Hardware- and Software-Based Collective Communication on the Quadrics Network

Fabrizio Petrini*, Salvador Coll*†, Eitan Frachtenberg* and Adolfy Hoisie*

*CCS-3 Modeling, Algorithms, and Informatics

Los Alamos National Laboratory

†Technical University of Valencia - SPAIN

scoll@lanl.gov

Outline

- Introduction
- Quadrics network design overview
  - Hardware
  - Communication/programming libraries
- Collective communication on the QsNET
- Barrier synchronization
- Broadcast
- Performance analysis
  - Experimental framework
  - Results
- Conclusions

Los Alamos NATIONAL LABORATORY

Introduction

- The efficient implementation of collective communication is a challenging design effort
- Very important to guarantee scalability of barrier synchronization, broadcast, gather, scatter, reduce, etc.
- Essential to implement system primitives to enhance fault-tolerance.
- Software or hardware support for multicast communication can improve the performance and resource utilization of a parallel computer
  - Software multicast: based on unicast messages, simple to implement, no network topology constraint, slower
  - Hardware multicast: require dedicated hardware, network dependent, faster

Los Alamos NATIONAL LABORATORY

Introduction

- Some of the most powerful systems in the world use the Quadrics interconnection network and the collective communication services analyzed in this job:
  - The Terascale Computing System (TCS) at the Pittsburgh Supercomputing Center – the second most powerful computer in the world

Introduction

- Some of the most powerful systems in the world use the Quadrics interconnection network and the collective communication services analyzed in this job:
  - The Terascale Computing System (TCS) at the Pittsburgh Supercomputing Center – the second most powerful computer in the world

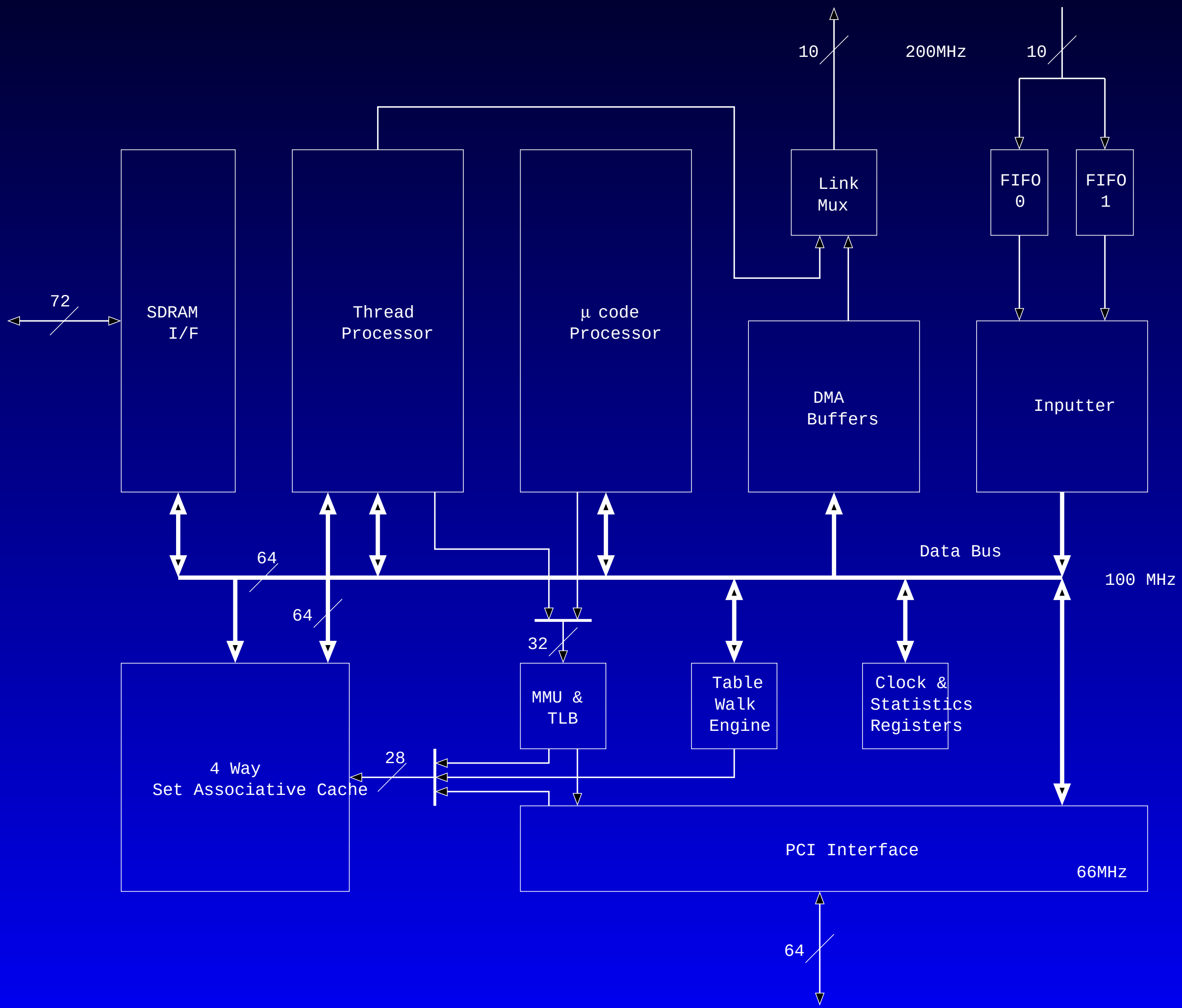
Introduction

- Some of the most powerful systems in the world use the Quadrics interconnection network and the collective communication services analyzed in this job:
  - The Terascale Computing System (TCS) at the Pittsburgh Supercomputing Center – the second most powerful computer in the world
  - ASCI Q machine, currently under development at Los Alamos National Laboratory (30 TeraOps, expected to be delivered by the end of 2002)

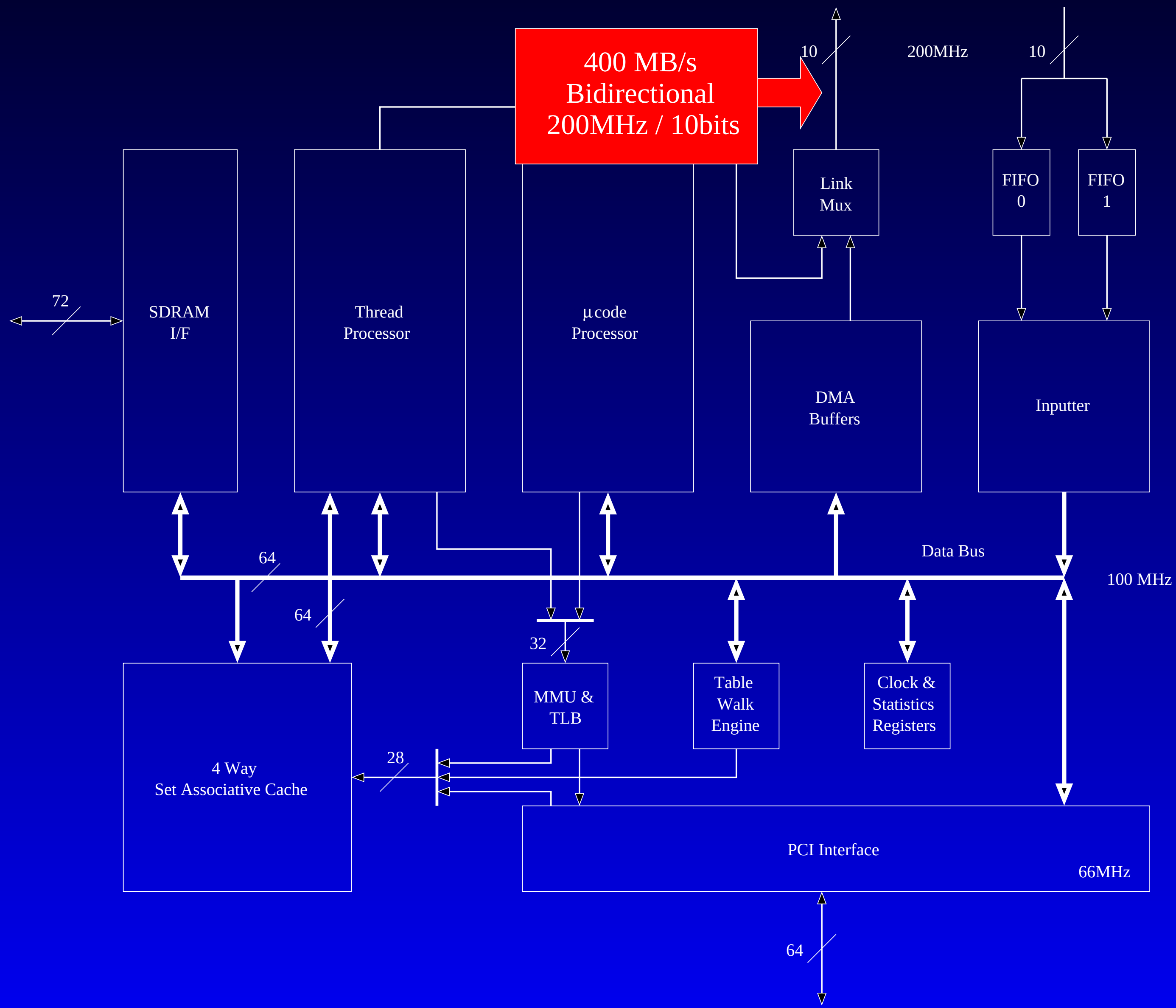
Quadrics Network Design Overview

- QsNET provides an abstraction of distributed virtual shared memory
- Each process can map a portion of its address space into the global memory
- These address spaces constitutes the virtual shared memory
- This shared memory is fully integrated with the native operating system
- Based on two building blocks:
  - a network interface card called Elan
  - a crossbar switch called Elite

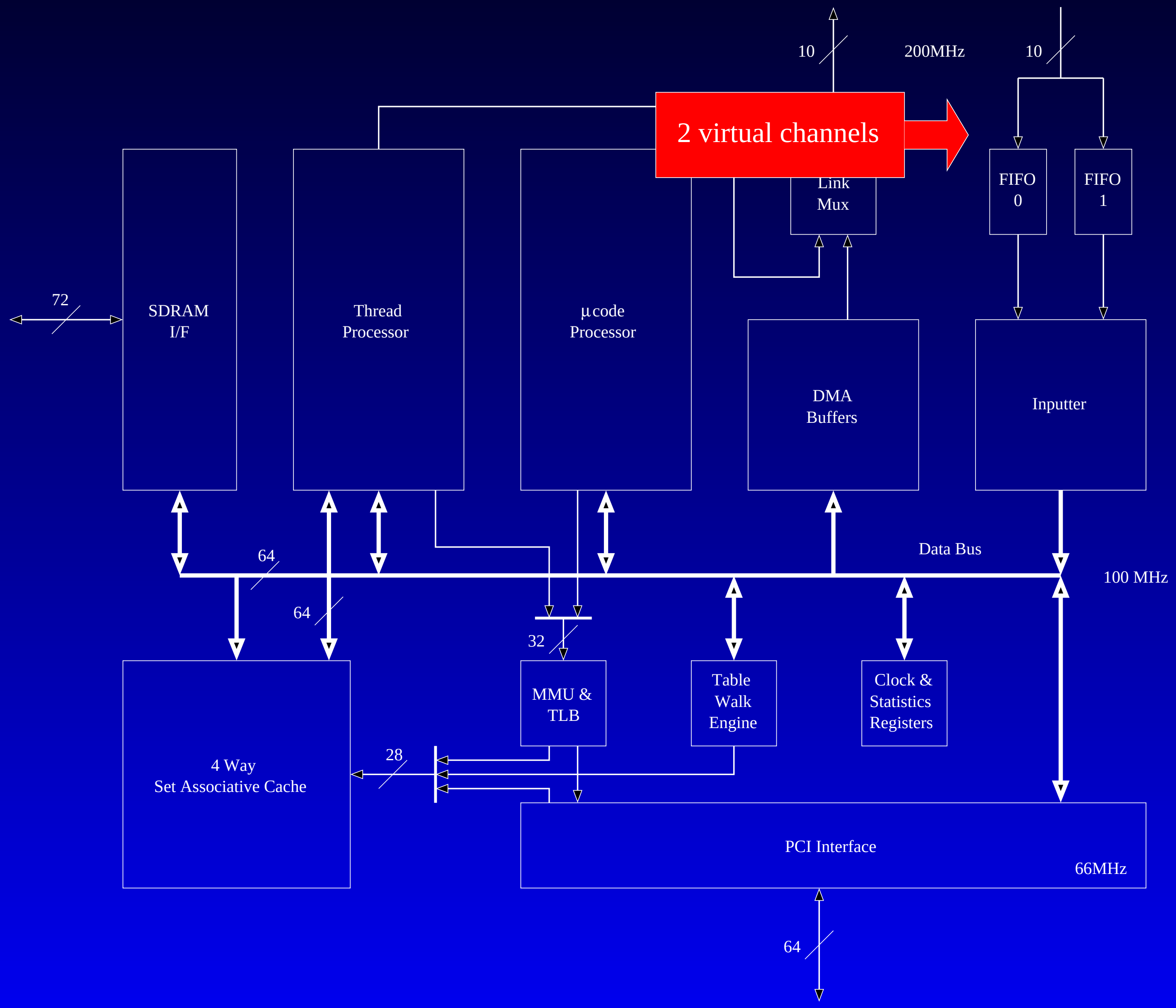
Elan


10
200MHz
10
Link Mux
FIFO 0
FIFO 1
72
SDRAM I/F
Thread Processor
µ code Processor
DMA Buffers
Inputter
Data Bus
100 MHz
64
64
32
MMU & TLB
Table Walk Engine
Clock & Statistics Registers
4 Way Set Associative Cache
28
PCI Interface
66MHz
64

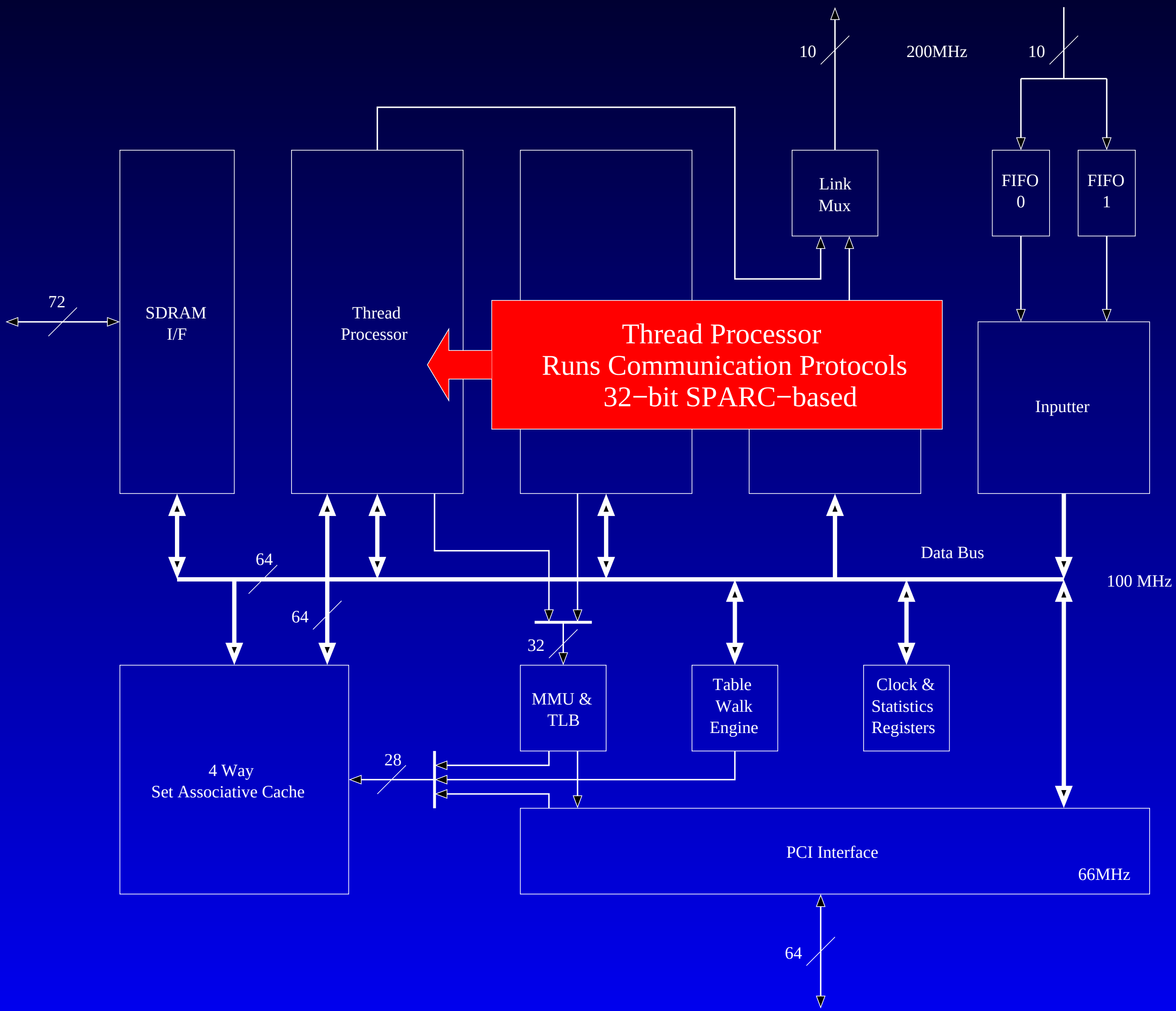
Elan


400 MB/s
Bidirectional
200MHz / 10bits
10
200MHz
10
Link
Mux
FIFO
0
FIFO
1
72
SDRAM
I/F
Thread
Processor
µcode
Processor
DMA
Buffers
Inputter
64
Data Bus
100 MHz
64
32
MMU &
TLB
Table
Walk
Engine
Clock &
Statistics
Registers
4 Way
Set Associative Cache
28
PCI Interface
66MHz
64

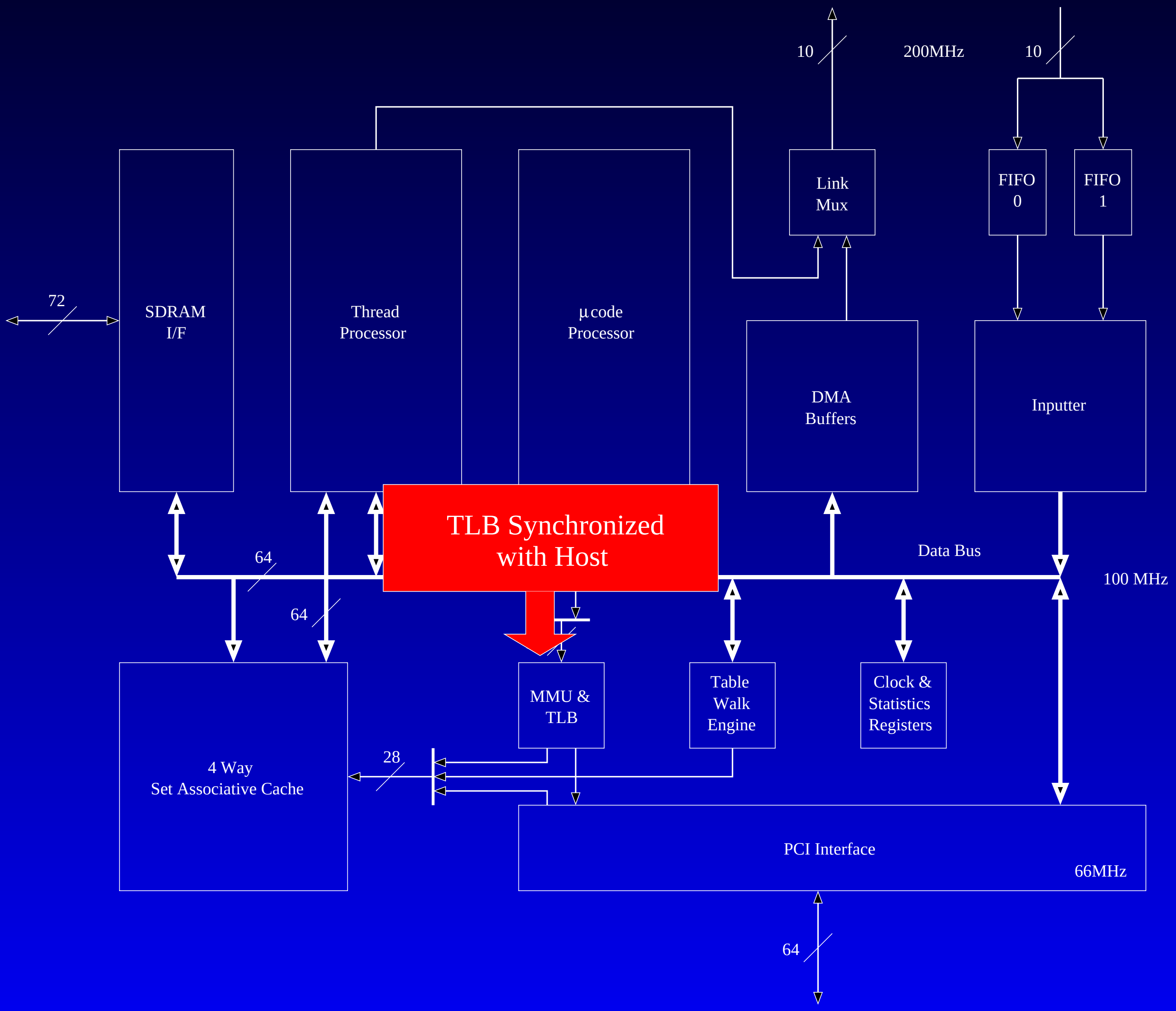
Elan


10
200MHz
10
2 virtual channels
Link
Mux
FIFO
0
FIFO
1
72
SDRAM
I/F
Thread
Processor
µcode
Processor
DMA
Buffers
Inputter
Data Bus
64
100 MHz
64
32
MMU &
TLB
Table
Walk
Engine
Clock &
Statistics
Registers
4 Way
Set Associative Cache
28
PCI Interface
66MHz
64

Elan

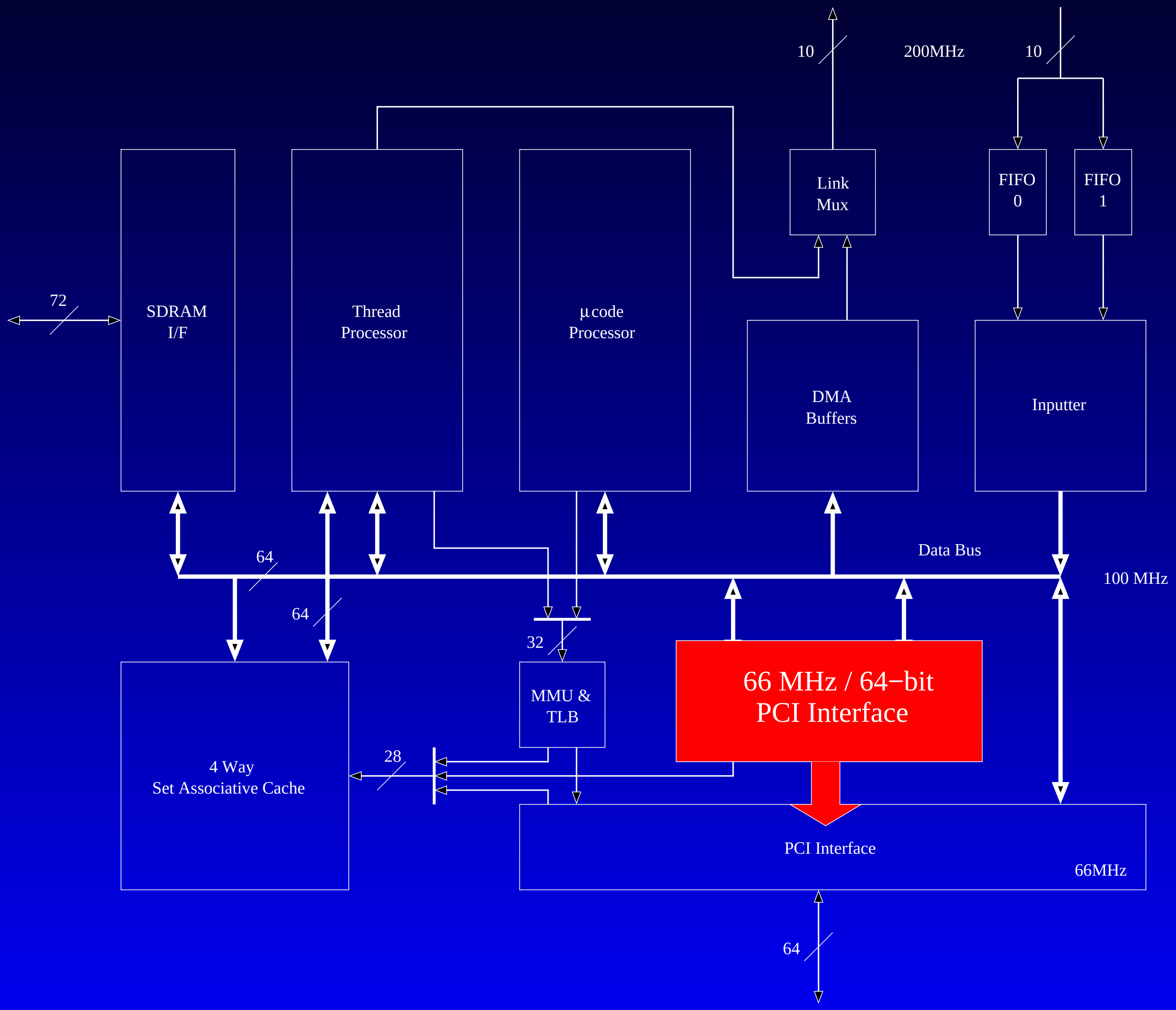

10
200MHz
10
Link Mux
FIFO 0
FIFO 1
72
SDRAM I/F
Thread Processor
Thread Processor
Runs Communication Protocols
32−bit SPARC−based
Inputter
64
Data Bus
100 MHz
64
32
MMU & TLB
Table Walk Engine
Clock & Statistics Registers
4 Way
Set Associative Cache
28
PCI Interface
66MHz
64

Elan



Los Alamos NATIONAL LABORATORY

Elan

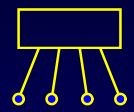


10
200MHz
10
Link
Mux
FIFO
0
FIFO
1
72
SDRAM
I/F
Thread
Processor
µcode
Processor
DMA
Buffers
Inputter
Data Bus
64
100 MHz
64
32
MMU &
TLB
66 MHz / 64−bit
PCI Interface
4 Way
Set Associative Cache
28
PCI Interface
66MHz
64

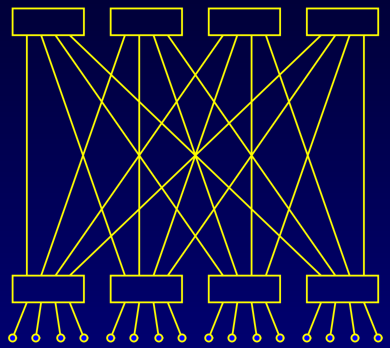
Elite

- 8 bidirectional links with 2 virtual channels in each direction
- An internal 16x8 full crossbar switch
- 400 MB/s on each link direction
- Packet error detection and recovery, with routing and data transactions CRC protected
- 2 priority levels plus an aging mechanism
- Adaptive routing
- Hardware support for broadcast

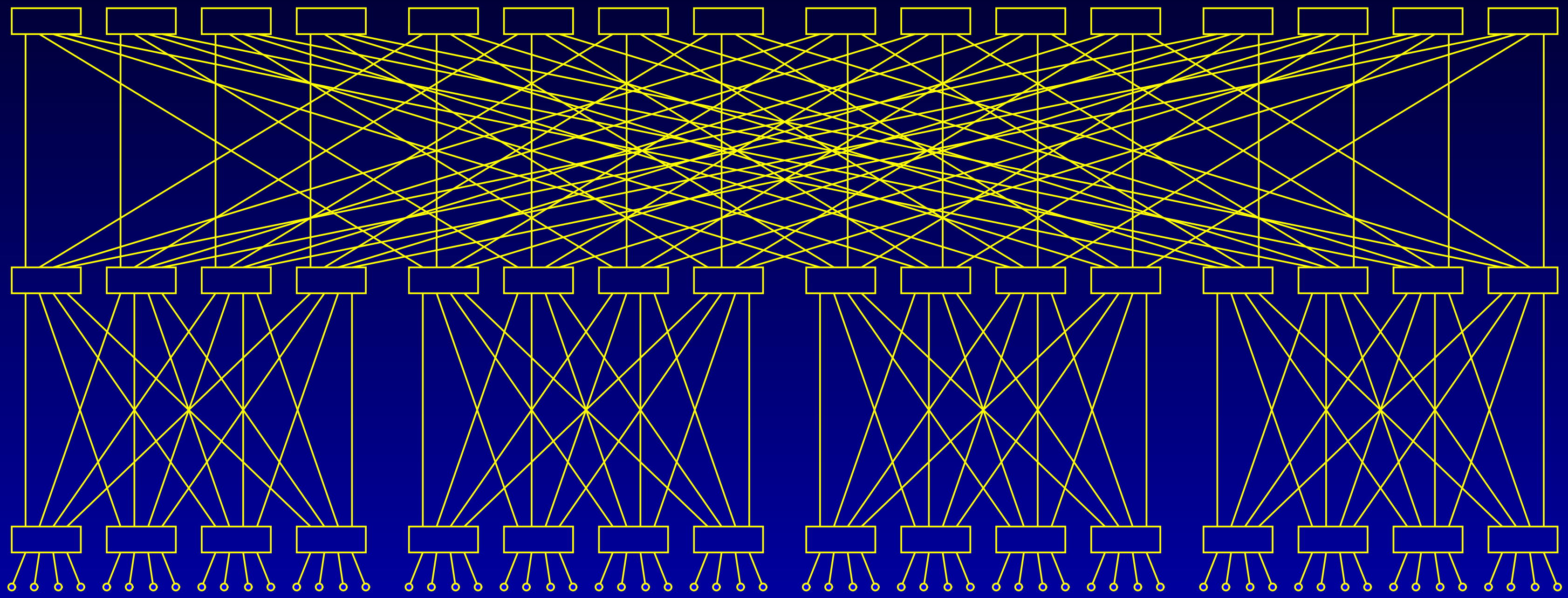
Network Topology: Quaternary Fat-Tree



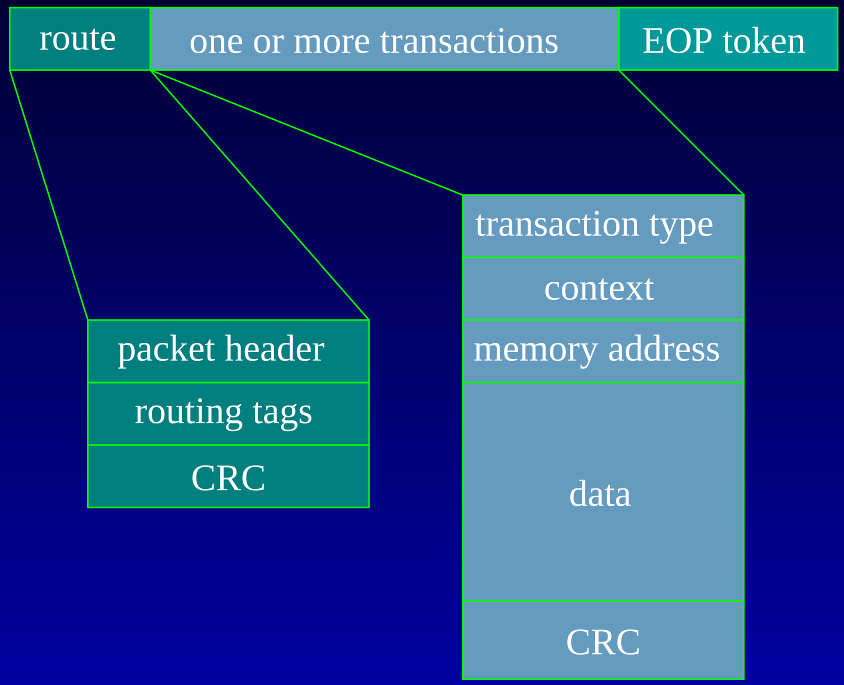
Network Topology: Quaternary Fat-Tree



Network Topology: Quaternary Fat-Tree



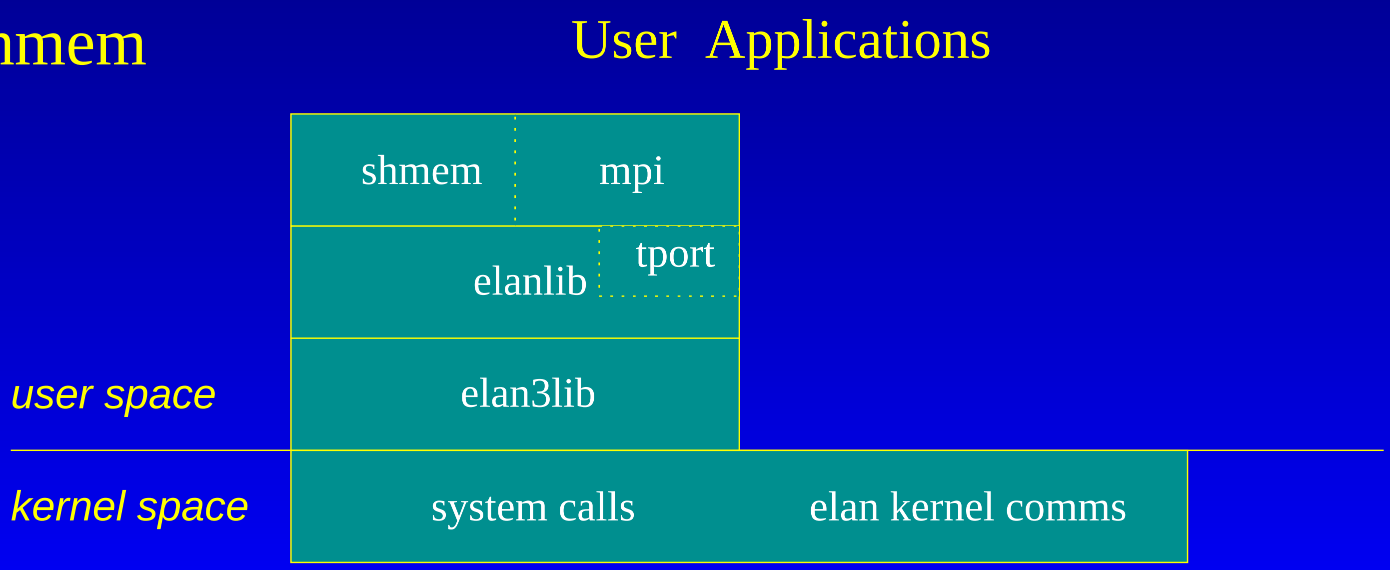
Packet Format



- 320 bytes data payload (5 transactions with 64 bytes each)
- 74-80 bytes overhead

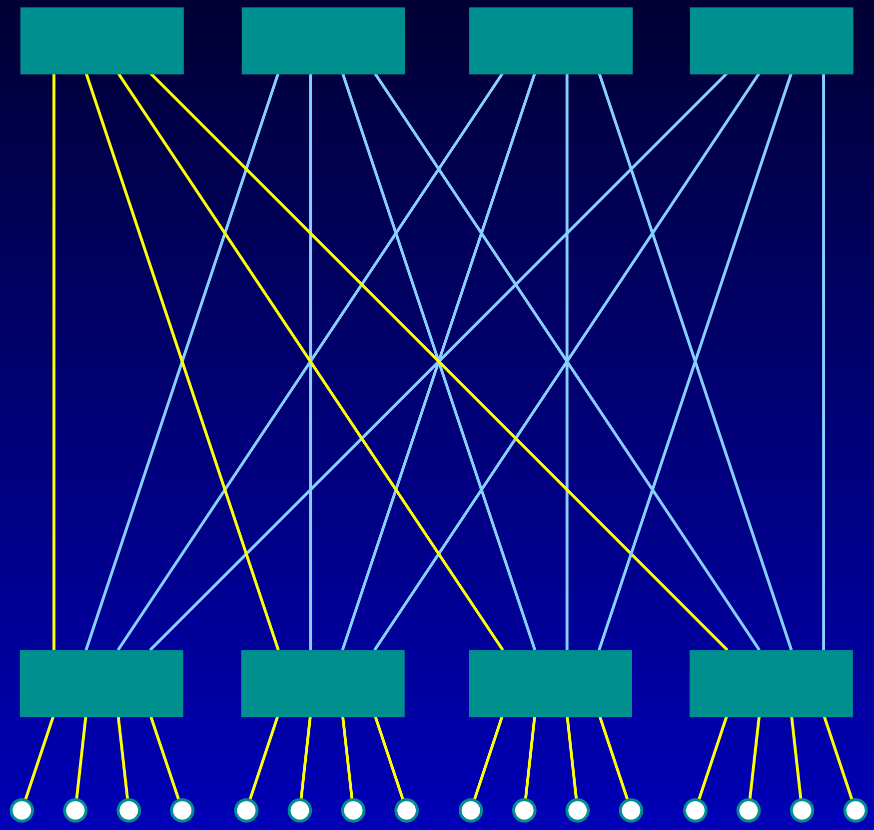
Programming Libraries

- Elan3lib
  - event notification
  - memory mapping and allocation
  - remote DMA
- Elanlib and Tports
  - collective communication
  - tagged message passing
- MPI, shmem



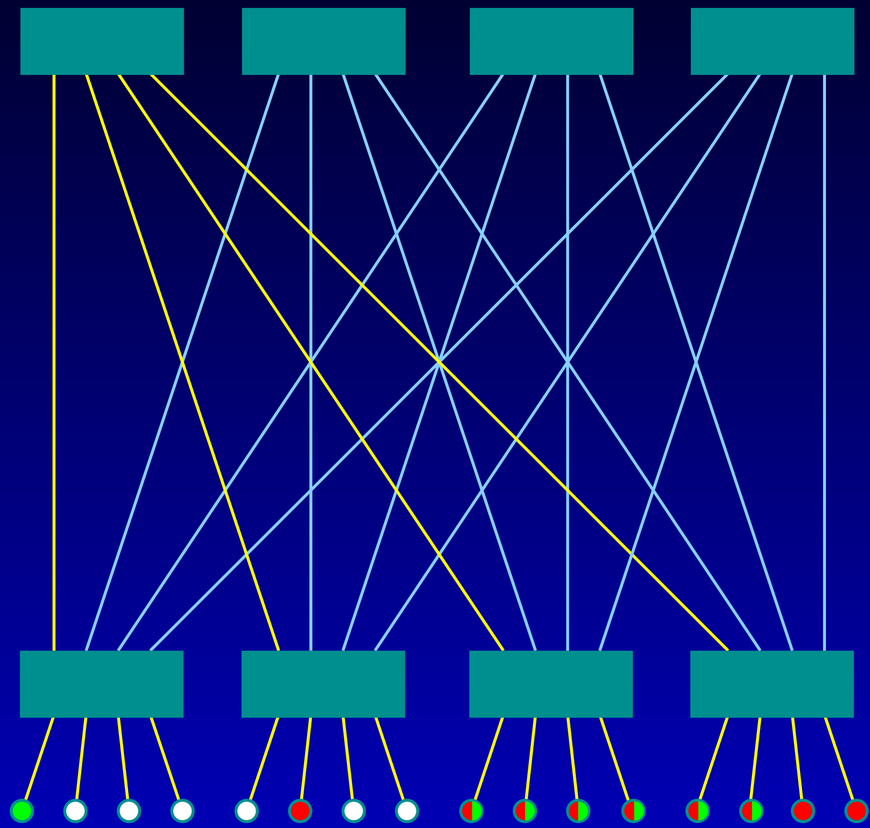
Los Alamos
NATIONAL LABORATORY

Collective communication on the QsNET

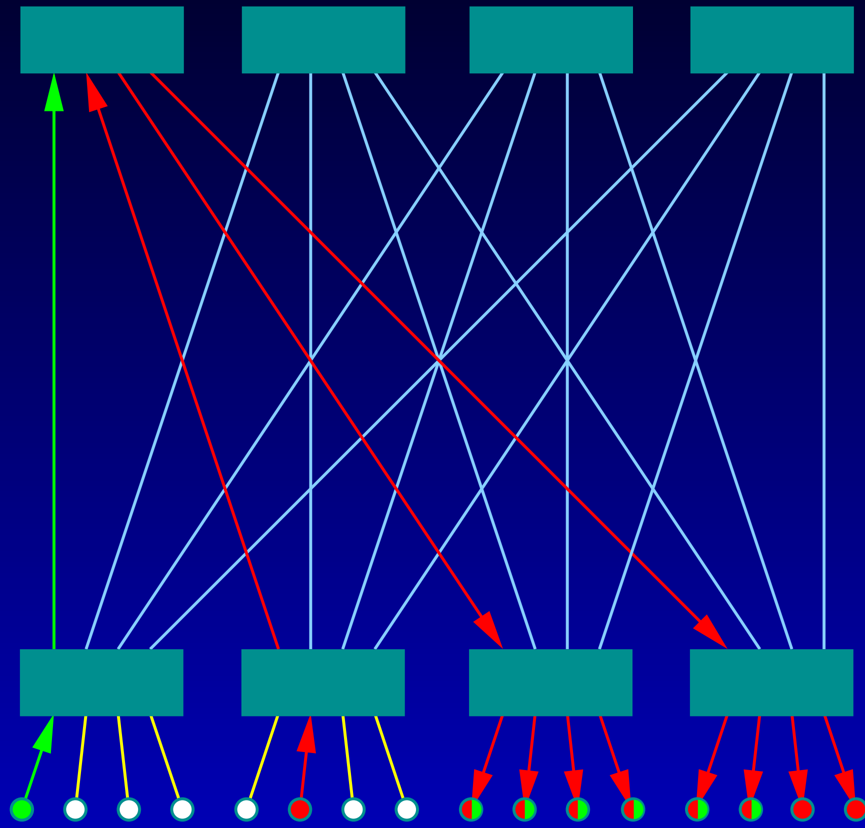


Broadcast tree for a 16-node network

Collective communication on the QsNET

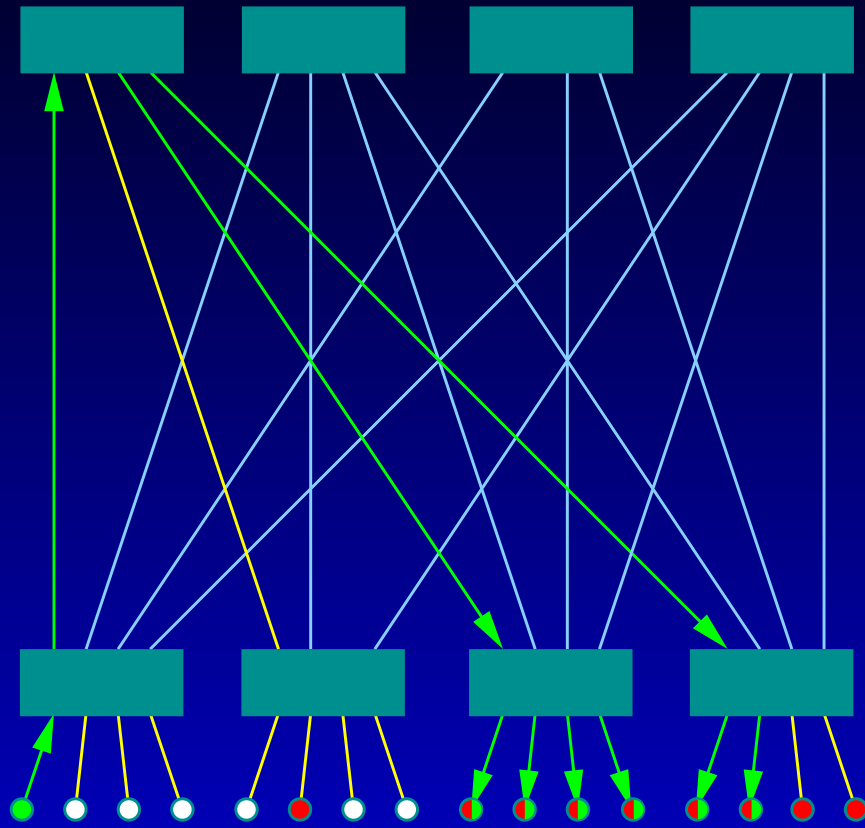


Collective communication on the QsNET



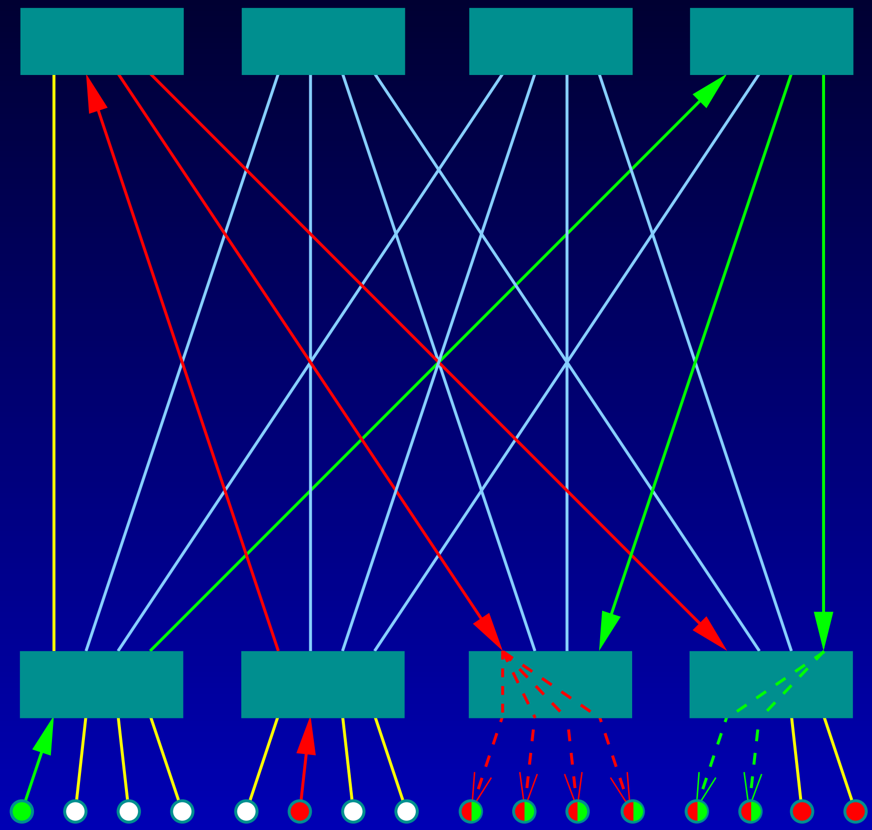
Serialization through the root switch to avoid deadlocks

Collective communication on the QsNET



Los Alamos NATIONAL LABORATORY

Collective communication on the QsNET



Deadlocked situation

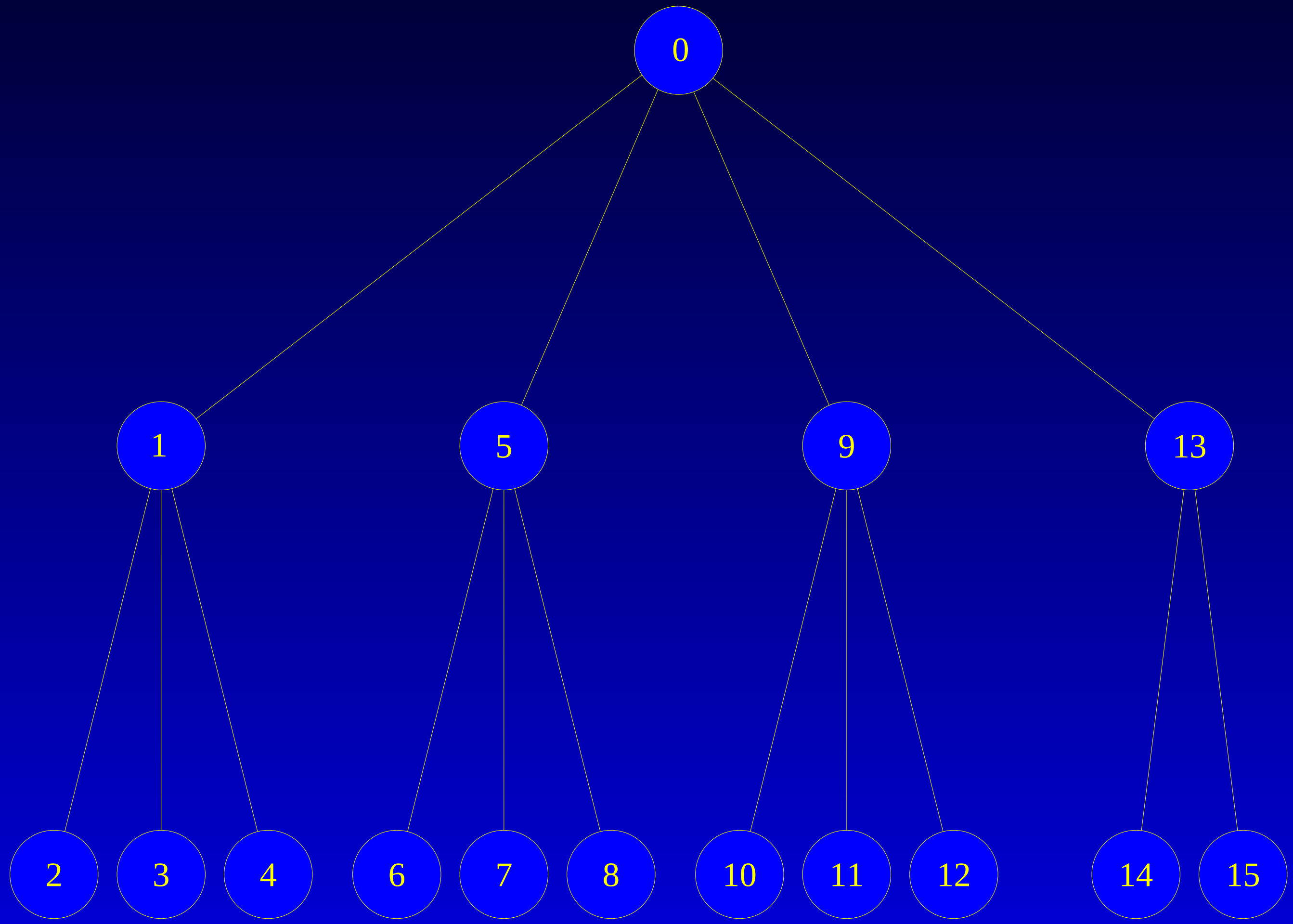
Barrier Synchronization

QsNET implements two synchronization primitives:

- Software-based: it uses a balanced tree and point-to-point messages
  - `elan_gsync()`
- Hardware-based: it uses the hardware multicast support
  - `elan_hgsync()`: busy-wait
  - `elan_hgsyncevent()`: event-based

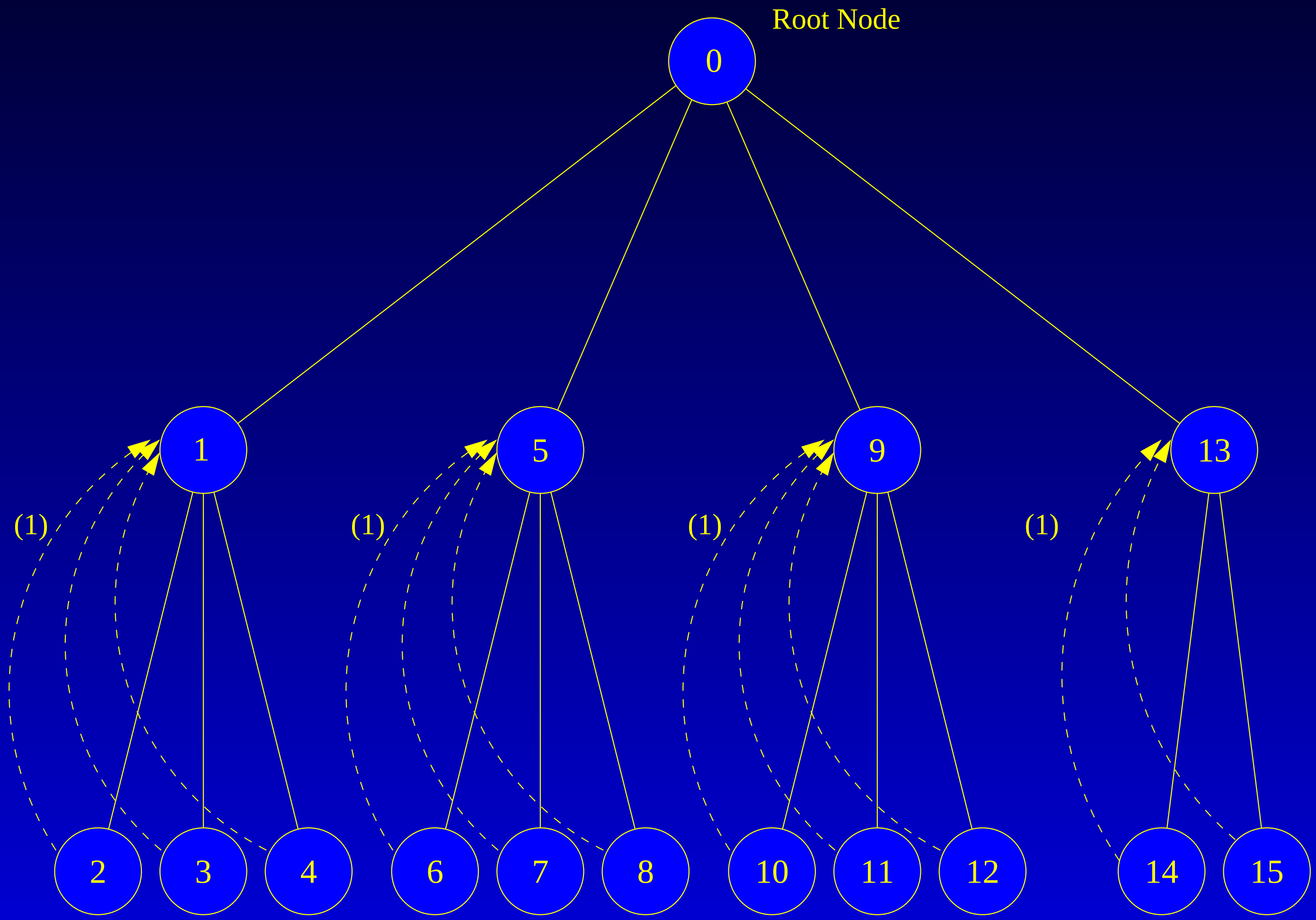
Software-Based Barrier

Each process waits for ’ready’ signals from its children



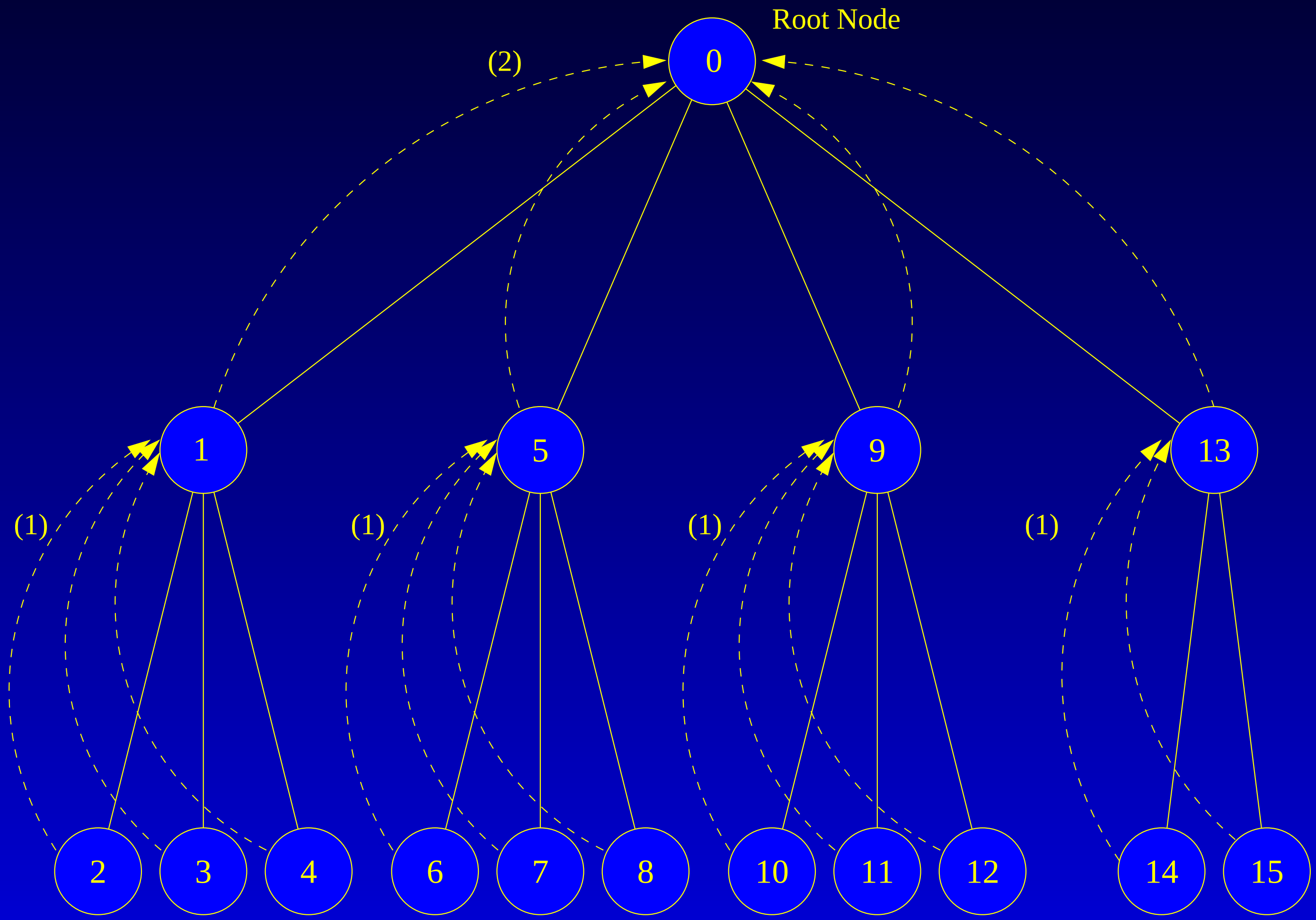
Software-Based Barrier

Each process waits for 'ready' signals from its children (1) ...

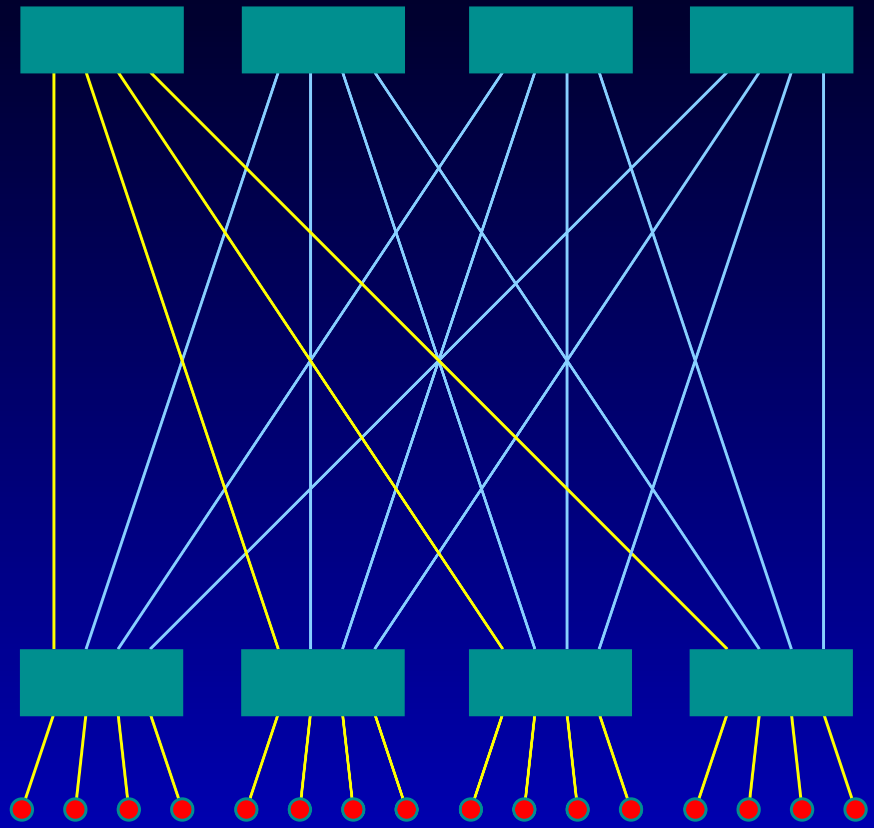


Software-Based Barrier

... and sends its own signal up to the parent process (2)

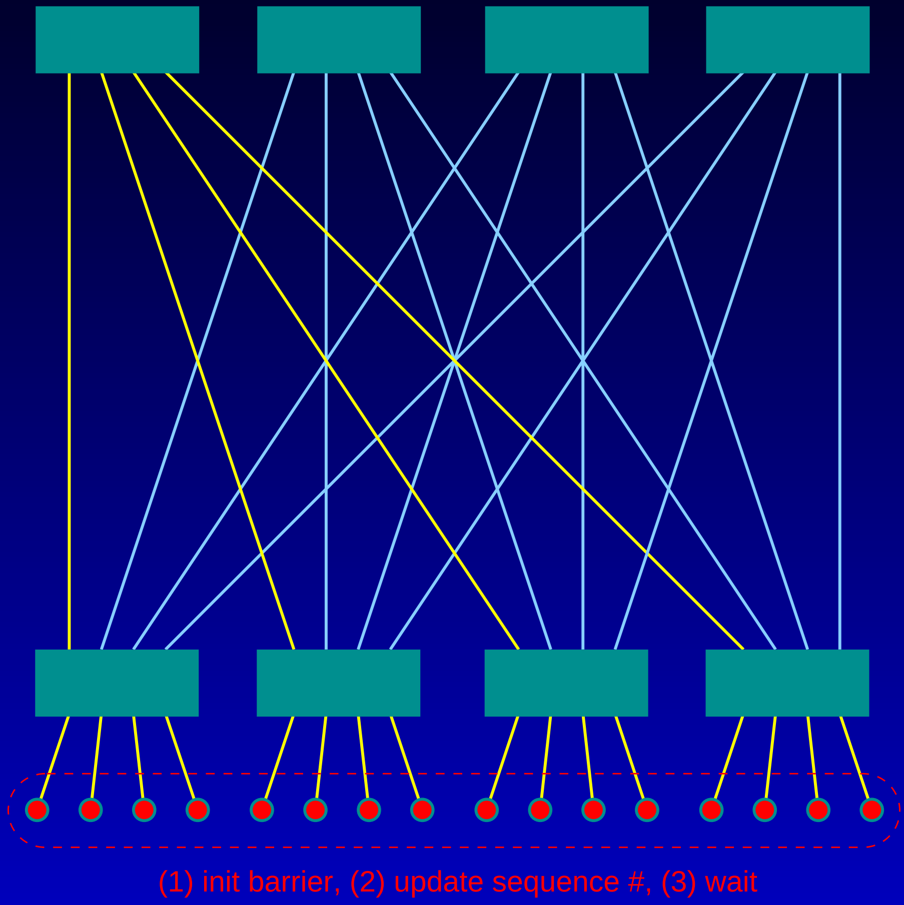


Hardware-Based Barrier



Example for 16 nodes

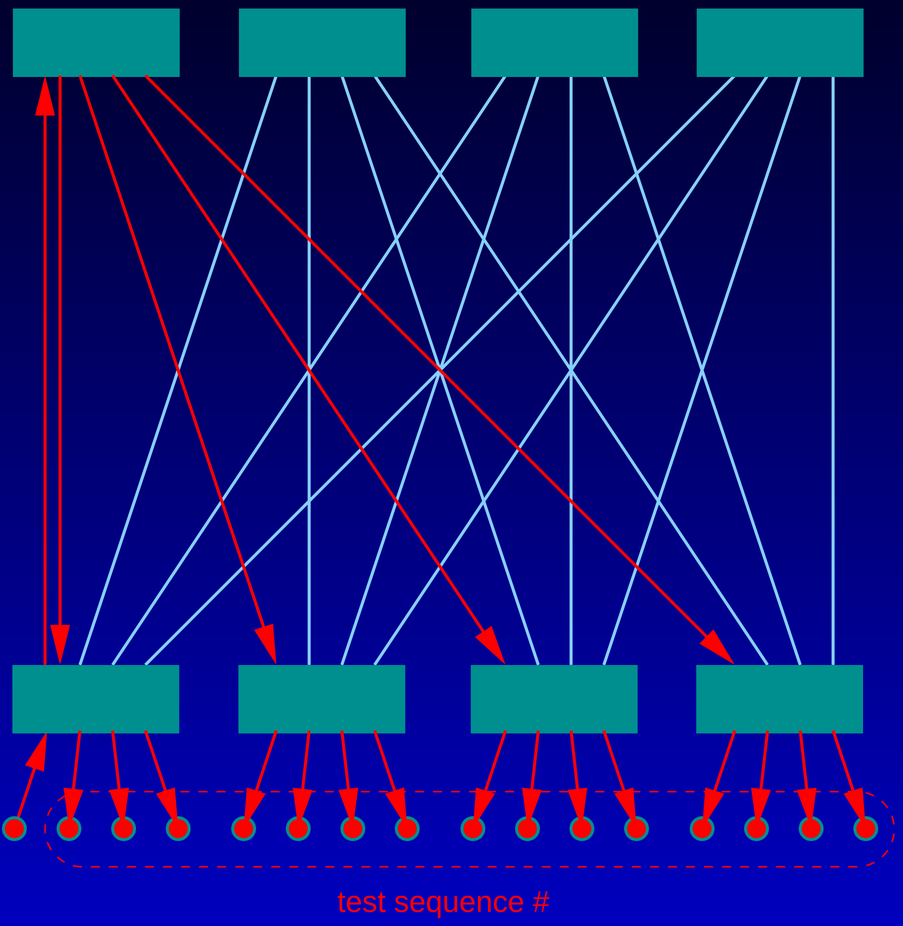
Hardware-Based Barrier



Init barrier

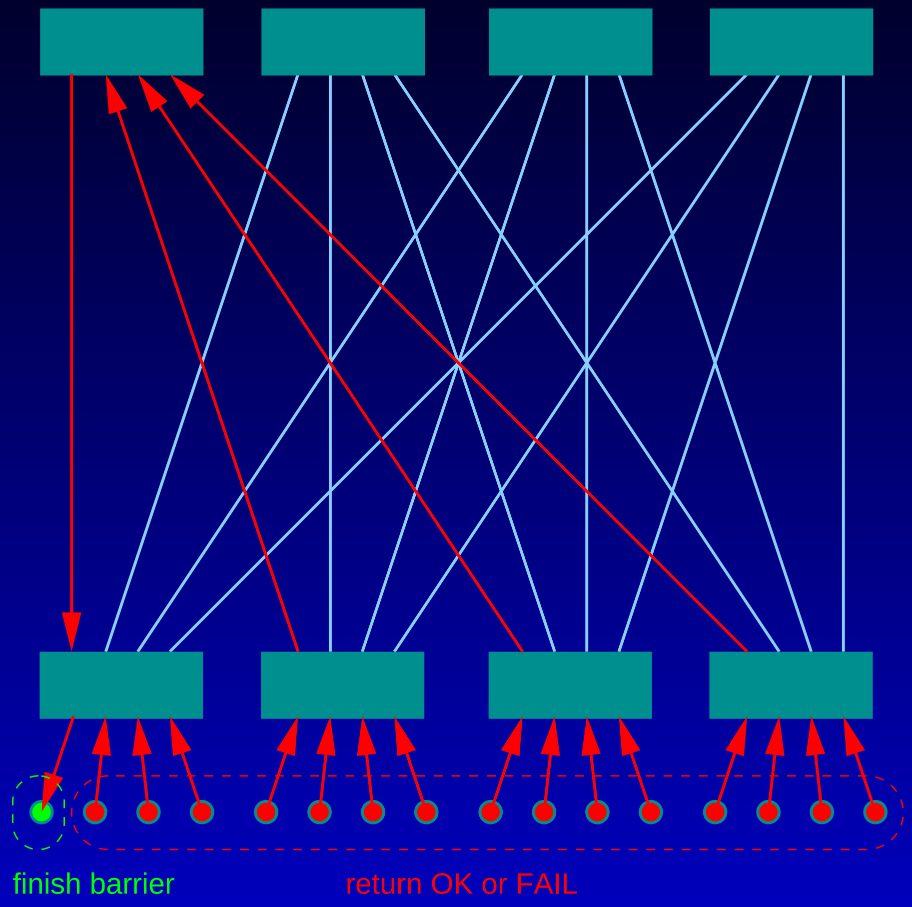
Los Alamos NATIONAL LABORATORY

Hardware-Based Barrier



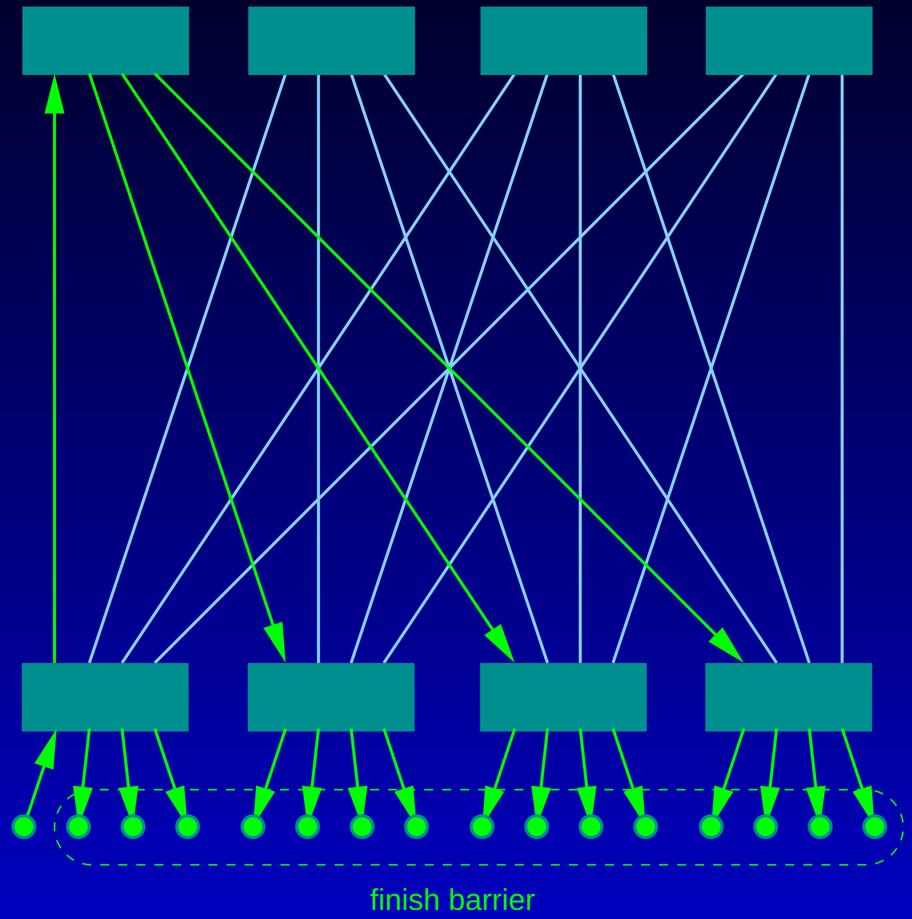
Multicast transaction

Hardware-Based Barrier



Acknowledgment

Hardware-Based Barrier



Final 'EOP' (End-Of-Packet) token

Broadcast

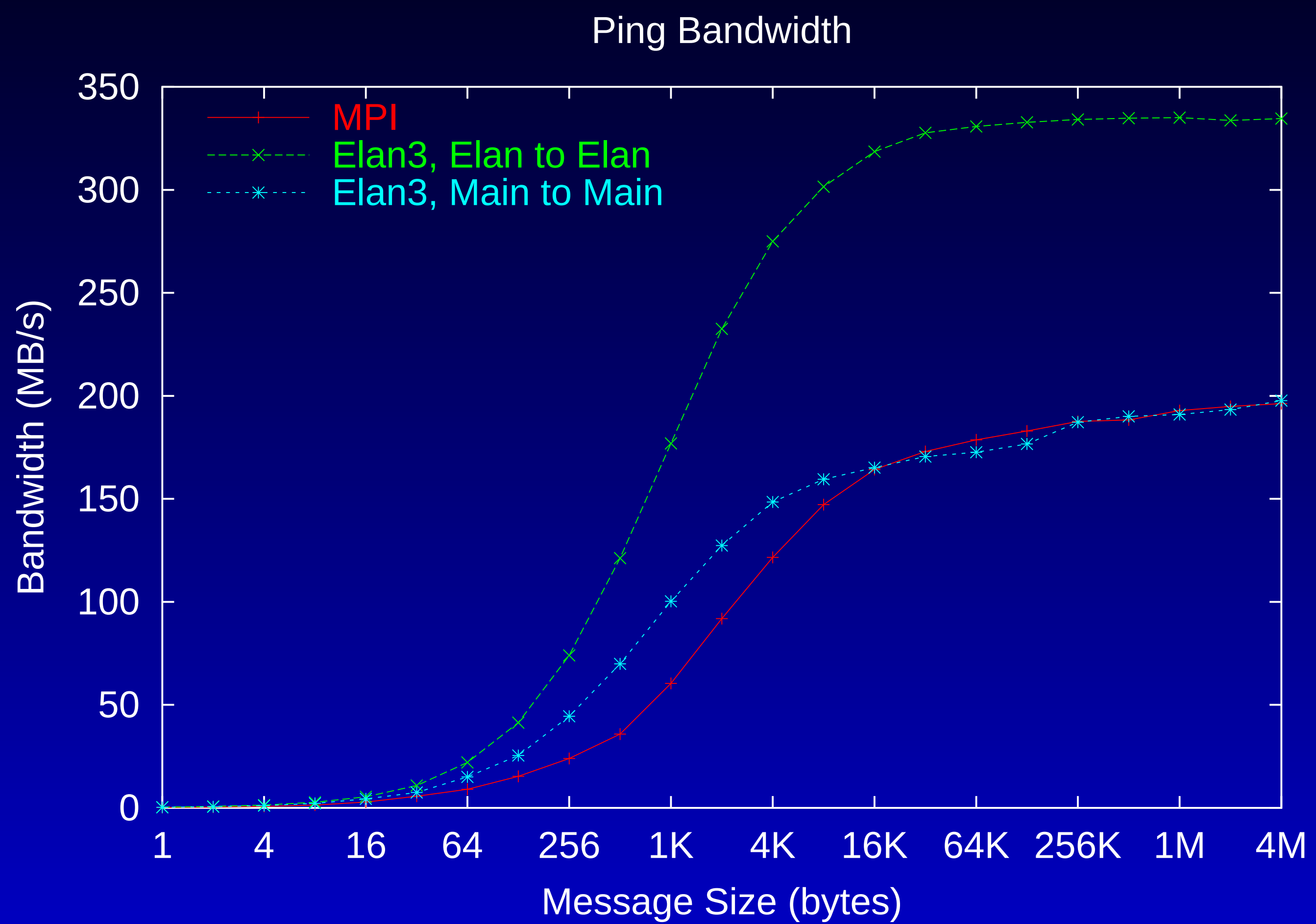
QsNET implements two broadcast primitives:

- Software-based: it uses a balanced tree and point-to-point messages
    - `elan_bcast()`
- Hardware-based: it uses the hardware multicast support
    - `elan_hbcast()`
- Both implementations perform an initial barrier to guarantee resources allocation

Performance Analysis

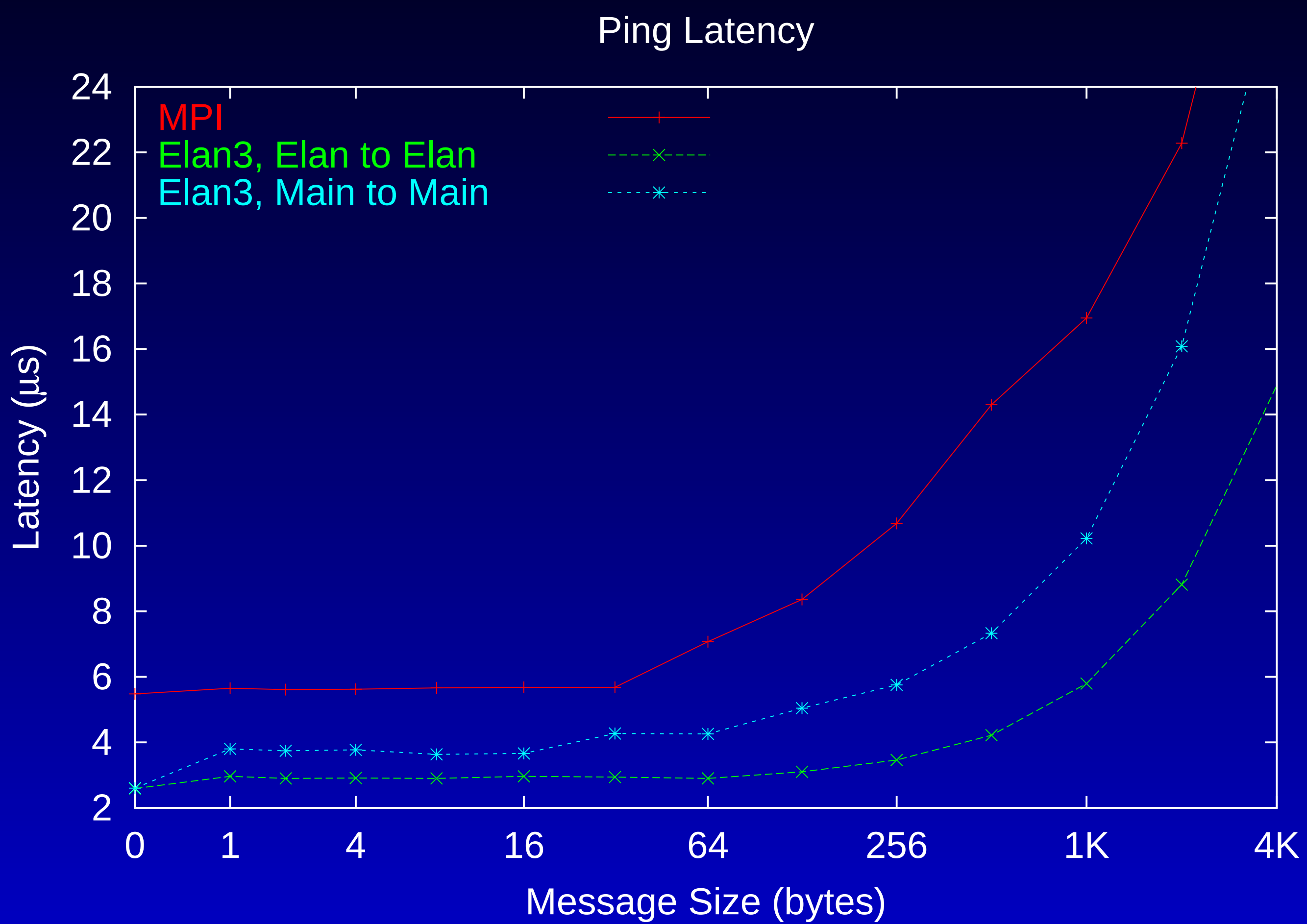
- The experimental results are obtained on a 64-node cluster of Compaq AlphaServer ES40s running Tru64 Unix.
- Each Alpahserver is attached to a quaternary fat-tree of dimension three through a 64 bit, 33 MHz PCI bus using the Elan3 card.
- In order to expose the real network performance, we place the communication buffers in Elan memory.
- We present:
  - unidirectional ping results, as a reference, and
  - barrier and broadcast results, analyzing the effect of additional background traffic

Unidirectional Ping



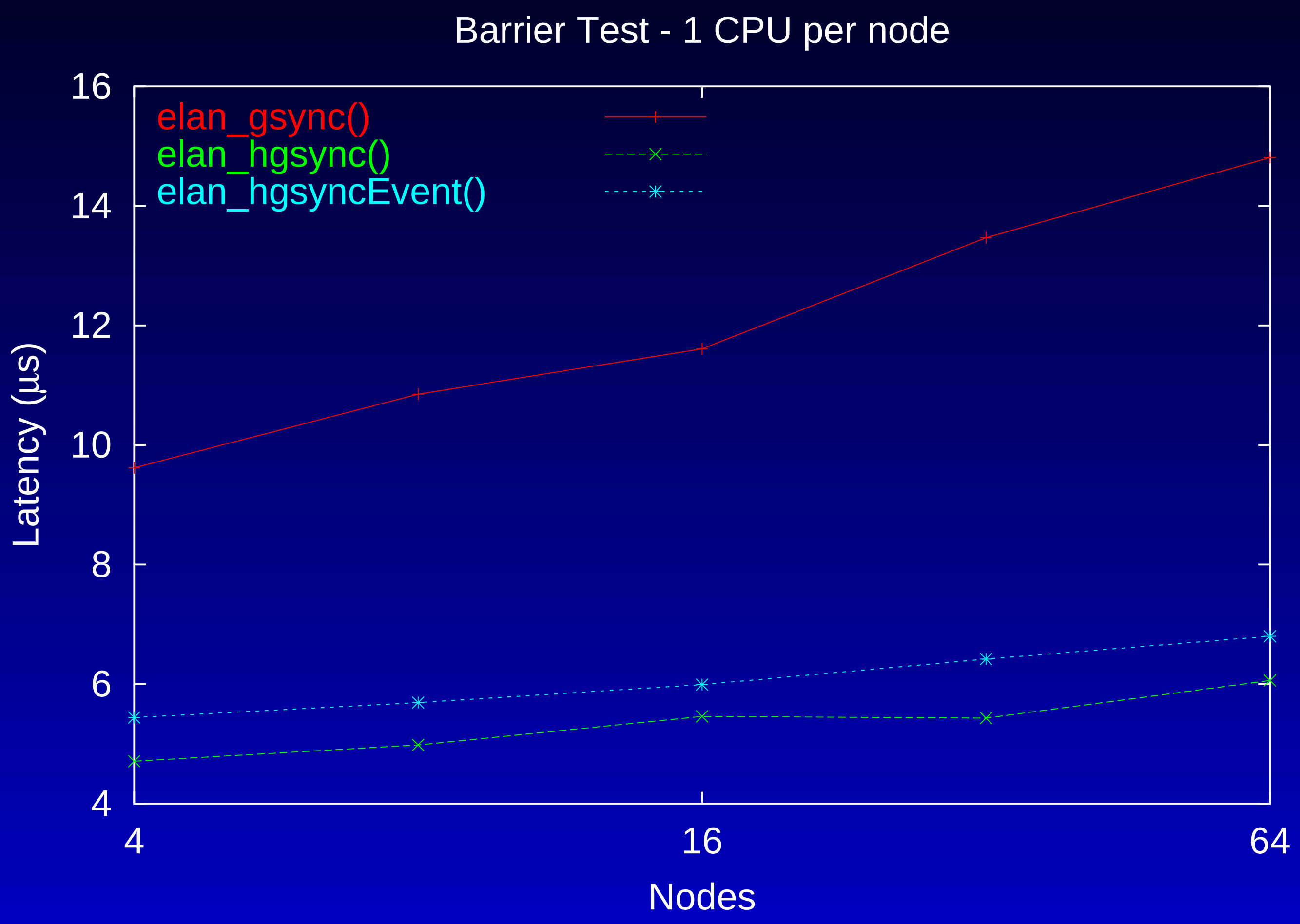
- Peak data bandwidth (Elan to Elan) of 335 MB/s $\simeq$ 396 MB/s (99% of nominal bandwidth)
- Main to main asymptotic bandwidth of 200 MB/s

Unidirectional Ping



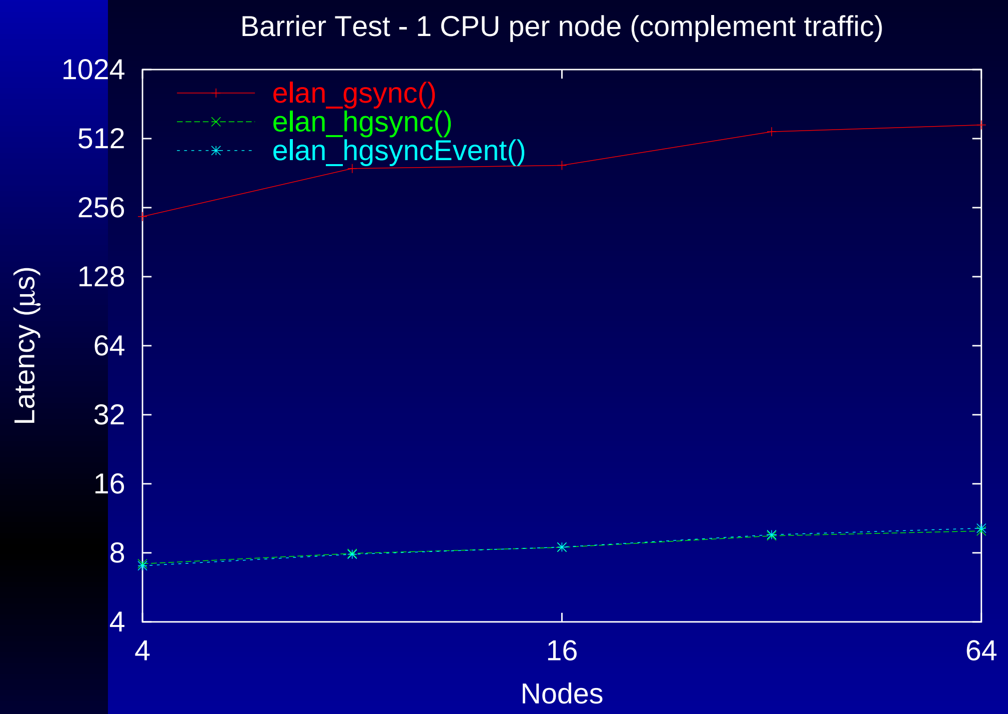
- Latency of 2.4 μs up to 64-byte messages (Elan to Elan memory)
- Higher MPI latency due to message tag matching

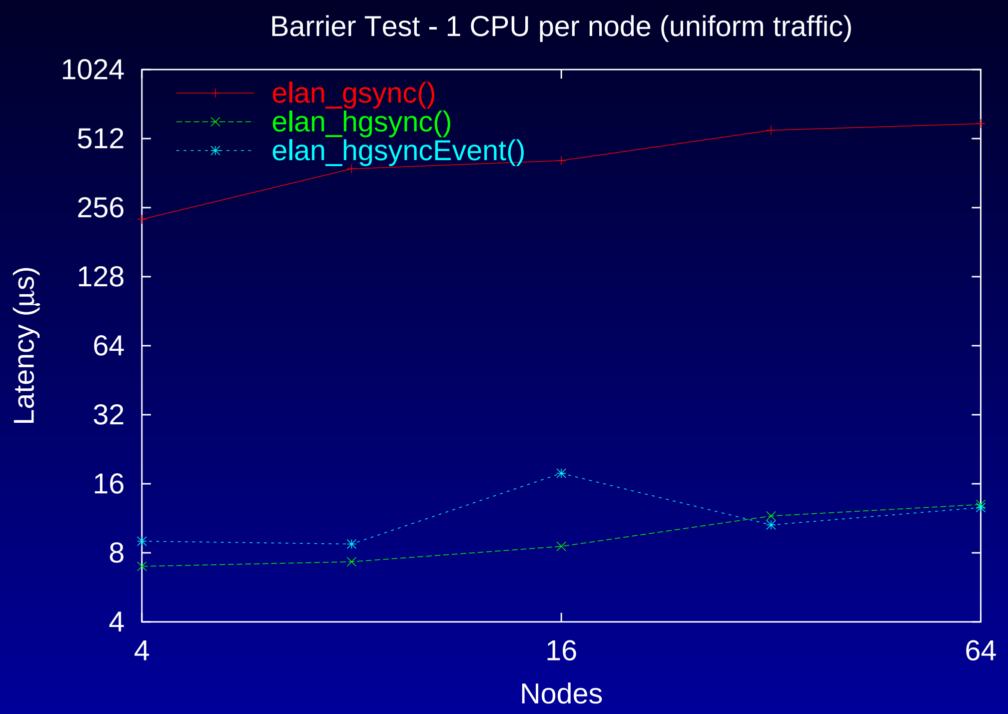
Barrier Synchronization



- Good hardware barrier scalability

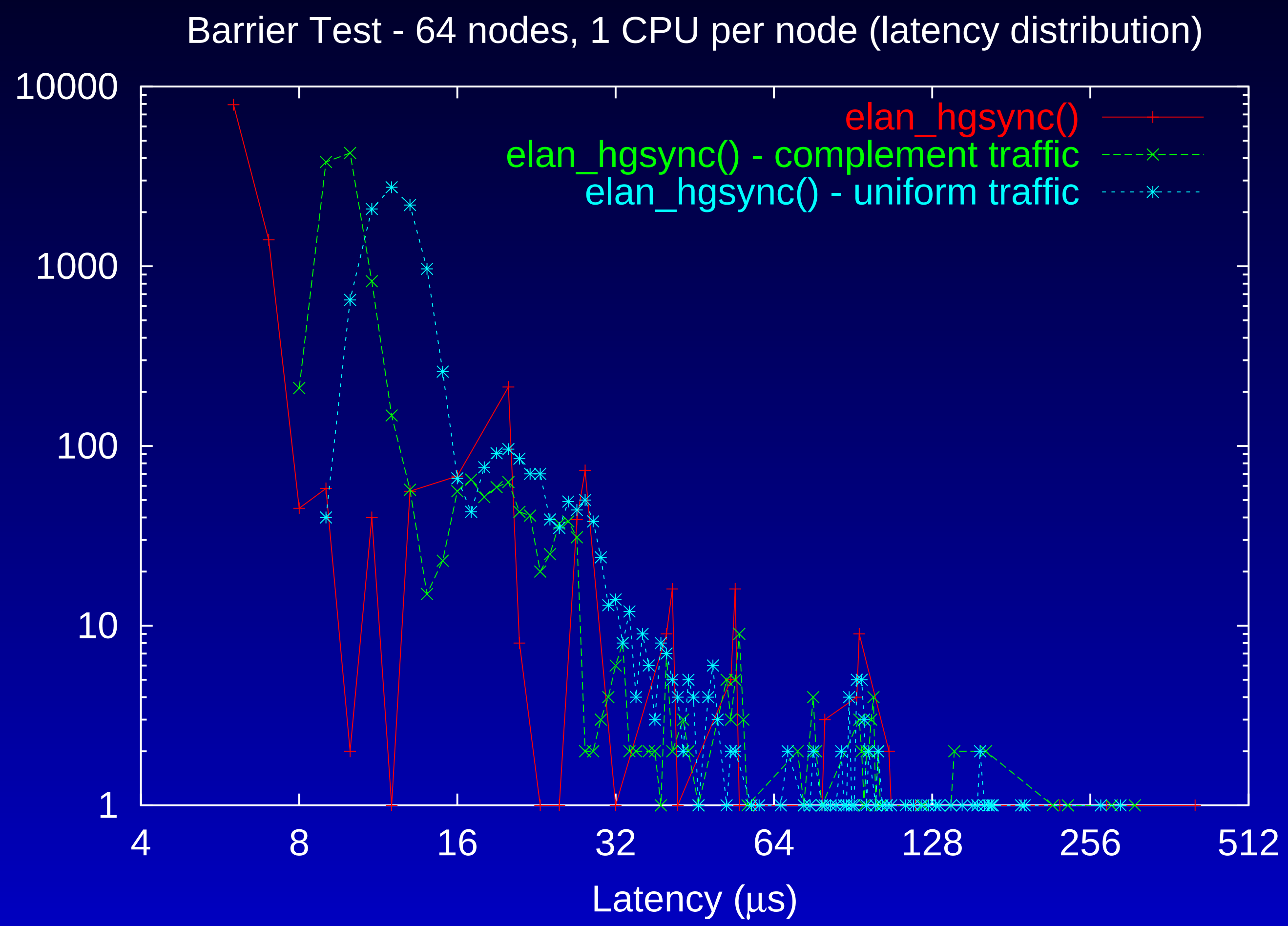
Barrier Synchronization with Background Traffic





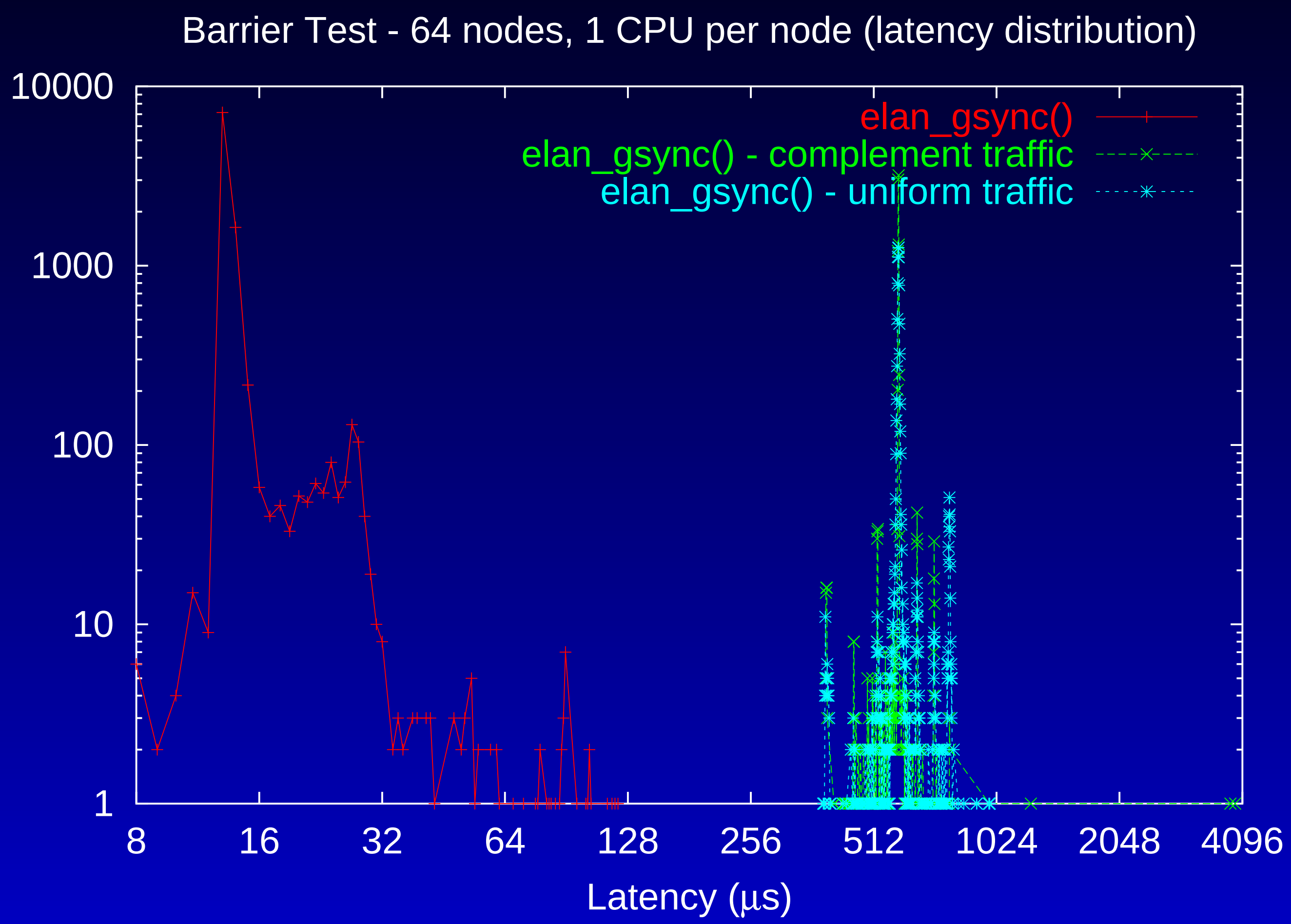
- Software barrier significantly affected (the slowdown is 40 in the worst case)
- Little impact on the hardware barriers, whose average latency is only doubled

Hardware Barrier with Background Traffic



- 94% of the operations take less than 9μs with no bakground traffic
- 93% of the tests take less than 20μs with uniform traffic

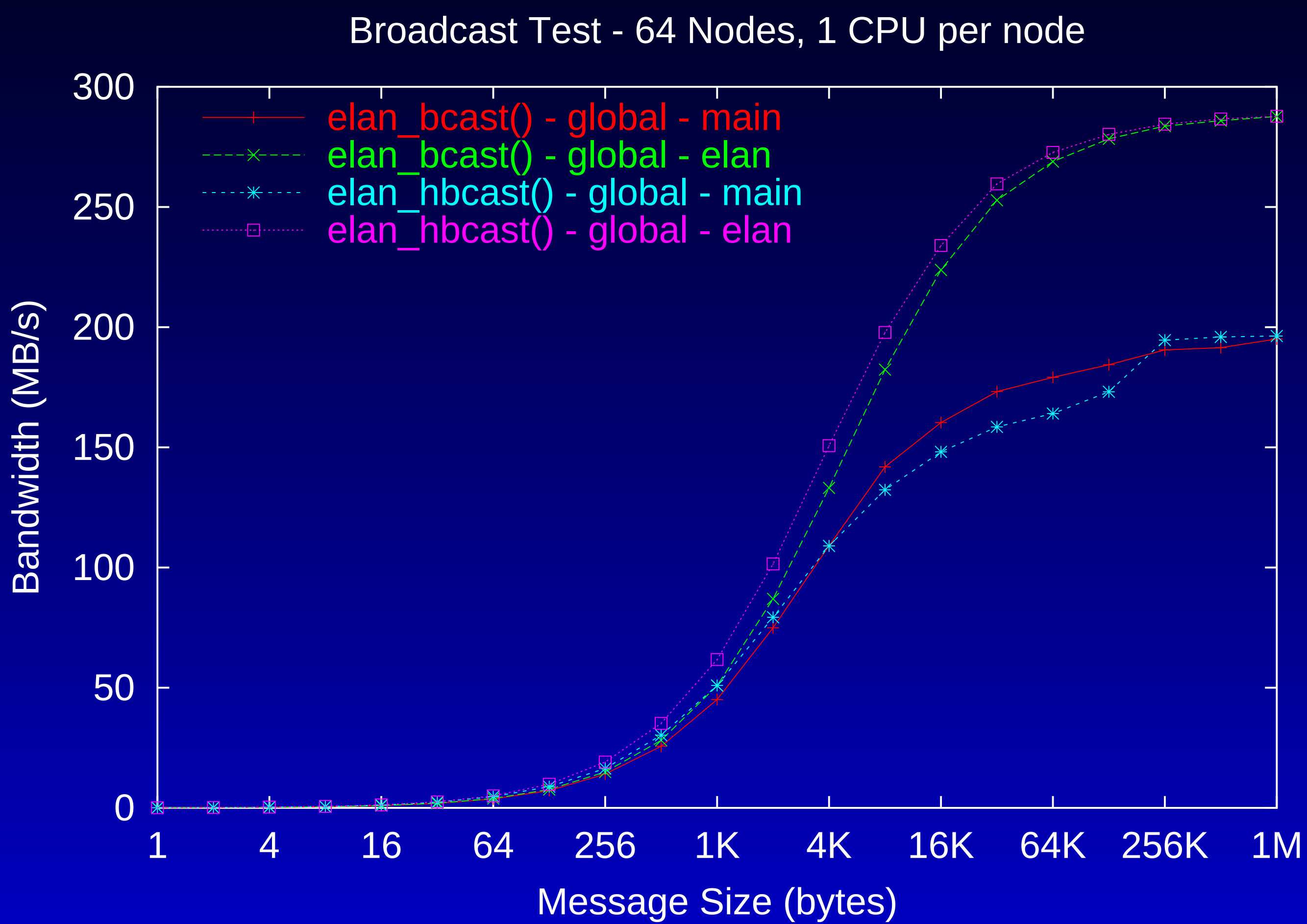
Software Barrier with Background Traffic



- 99% of the barriers take less than 30μs with no bakground traffic
- 93% of the synchronizations complete with less than 605μs with uniform traffic

Los Alamos NATIONAL LABORATORY

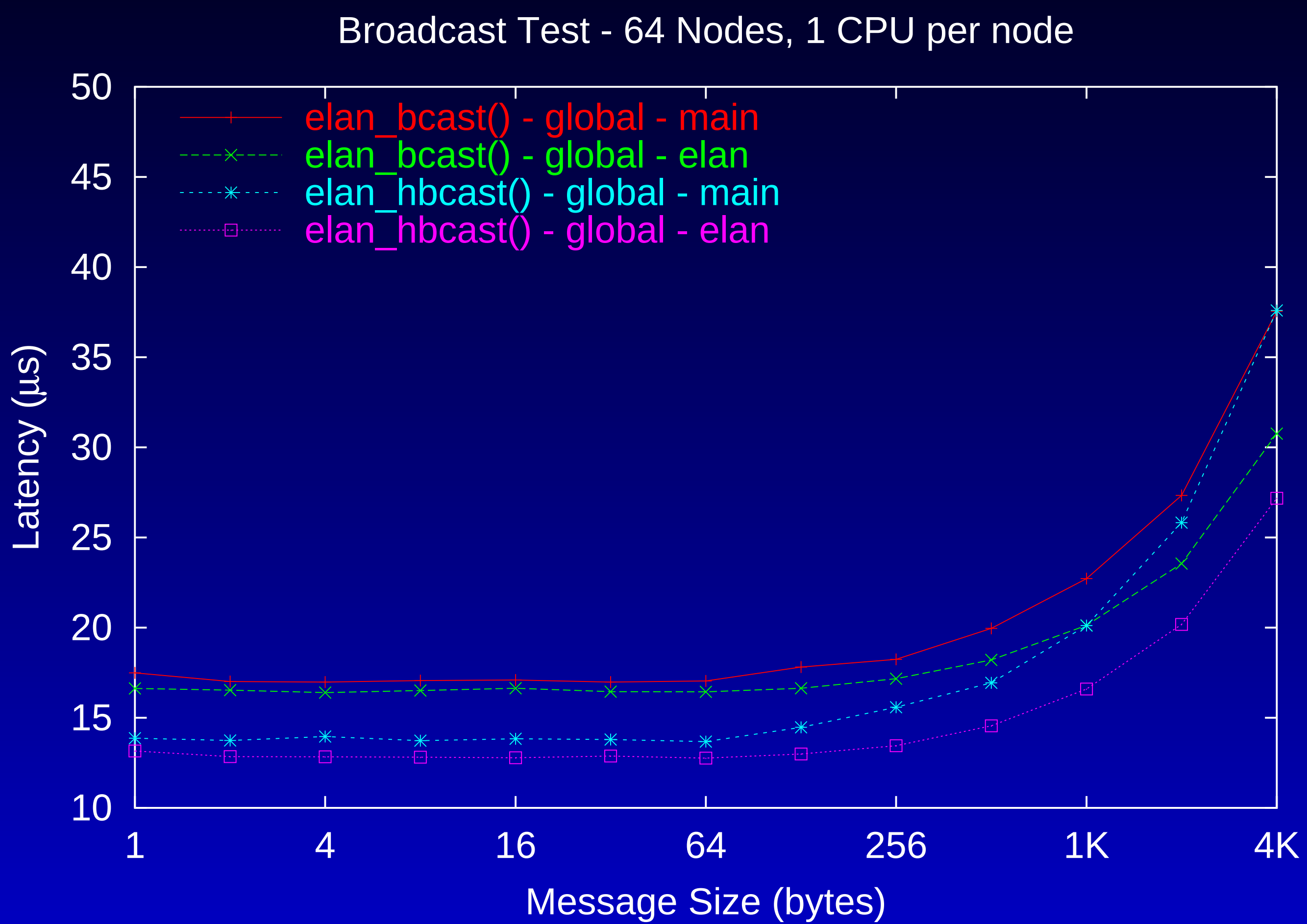
Broadcast Bandwidth



- Asymptotic bandwidth of 288MB/s when using Elan memory for both implementations

Los Alamos NATIONAL LABORATORY

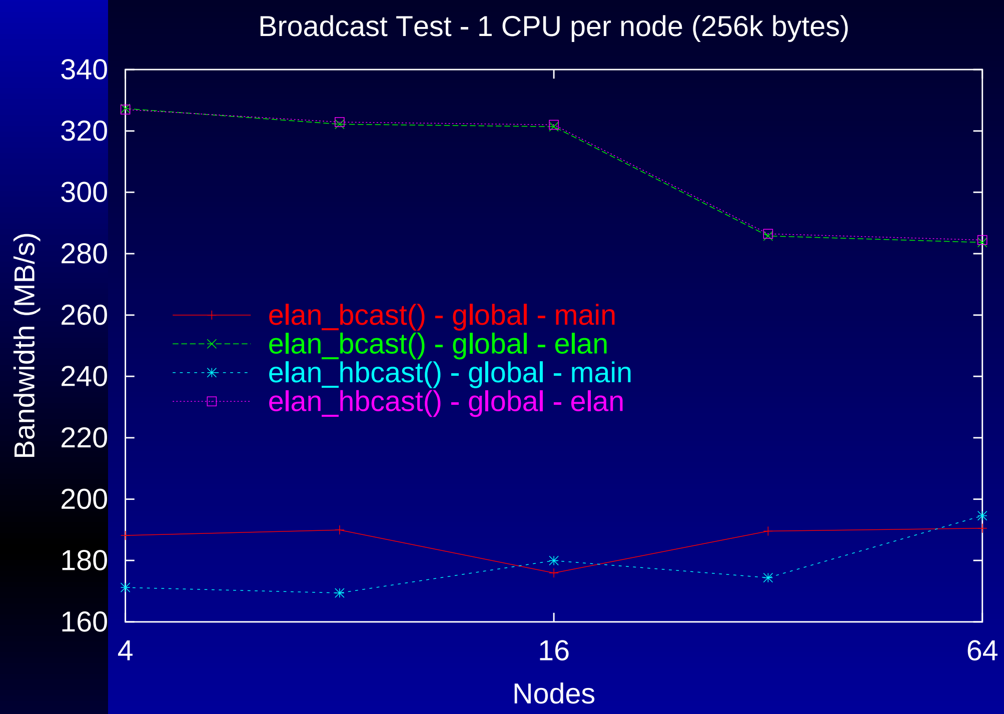
Broadcast Latency

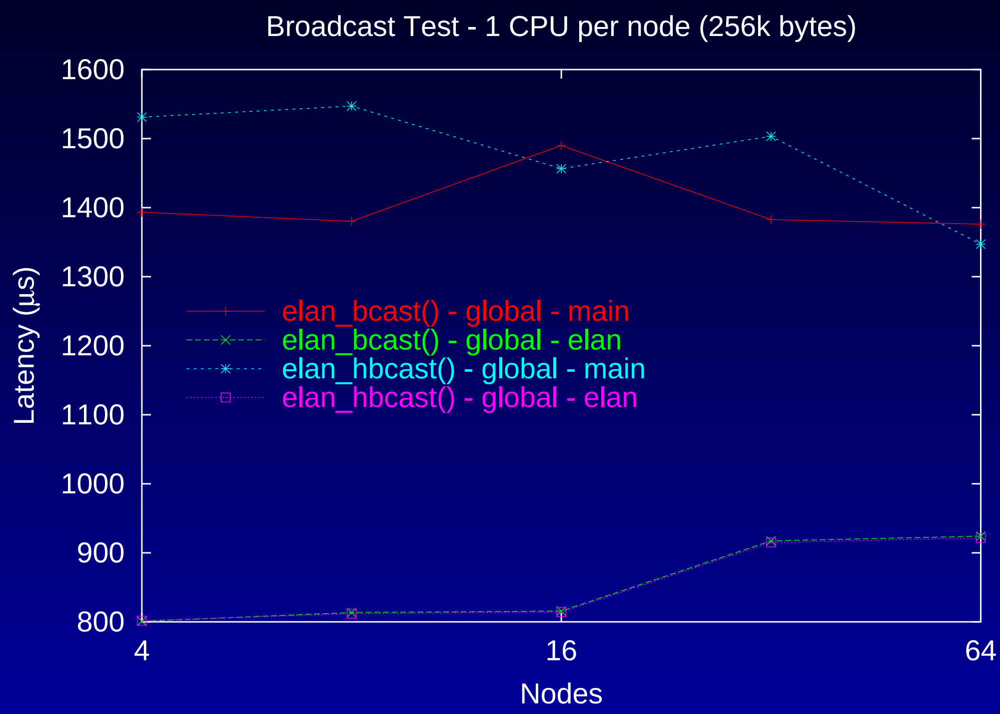


- Hardware latency with Elan buffers below 13μs for messages up to 256 bytes
- Software latencies are 3.5μs higher than hardware latencies

Los Alamos NATIONAL LABORATORY

Broadcast Scalability

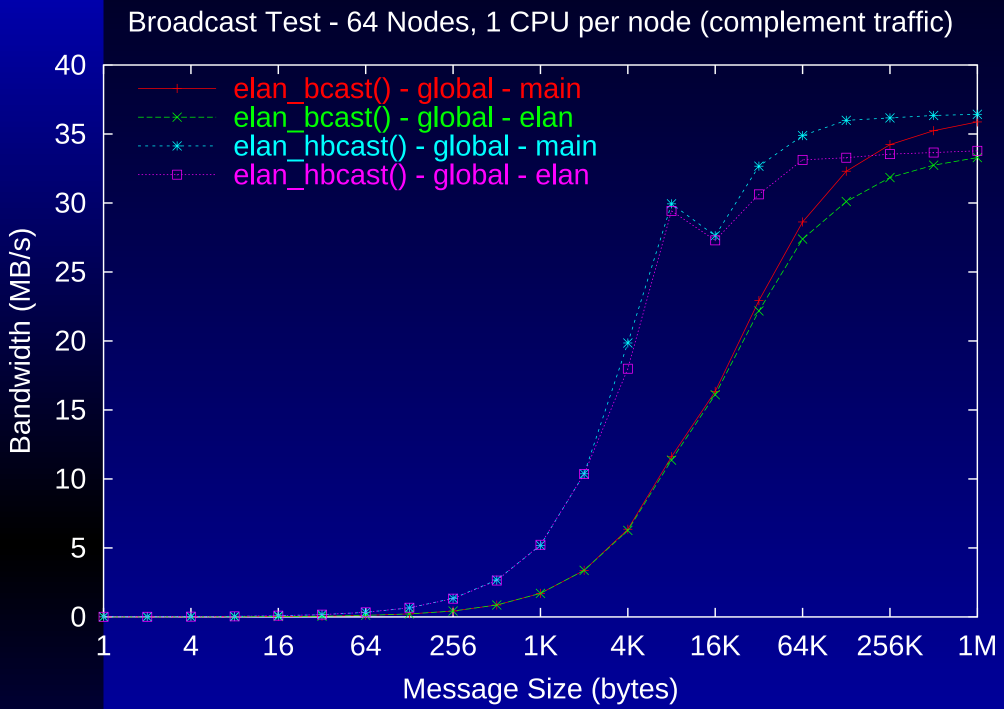


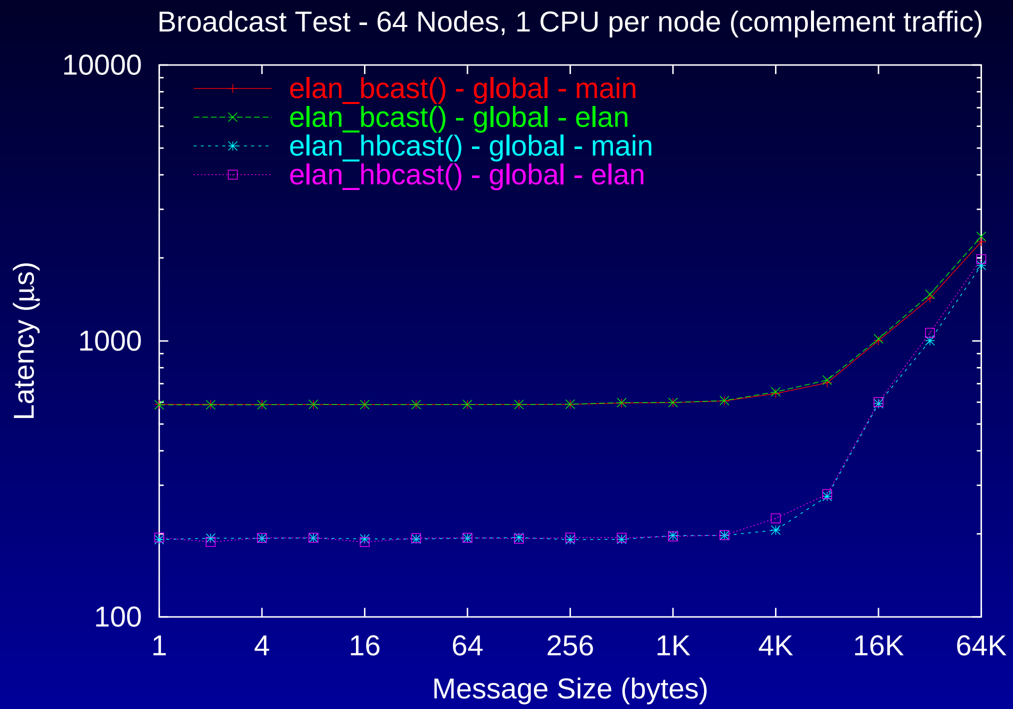


- No significant effect when using buffers in main memory
- With buffers in Elan memory performance depends on the number of switch layers traversed

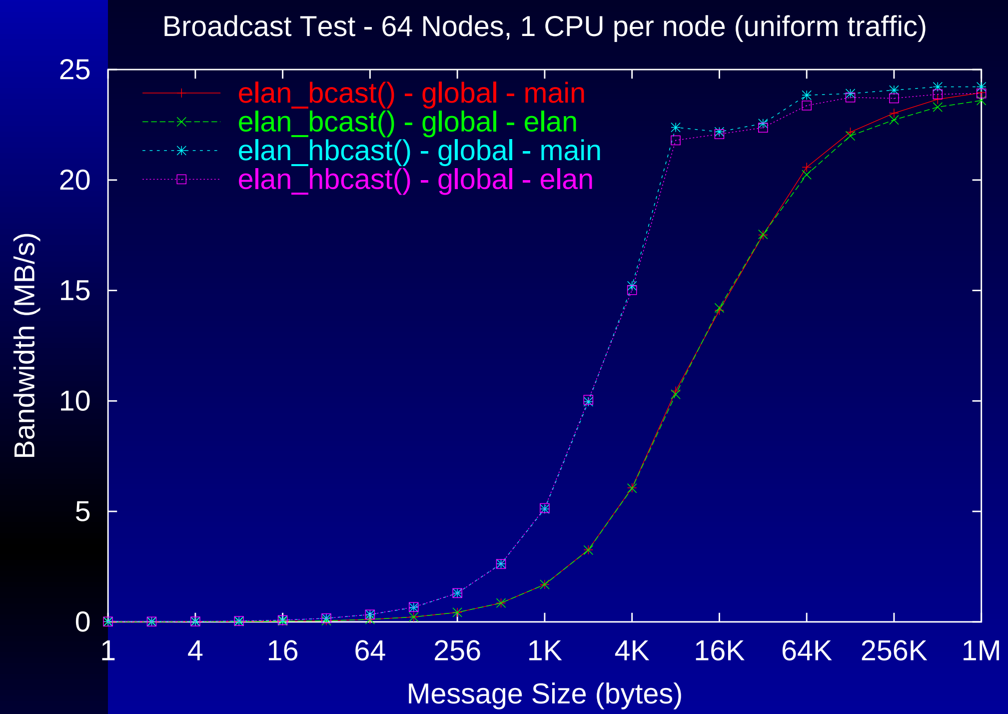
Los Alamos NATIONAL LABORATORY

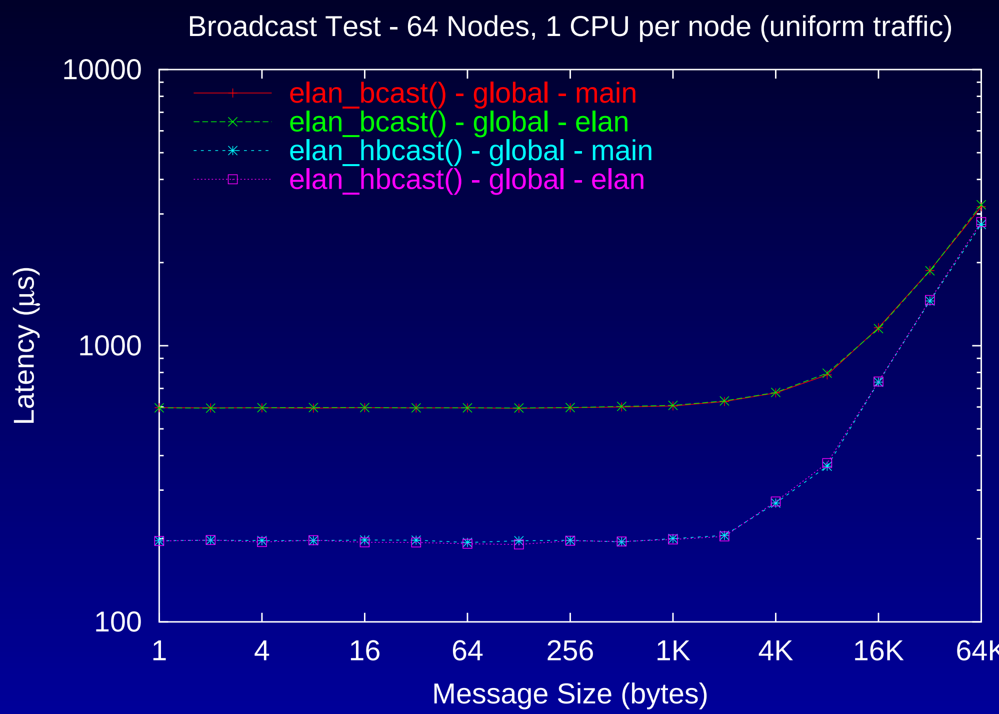
Broadcast with Background Traffic





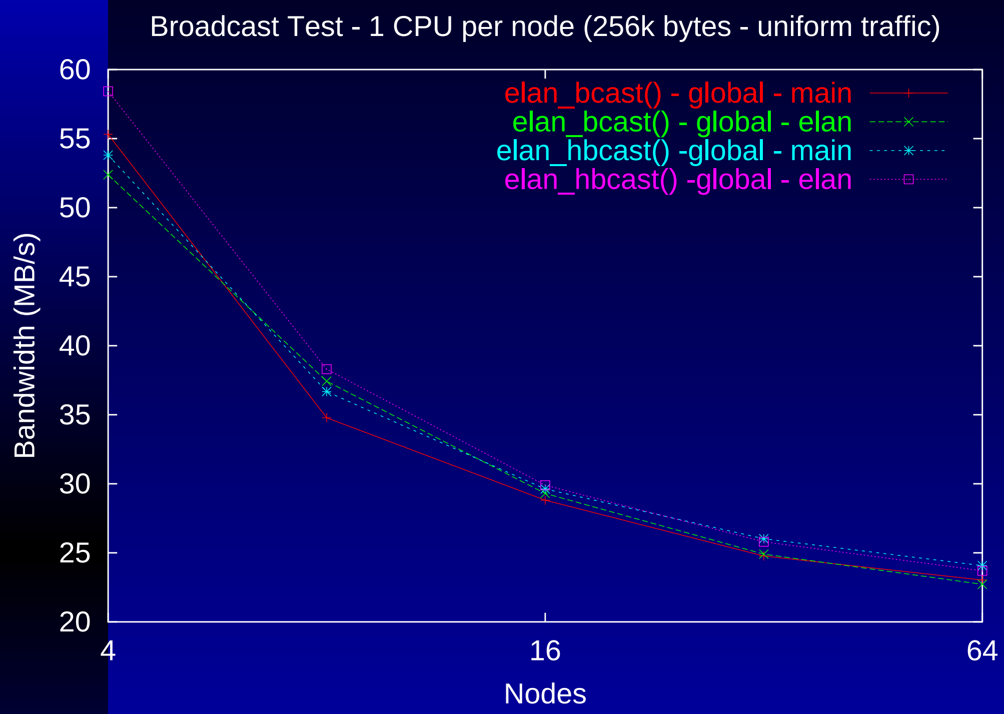
Broadcast with Background Traffic

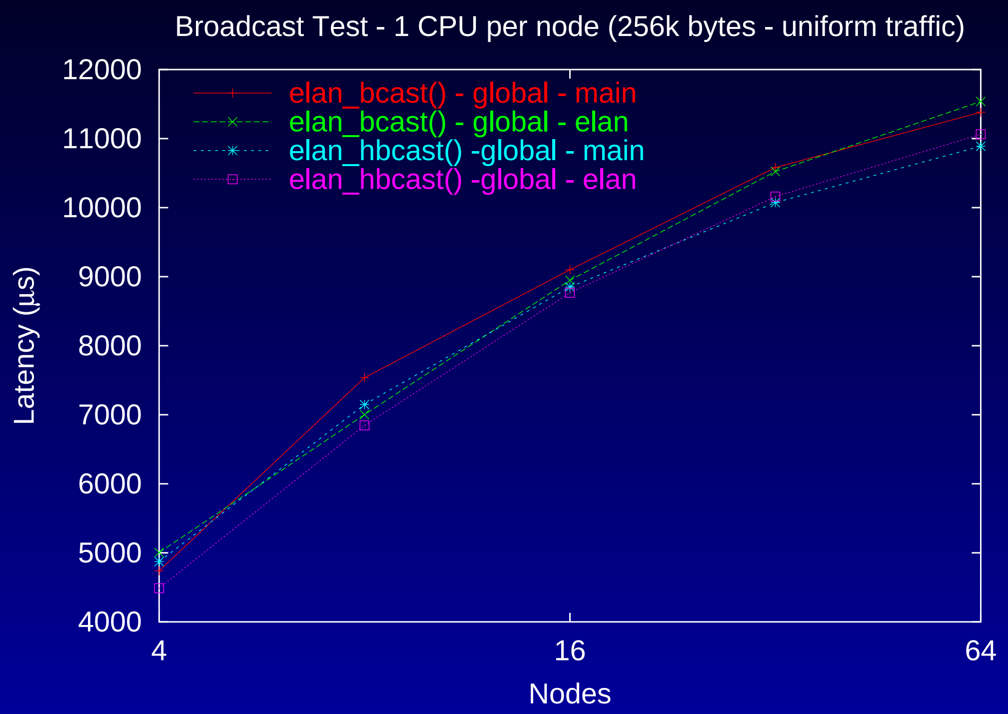




- Latency differences between hw and sw implementations increase
- Better performance with buffers in main memory (due to the background traffic application)

Broadcast with Background Traffic





- Significant performance degradation for all the alternatives

Los Alamos NATIONAL LABORATORY

Conclusions

- Hardware-based synchronization takes as little as 6μs on a 64-node Alphaserver cluster, with very good scalability.

Conclusions

- Hardware-based synchronization takes as little as 6μs on a 64-node Alphaserver cluster, with very good scalability.
- Good latency and scalability are achieved with the software-based synchronization too, which takes about 15μs.

Los Alamos NATIONAL LABORATORY

Conclusions

- Hardware-based synchronization takes as little as 6μs on a 64-node Alphaserver cluster, with very good scalability.
- Good latency and scalability are achieved with the software-based synchronization too, which takes about 15μs.
- The hardware barrier is almost insensitive to background traffic, with 93% of the synchronizations completed in less than 20μs.

Los Alamos NATIONAL LABORATORY

Conclusions

- Hardware-based synchronization takes as little as 6μs on a 64-node Alphaserver cluster, with very good scalability.
- Good latency and scalability are achieved with the software-based synchronization too, which takes about 15μs.
- The hardware barrier is almost insensitive to background traffic, with 93% of the synchronizations completed in less than 20μs.
- With the broadcast, both implementations can deliver a sustained bandwidth of 288 MB/s Elan memory to Elan memory and 200 MB/s main memory to main memory.