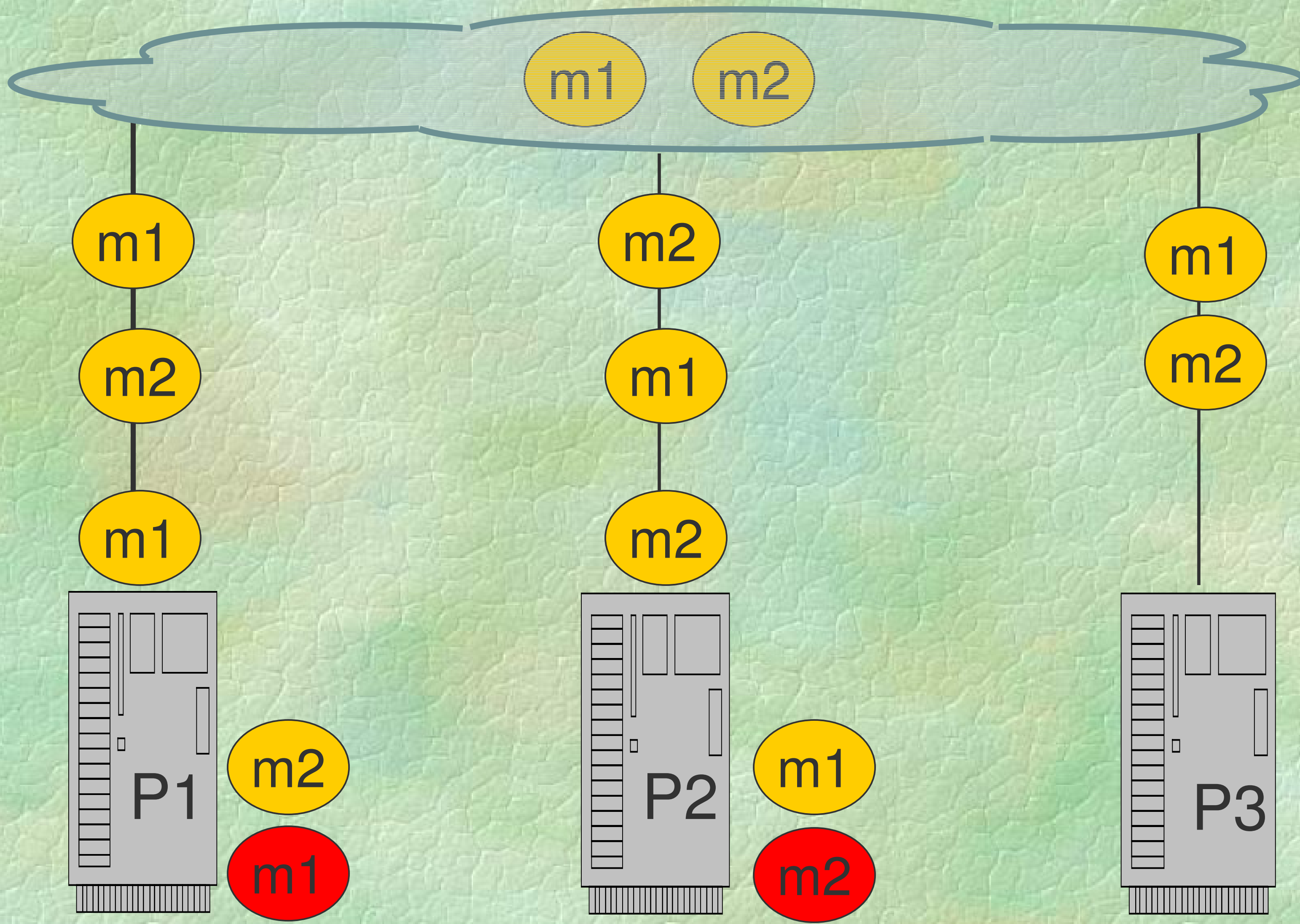
Hardware Solution

- Low latency with high throughput is achievable by means of special hardware
- Building a novel hardware, however, is expensive and time-consuming
- Is it possible to achieve the goal using off-the-shelf hardware?

Network Message Order

- Messages are ordered by the shared network
- The order may be not the same for different replicas due to:
  - Message losses
  - Internal loopback
- Neutralizing measures:
  - A flow control mechanism to minimize message losses
  - Disabling loopback

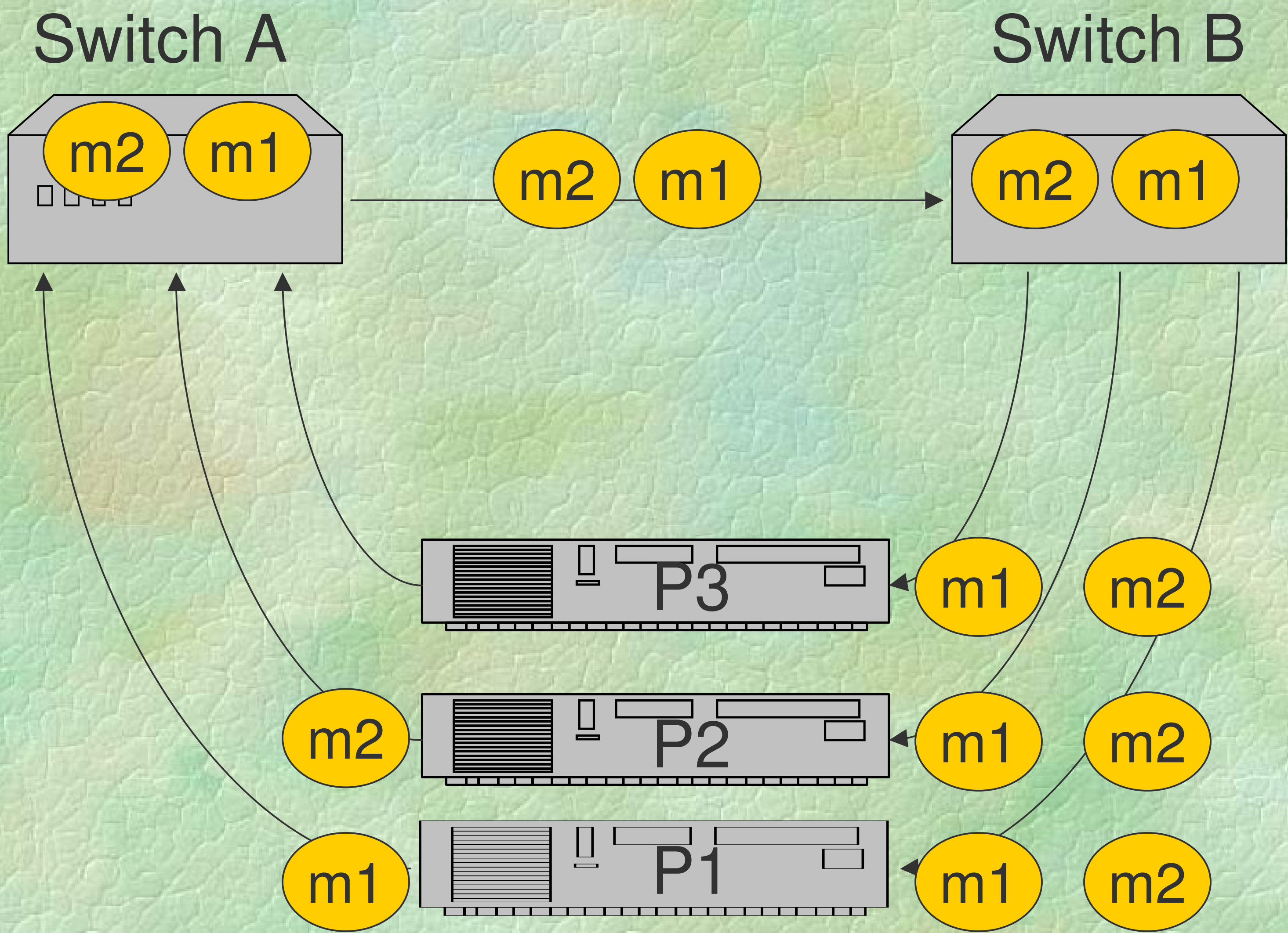
Network Order



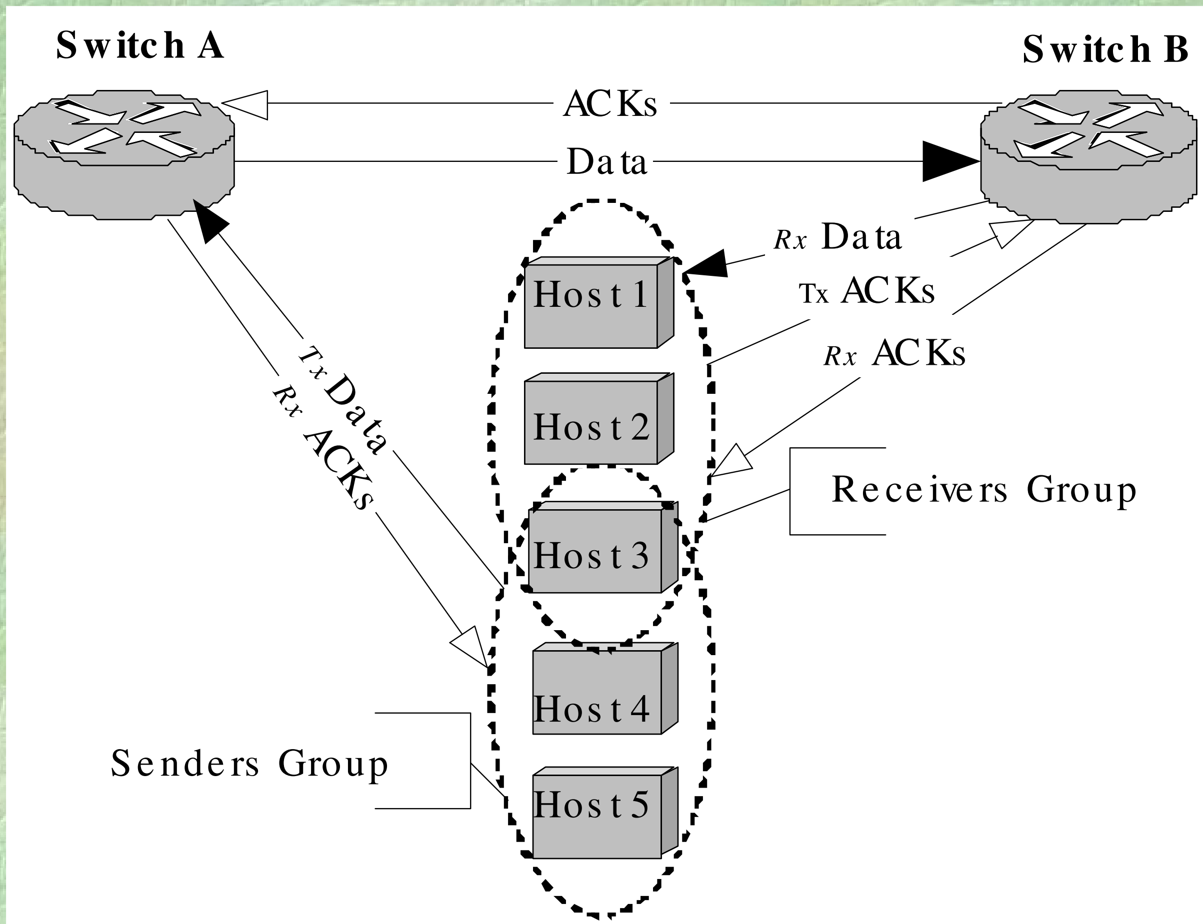
The solution is Virtual Sequencer

- Each machine is equipped with TWO Network Interface Cards (NIC1 and NIC2)
- Data is sent via NIC1
- Data is received via NIC2
- NIC1 cards are connected to switch A
- NIC2 cards are connected to switch B
- **Switch A and the link from it to switch B** serve as Virtual Sequencer

Two-Switch Network



Two-Switch Network



Implementation Issues

- Recommended Switch Features
  - IGMP snooping (highly recommended)
  - Traffic shape (recommended)
  - Jumbo frames (optional)
- Required OS Feature
  - Ability to disable Reverse Path Forward (RPF) checking

Test Bed

- 5 Pentium 500 Mhz machines with
  32 bit x 33 Mhz PCI bus
- Each machine is running Debian GNU/Linux 2.4.25
- Each machine is equipped with 2 Intel Pro1000MT Desktop Adapters
- 2 Dell 6024 switches

Definitions

- Preliminary Order (PO)
  - PO is guessed on the bases off the network order by each a machine and can be changed later.
- Uniform Total Order (UTO)
  - UTO is never changed (even for faulty machines)
  - UTO is established by collecting acknowledgments from all the machines
- Application UTO Latency:
  - UTO delivery time – Application send time

All-to-All Experiment

Number of Machines	Throughput (Mb/s)	PO Latency (ms)	UTO Latency (ms)
3	361.7	2.183	4.163
4	375.3	3.398	5.362
5	383.9	3.410	6.782

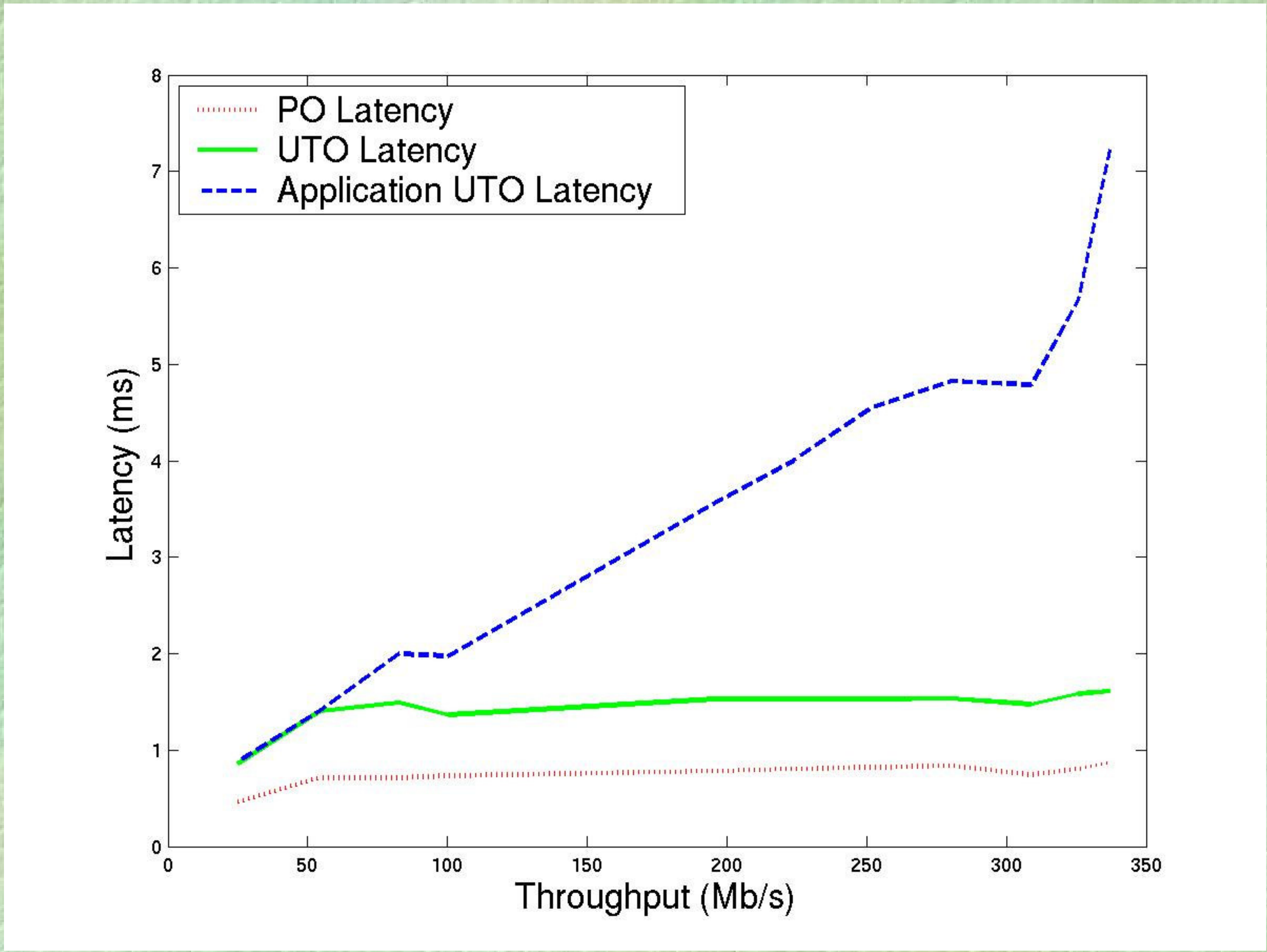
Attention: PCI bus is a real bottleneck!

Disjoint sets of senders and receivers

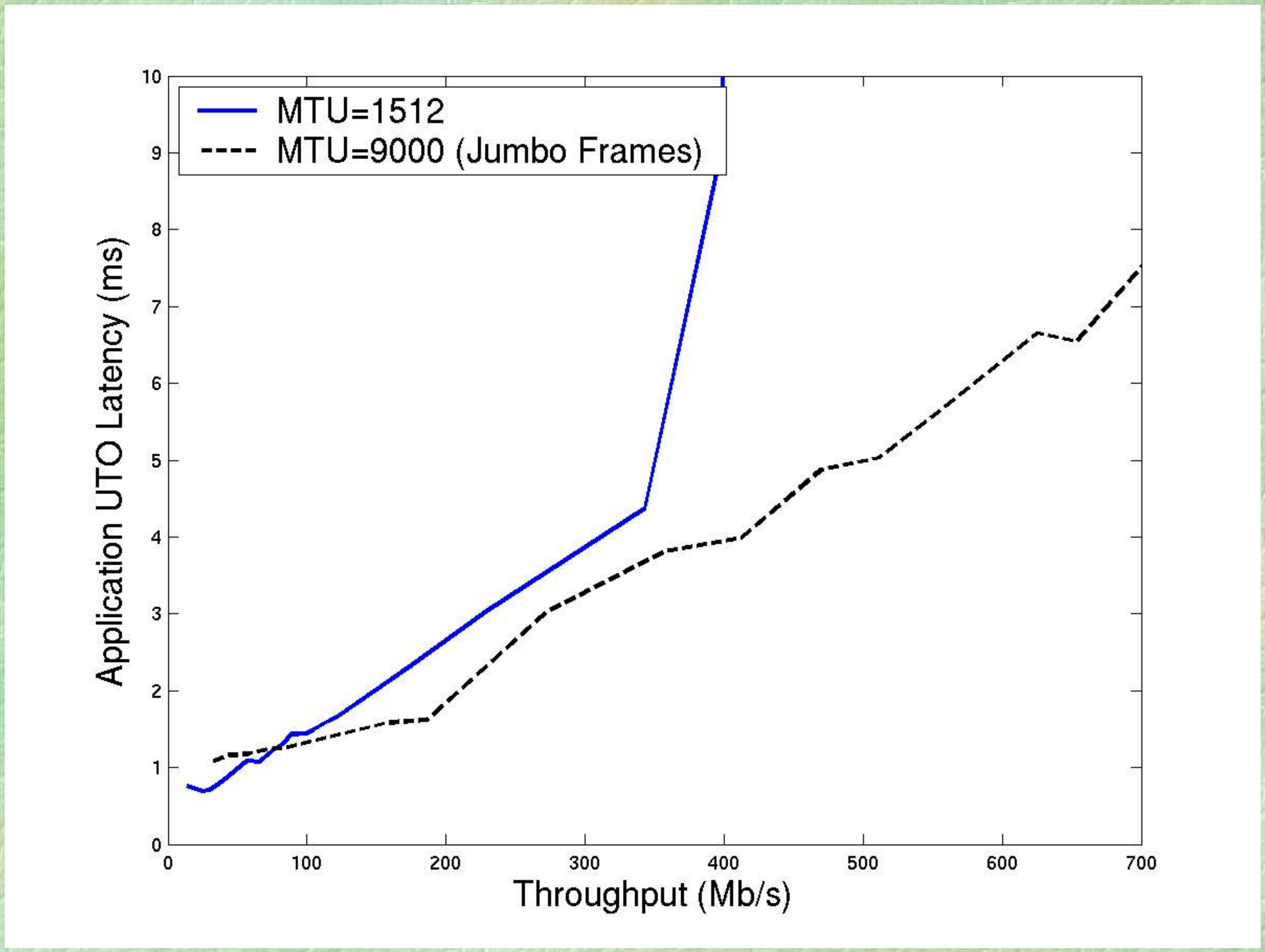
Senders	Receivers 1	2	3	4
1	505.9	495.4	489.2	476.4
2	518.3	504.8	493.7	
3	519.5	505.9		
4	519.8			

Conclusion: Scalable in number of senders. Less scalable in number of receivers due to increase in ACKs number

Latency vs. Throughput (All-to-All)



Latency vs. Throughput (disjoint sets of senders and receivers)



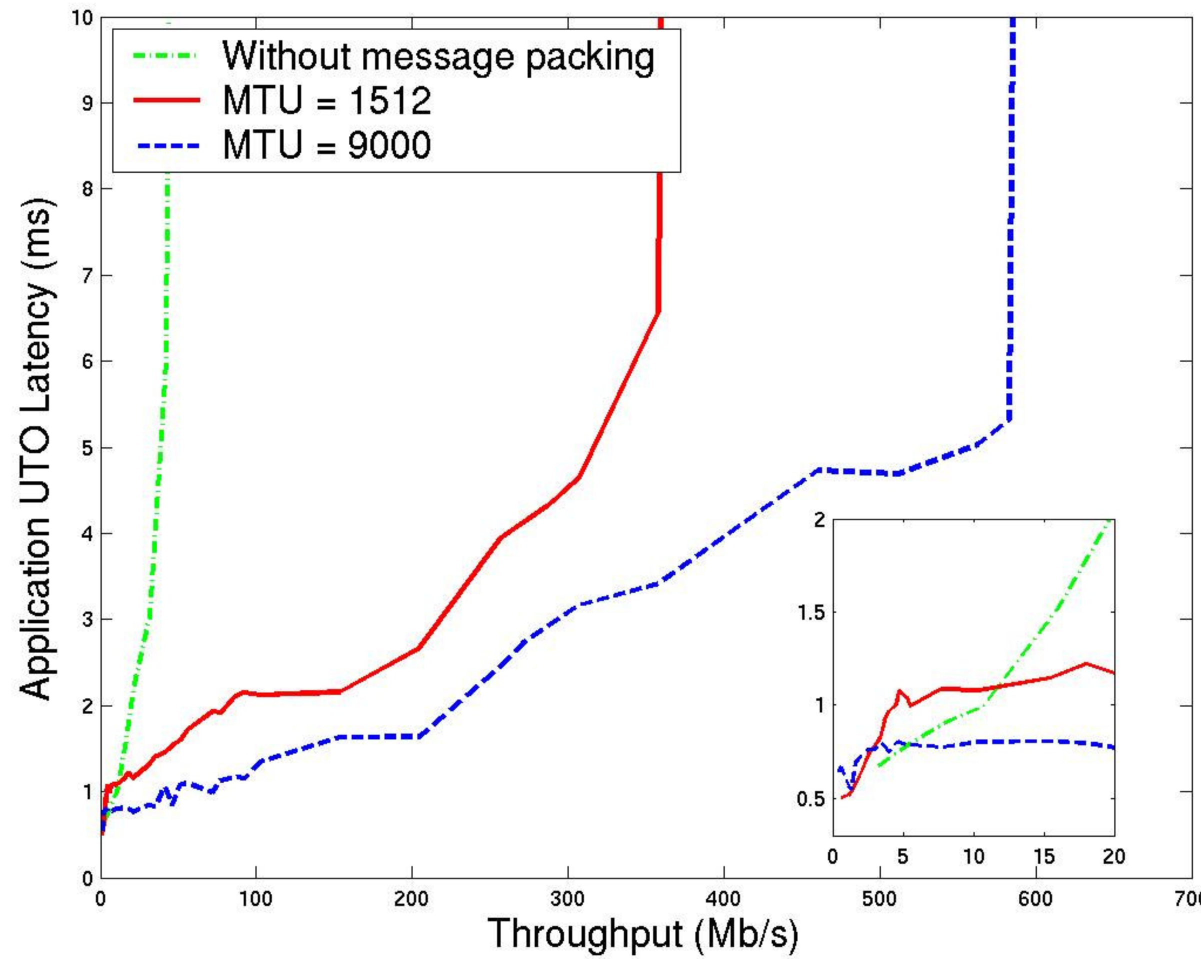
Locks and Preliminary Order

- Lock requests may be served faster based on preliminary order
- Each server may rejected/accepted a lock according to Preliminary Order
- Only iff the request order has been guessed correctly by a server, the response from the server is taken into account

Packet Aggregation

- In the experiments, results were obtained for MTU-size messages, however
- Lock requests are usually small messages
- A solution implementing simple aggregation of lock requests will increase the locking mechanism latency
- To keep the latency low, a modification of Nagle algorithm was used to perform "smart" packet aggregation.

Packet Aggregation Results



Future Work

- Obtaining Patent
- Adapting to other networks
- Implementing:
  - efficient fault detection and fault tolerance
- Using NICs with CPU to:
  - implement zero-copy driver
  - send and aggregate acknowledgments by NIC
- Checking scalability