Intelligence in high performance network interface cards: Hardware vs. Software

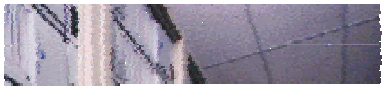
Bernard Tourancheau
Sun Labs Europe
and
Roland Westrelin
ENS−INRIA

Outline of the talk

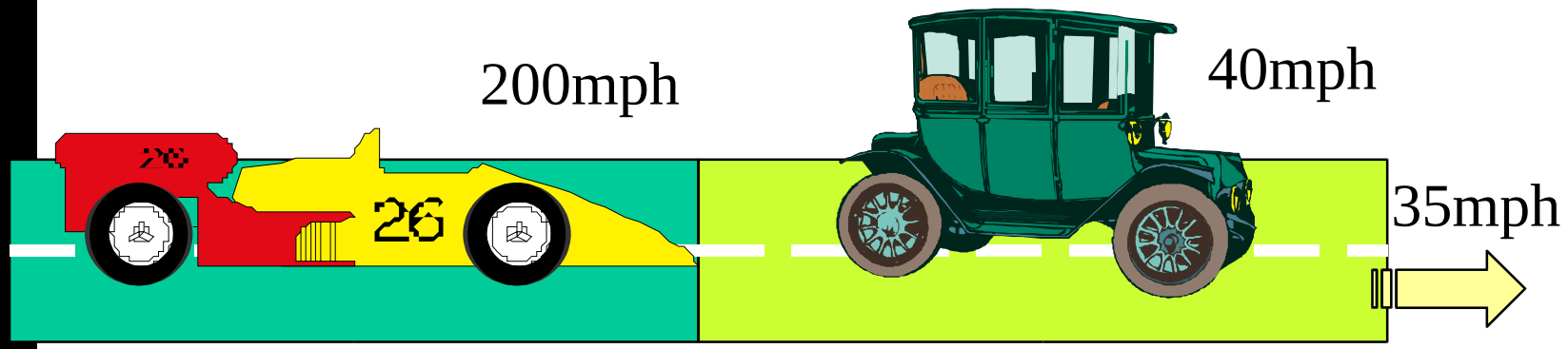
- Context:
  - COTS cluster computing with high speed networks
  - The network interface (NI) is a 2x2 switch
  - Communication systemes architecture
- Software at the network interface (NI) level
- Support for high−level MPI functions at NI level
- Conclusions

COTS clusters network needs

- Throughput computing versus parallel computing
  - Throughput farms only needs low cost networks
  - Supercomputing needs high speed networks
- Communication scalabitility issues
  - Point−to−point throughput
  - Latency
- IO issues
  - Remote storage access

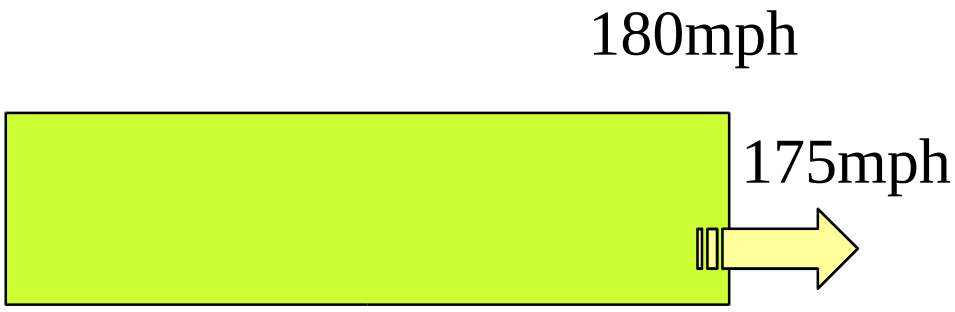


Why optimized network software?



High speed LAN

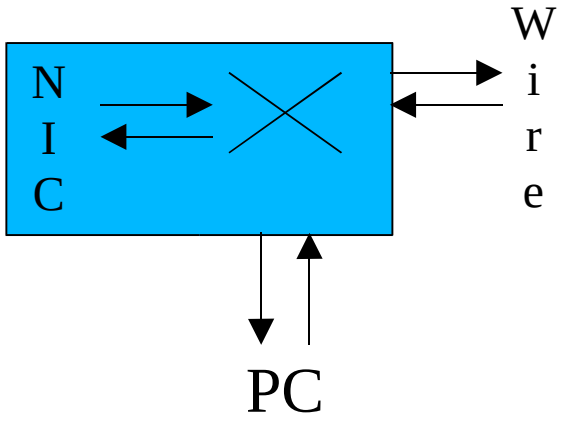
Traditional communication layers



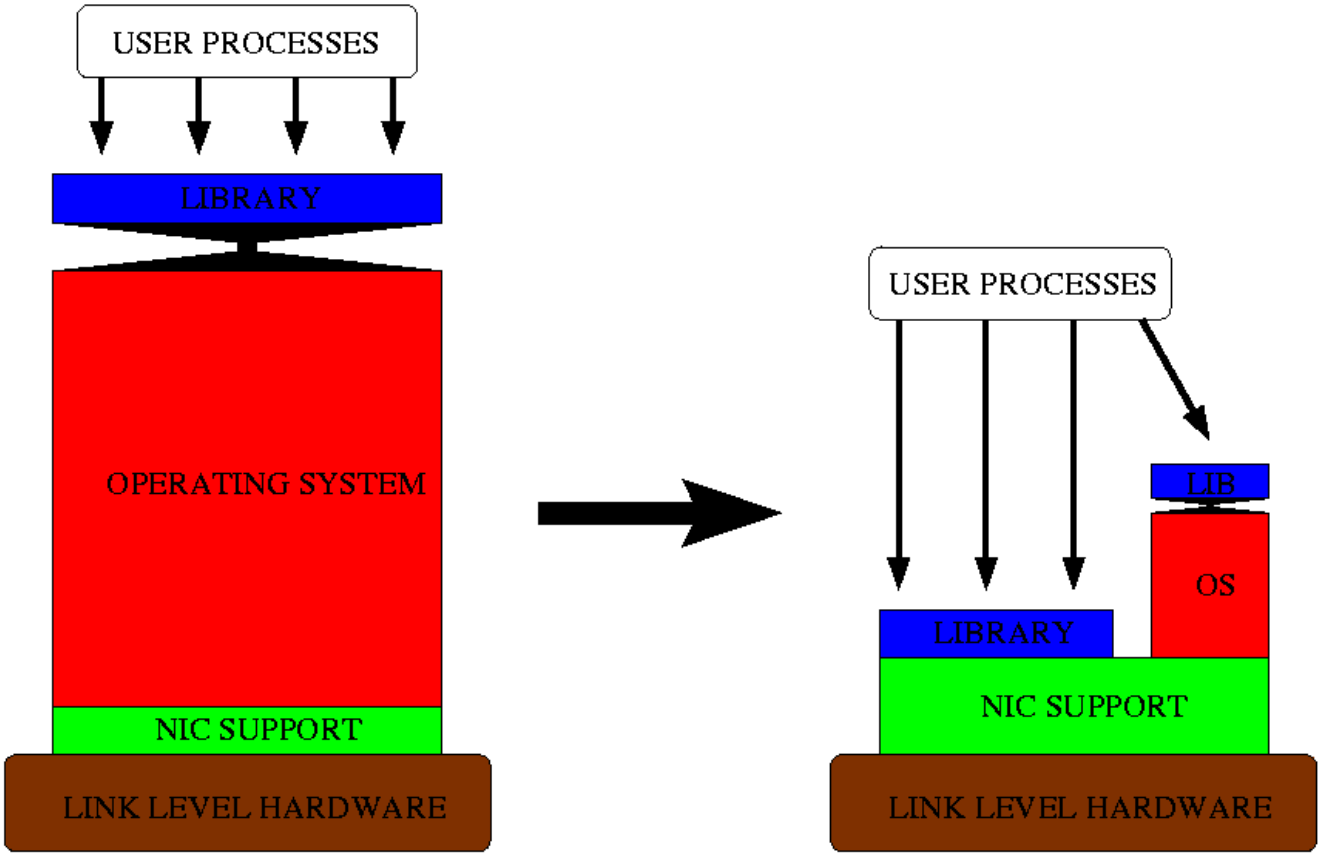
Optimized communication layers

Network interface = switch

- several data paths
  - to/from the wire
  - to/from the computer
  - to/from the NIC
- Several trails
  - wire−computer, network IO
  - wire−wire, forwarding
  - computer−computer, memory copy
  - wire/computer−NIC, off−load treatment



From legacy protocol stacks to user−level interfaces



Issues in the design of light–weight communication layers

- Semantics of communications
  - Message passing, Active messages, ShMem, RPC...
- Control and data transfers between NIC and host
  - PIO/DMA
  - interrupt/polling
- Protection:
  - user level access to the NIC, DMAs
- Reliability:
  - flow control, losses/corruptions, hw/sw failures
- SMP support
  - Through the network or shared memory

High speed LAN NIC products

All in wardware

- VIA aware
  - Giganet
  - Servernet
  - Synfinity
  - ...
- Most GigaEthernet
- SCI

Programmable

- Myricom (Myrinet)/GM
- Quadrics
- Alteon (GigaEthernet)

Programmable NICs

- Advantages:
  - Programmability !
  - Bug fixes/enhancements/prototyping
  - Evolution (IB)
  - Firmware specialisation
- Drawbacks:
  - Performance (?)
  - Price (?)

Outline of the talk

- Context
- Software at the network interface (NI) level
  - BIP communication system for Myrinet
  - BIPng design and performance
- Support for high−level MPI functions at NI level
- Conclusions

Basic Interface for Parallelism: BIP

A library, an OS module and a NIC firmware for the Myrinet programmable network.

- Developed by
  - Patrick Geoffray, Loïc Prylli, Bernard Tourancheau and Roland Westrelin at ENS Lyon and University of Lyon (France)
- Goals:
  - Maximization of the application level performance (close to the hardware maximum)
  - Providing parallel programming interfaces: MPI
- Targets:
  - Myrinet, Linux

BIP principles

- For the latency (small messages):
  - Sending side: PIO copy to a FIFO in the board memory
  - Receiving side: copy to a FIFO in main memory (an extra copy to the application buffer)
  - Avoids handshakes between the host and the NIC
  - 3µs for a message of size 0 (Myrinet 2000)

- For the bandwidth (large messages):
  - Pinning down buffer pages in main memory
  - DMA transfers directly to/from the application buffers, offload the CPU, zero−copy mechanism
  - Pipelining of the 4 steps of the communications
  - 250 MB/s for messages of several MB (Myrinet 2000)

Why rewrite the BIP firmware?

- Study of the initial BIP weaknesses
  - Scalability, extensibility ...
  - Messages splitting not at the firmware level,
  - Poor behavior in case of contentions,
  - Large consecutive communications gap (LogP)
  - The ease of moving more complex protocol or processing at the firmware level
- See if a programmable NIC can compare with hw solutions for small messages
  - Improve performance.

The new architecture BIPng

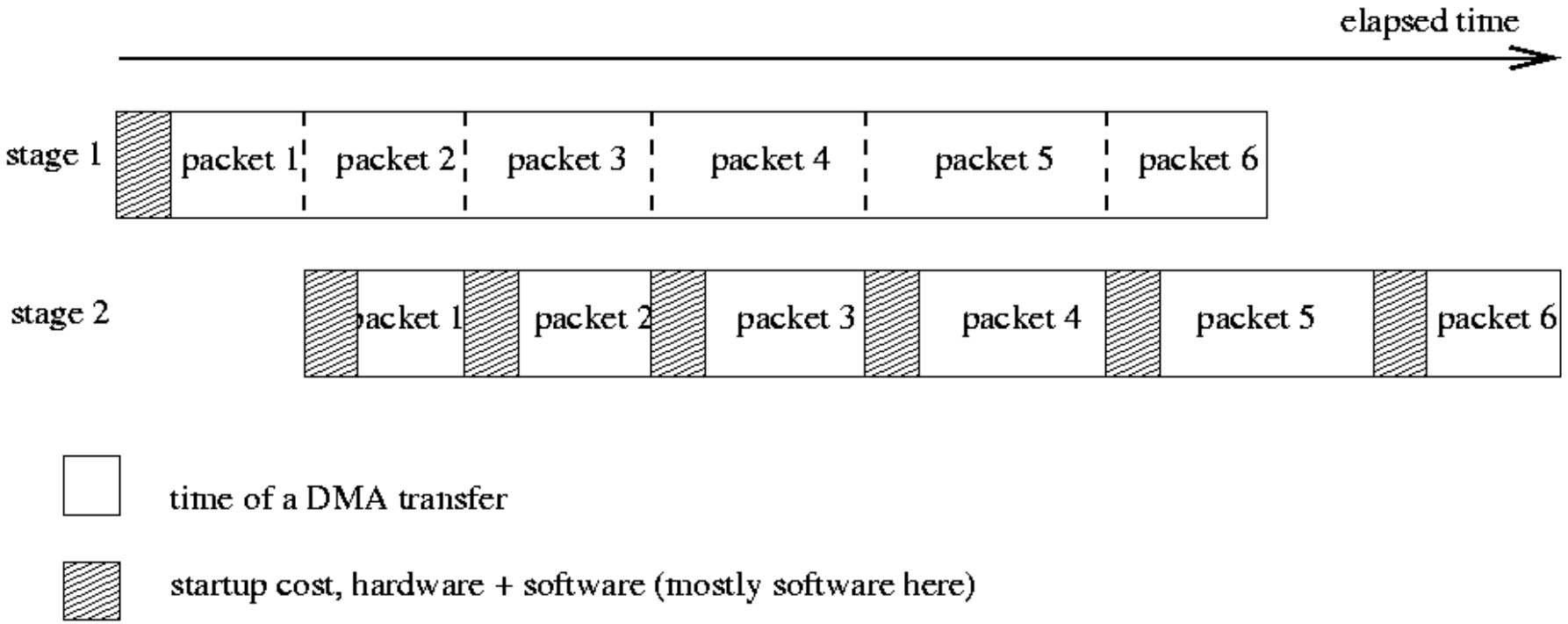
- A low−level layer interfaces with the hardware
  - A set of ’threads’ handles DMA requests
  - New pipelined communications strategies
  - Usage of hardware features whenever possible
  - Asynchronous notification through call−backs
- A high−level layer provides ’BIP’ services
  - A set of threads reacts to host requests and packet arrivals
  - Converts them to lower level layer requests

An event−driven heart (cf GM)

The firmware works as if multiple threads are running :

- A thread is composed of a set of functions
- The (non−preemptive) scheduler is an event−loop :
  - Looks at the software and hardware states and the host requests,
  - Selects a thread,
  - Branches to the corresponding function

Cut through pipeline (cf Trapeze)



- Stage 1 produces data as fast as it can
- Stage 2 consumes the data from stage 1 as they become available

Efficient usage of hardware resources

- A thread does some processing and starts DMAs
- But starting a new DMA is only possible once the previous one is complete
- An optimization allows processing to continue by posting asynchronous DMA requests
- This match the hardware chained PCI DMA feature

Experiments

Testbed: PII 400Mhz, PCI 32bits/33Mhz, Myrinet−1280, LANai7 66Mhz

Message size (bytes)	Half way latency (µs)				
	BIP/ Myrinet	BIPng/ MyriPIO	BIPng/ MyriDMA	VIA/ Giganet	SCI
256	11.89	10.23	14.59	14.75	5.62
512	18.89	13.11	17.46	17.14	8.94
1024	42.13	18.02	21.80	22.7	15.57
2048	53.33	27.59	31.02	32.7	28.8
4096	76.16	45.19	48.96	52.44	55.35

Improvement :

- ping−pong up to 48%
- ping−ping up to 67%
- full−duplex up to 49%

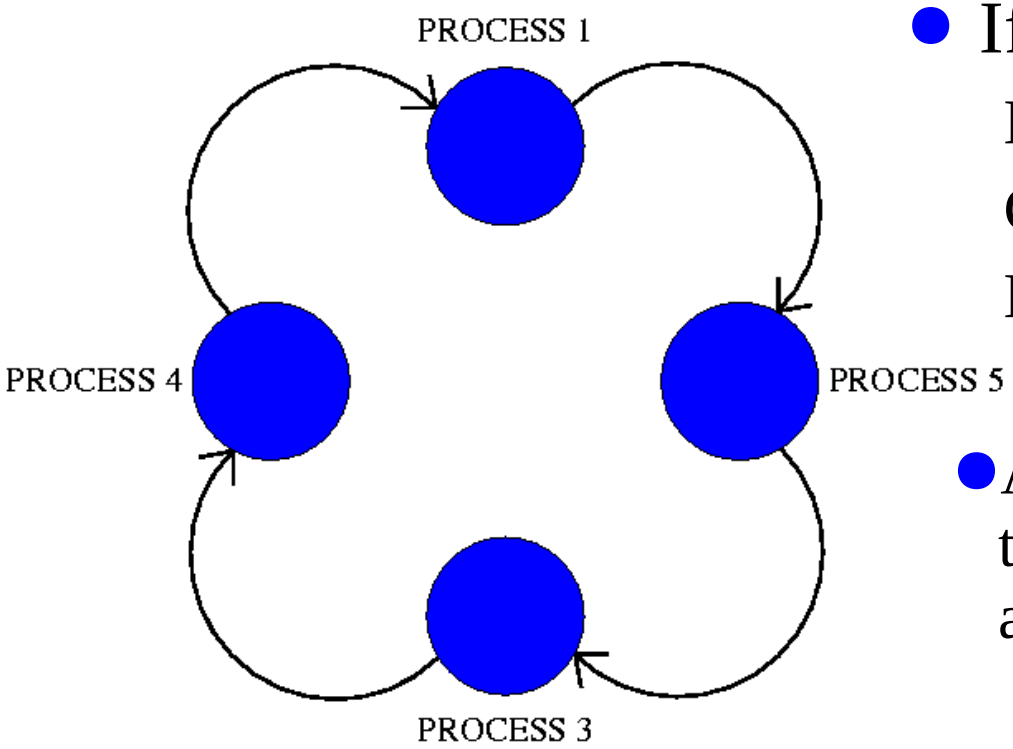
Outline of the talk

- Context
- Software at the network interface (NI) level
- Support for high−level MPI functions at NI level
  - MPI−BIP implementation
  - Communication−computation overlap problem
  - MPI−BIPng prototype solution
- Conclusions

MPI−BIP

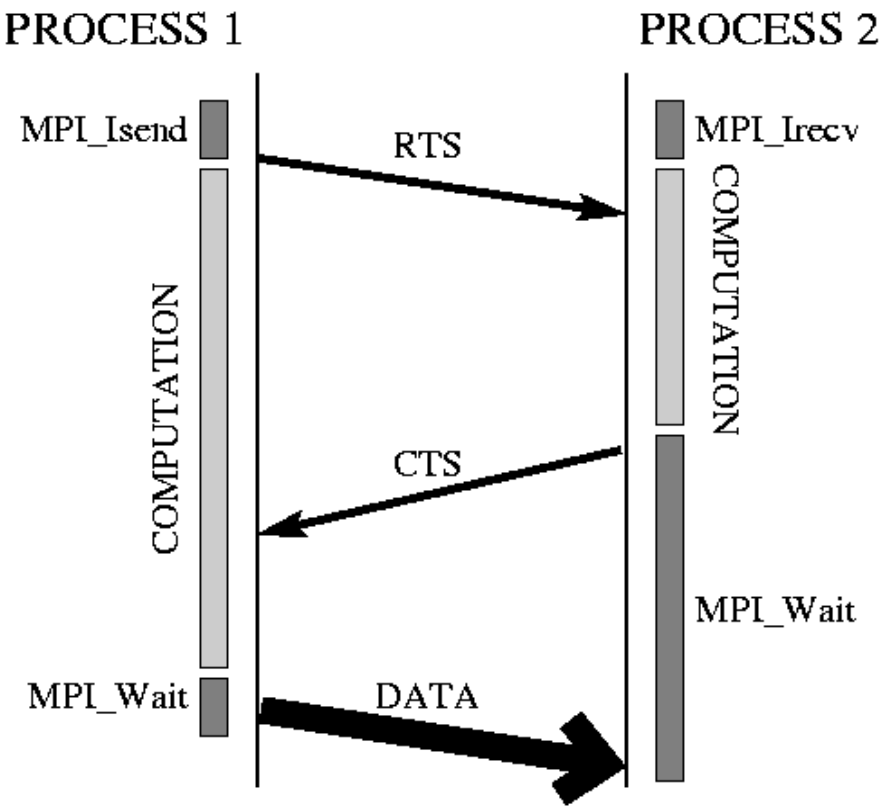
- Port of MPIch on top of BIP
- Relies on MPIch + a layer to adapt BIP to the MPICH needs (add a few µs of overheads)
- Uses 3 protocols:
  - 'Short': one copy on the receive side and credit flow control
  - Three−way 'eager': request + acknowledgement. (Possible buffer allocation and extra copy on the receive side)
  - 'Rendez−vous': request+acknowledgement. The message is split in fragments
- Defines a new portability layer in MPICH used in several MPI implementations

Overlapping, a problem with MPI implementations



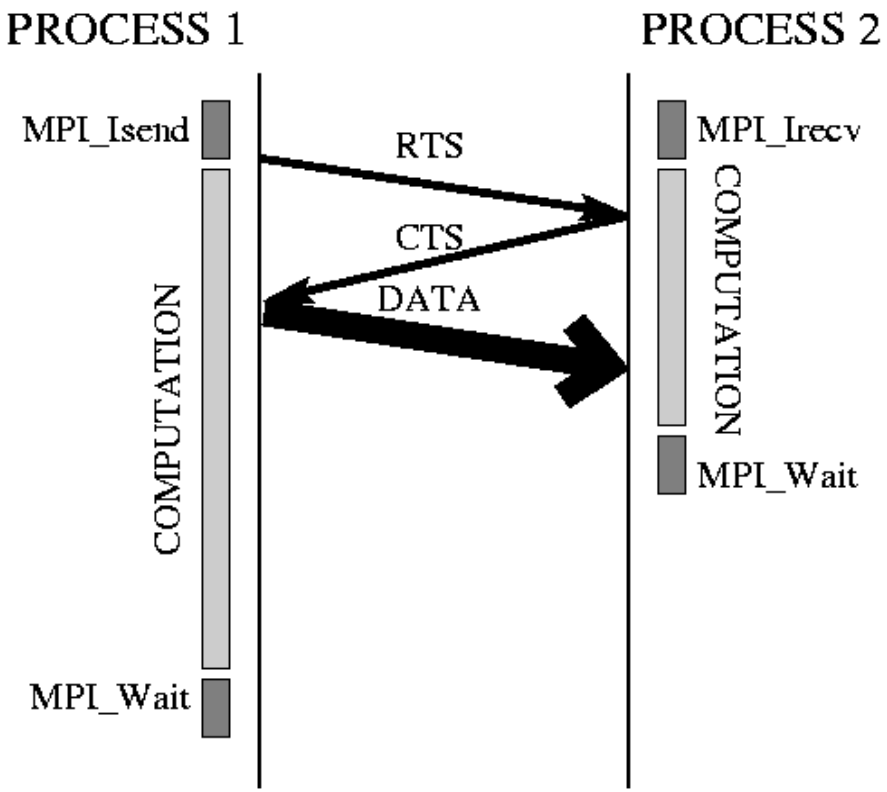
- If each process does:
  - MPI_Isend
  - Computation for a long time
  - MPI_Wait
- Almost embarrassingly parallel but the speedup will be 1 with almost all the MPI implementations !

Why is overlapping impossible ?



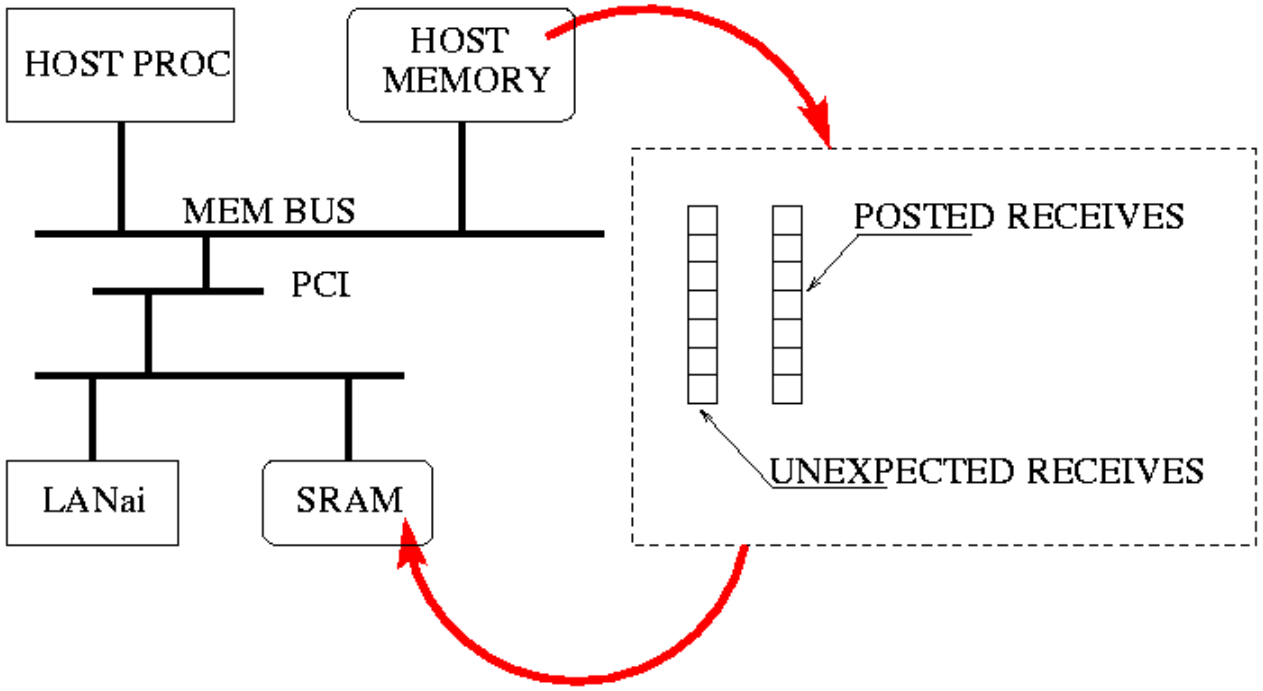
- In zero−copy communications the receive buffer must be ready before data are sent thus RTS.
- User−level systems use polling to achieve performance.
- The thread that computes also handles the protocol.
- The matching between send and receive is done when the RTS is received.
- This RTS/CTS protocol within the compute thread prevents communication−computation overlap.

Solution 1



- Operating system help + interrupts.
- Software overhead is high.
- Improvement: intelligence at the NIC level for selective interrupts i.e. The NIC forces the reception of messages that are waiting for a long time with an interrupt

Our choice : Solution 2



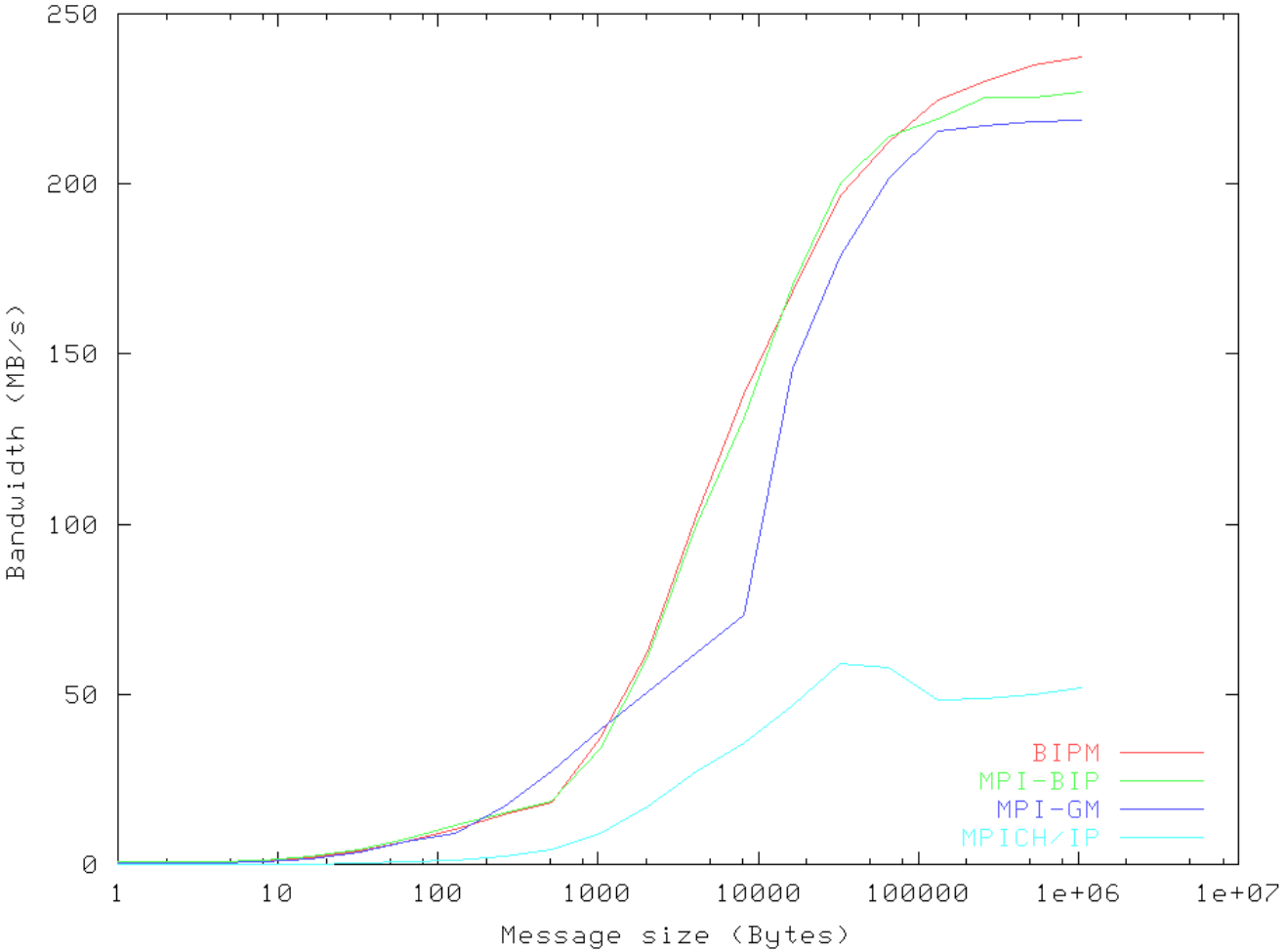
- Move the processing of the protocol at the firmware level
- Matching between send and receives request is done at the NIC level

Related subjects

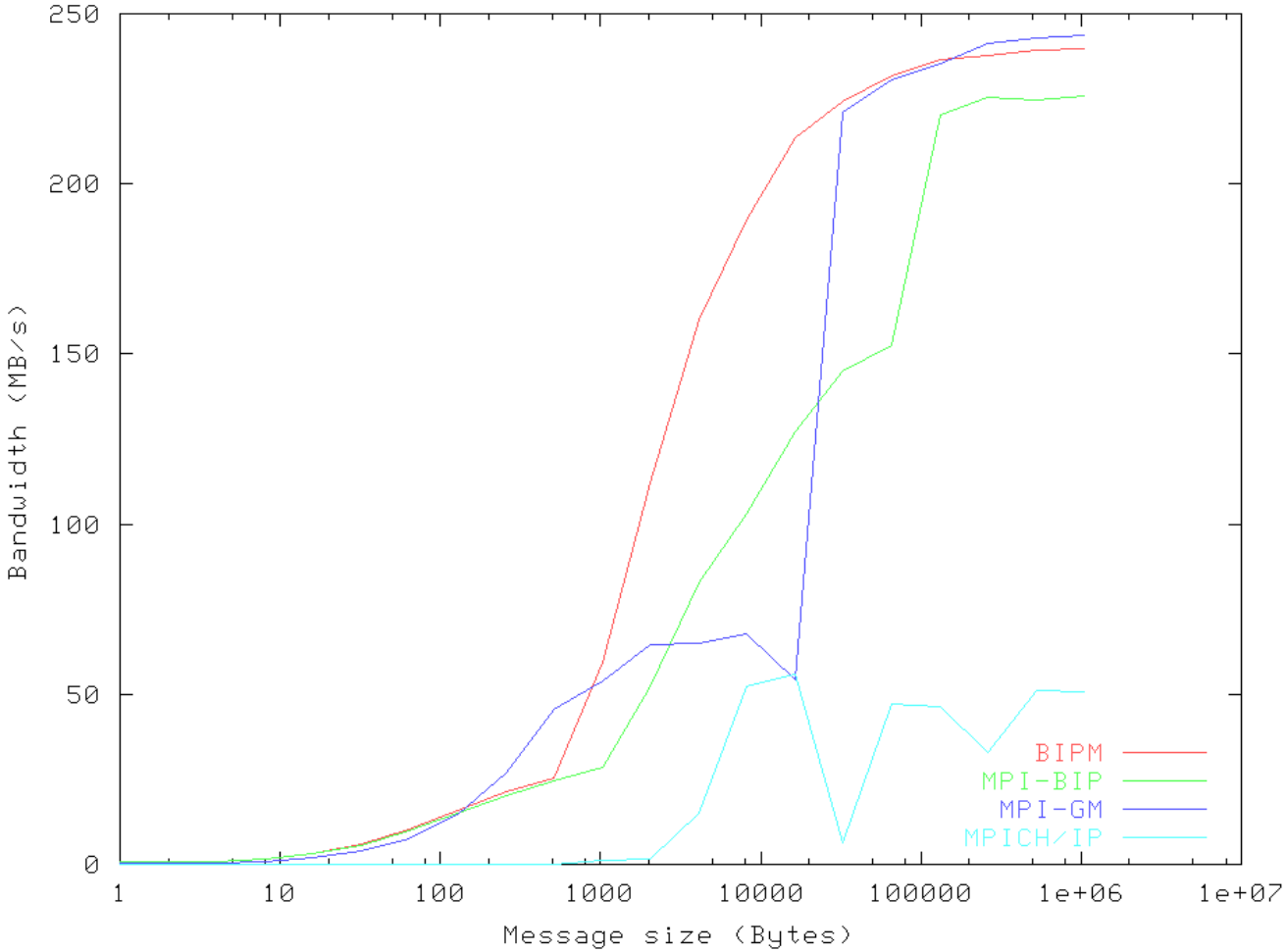
- Low latencies for the small messages
  - 1 message protocol + memory copy
  - Thus the matching for these messages must be performed by the host
- SMP support
  - If it uses shared memory then the matching has to be done on the host
  - If not, done at the NIC level

Performance: ping−pong

Testbed: PIII 600Mhz, PCI 64bits/66Mhz, Myrinet−2000, LANai9 200Mhz



Performance: ping−ping



.Nai9

Application performance: NAS parallel benchmarks

Testbed: 4 dual PIII 600Mhz, PCI 64bits/66Mhz, Myrinet−2000, LANai9 200Mhz

	Seq.	4 processors				8 processors			
		MPICH/IP	MPI−BIP	MPI−GM	MPI−BIPng	MPICH/IP	MPI−BIP	MPI−GM	MPI−BIPng
ne	12s	5.6s	2.8s	3.2s	2.8s	6.1s		2.3s	2.1s
ed		2.1	4.3	3.7	4.3	2		5.2	5.7
ne	1655s	442s	433s	793s	432s	265s	234s	239s	232s
ed		3.7	3.8	2.1	3.8	6.2	7.1	6.9	7.1

Conclusions

- The new design provides performances close or better than the hardware NICs ones
- This shows that careful software pipeline design is competitive with ’brute force’
- The new design provides an abstract API layer to write extensions
- Part of the MPI protocol was moved at the NIC level which demonstrates an extension.
- Such results are promising for further work like IP, direct IO, multicast, ...

Web

- http://resam.univ−lyon1.fr
- bernard.tourancheau@sun.com
- roland.westrelin@ens−lyon.fr