End nodes challenges with multigigabit networking

GARR – WS4

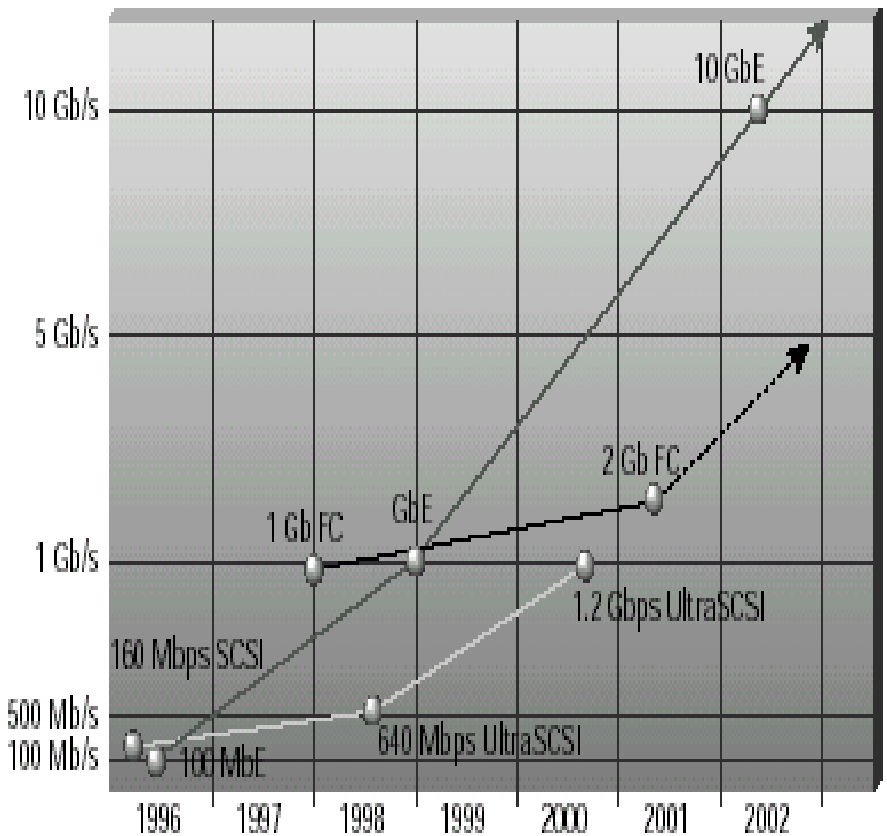
Roberto Innocente

inno@sissa.it

Gilder's law

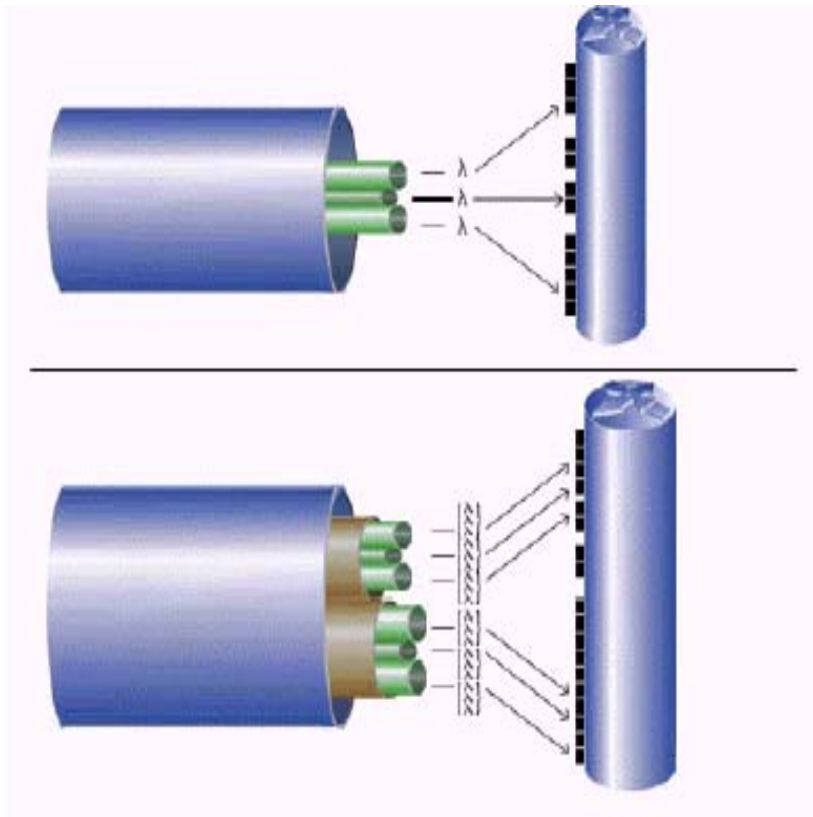
- proposed by G.Gilder(1997) :
  - **The total bandwidth of communication systems triples every 12 months**
- compare with Moore's law (1974):
  - **The processing power of a microchip doubles every 18 months**
- compare with:
  - memory performance increases 10% per year

(source Cisco)



Optical networks

- Large fiberbundles multiply the fibers per route : 144, 432 and now 864 fibers per bundle
- DWDM (Dense Wavelength Division Multiplexing) multiplies the b/w per fiber :
  - Nortel announced a 80 x 80G per fiber (6.4Tb/s)
  - Alcatel announced a 128 x 40G per fiber(5.1Tb/s)



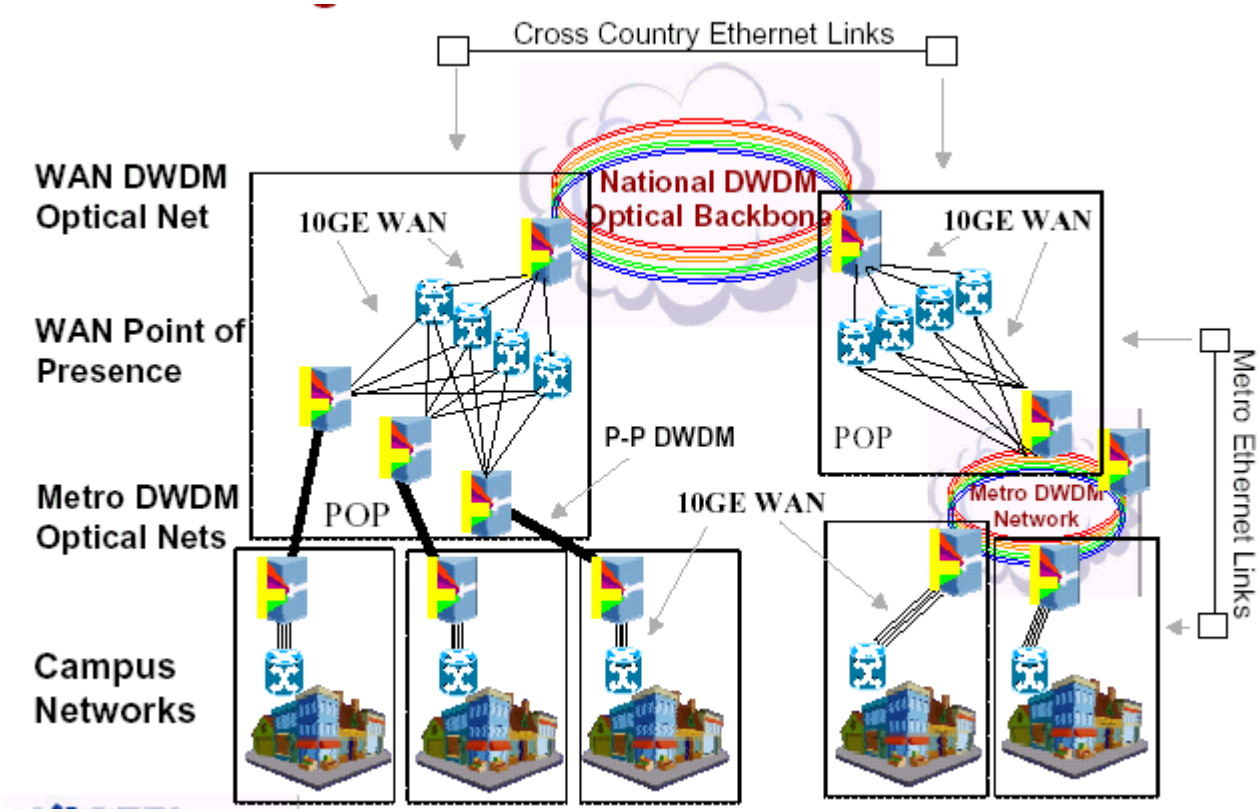
10 GbE /1

- IEEE 802.3ae 10GbE ratified on June 17, 2002
- Optical Media ONLY : MMF 50u/400Mhz 66m, new 50u/2000Mhz 300m, 62.5u/160Mhz 26m, WWDM 4x 62.5u/160Mhz 300m, SMF 10Km/40Km
- Full Duplex only (for 1 GbE a CSMA/CD mode of operation was still specified despite no one implemented it)
- LAN/WAN different :10.3 Gbaud – 9.95Gbaud (SONET)
- 802.3 frame format (including min/max size)

10GbE /2

from Bottor (Nortel Networks) :

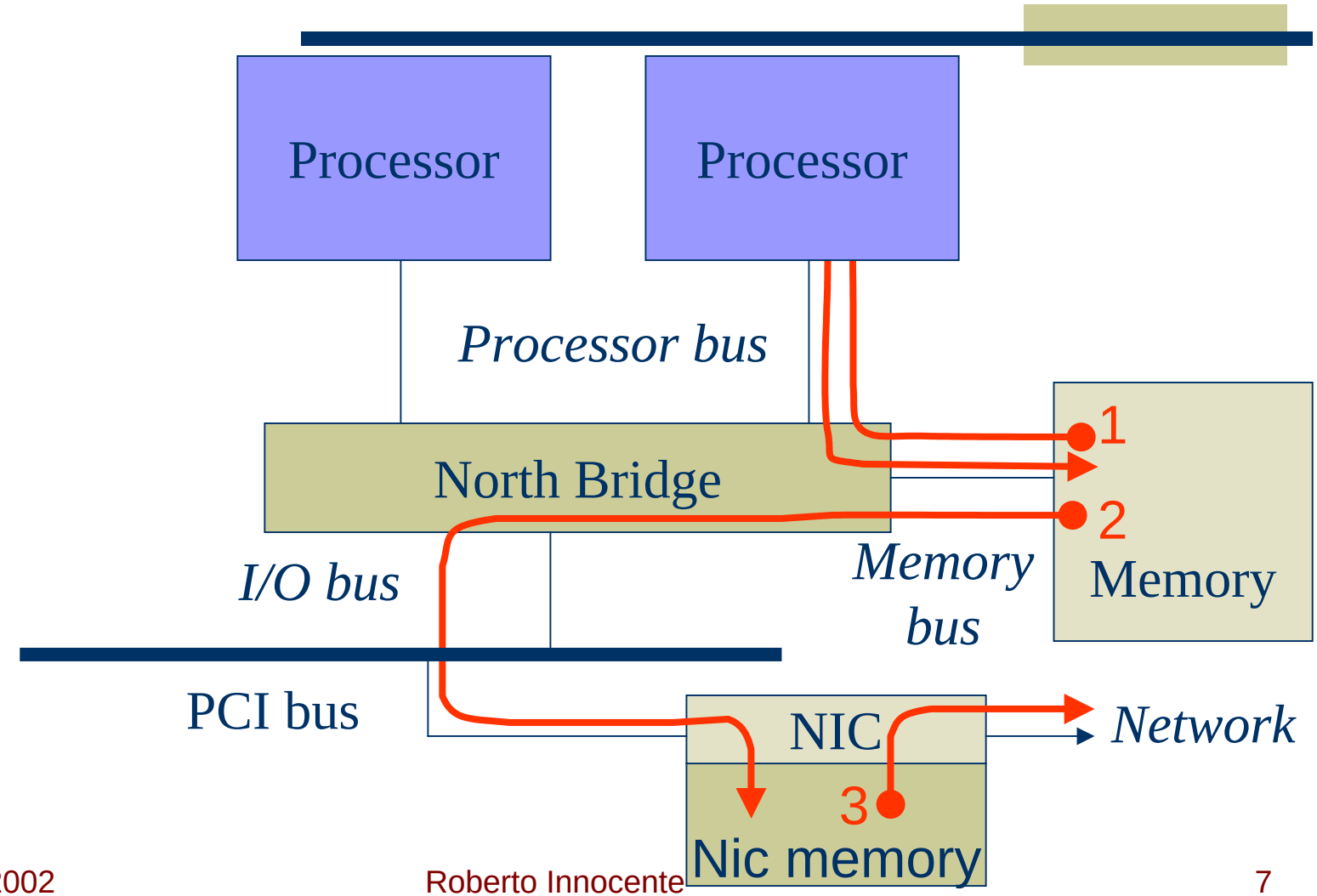
marriage of Ethernet and DWDM



Challenges

- Hardware
- Software
- Network protocols

Standard data flow /1



Standard data flow /2

- Copies :
  1. Copy from user space to kernel buffer
  2. DMA copy from kernel buffer to NIC memory
  3. DMA copy from NIC memory to network
- Loads:
  - 2x on the processor bus
  - 3x on the memory bus
  - 2x on the NIC memory

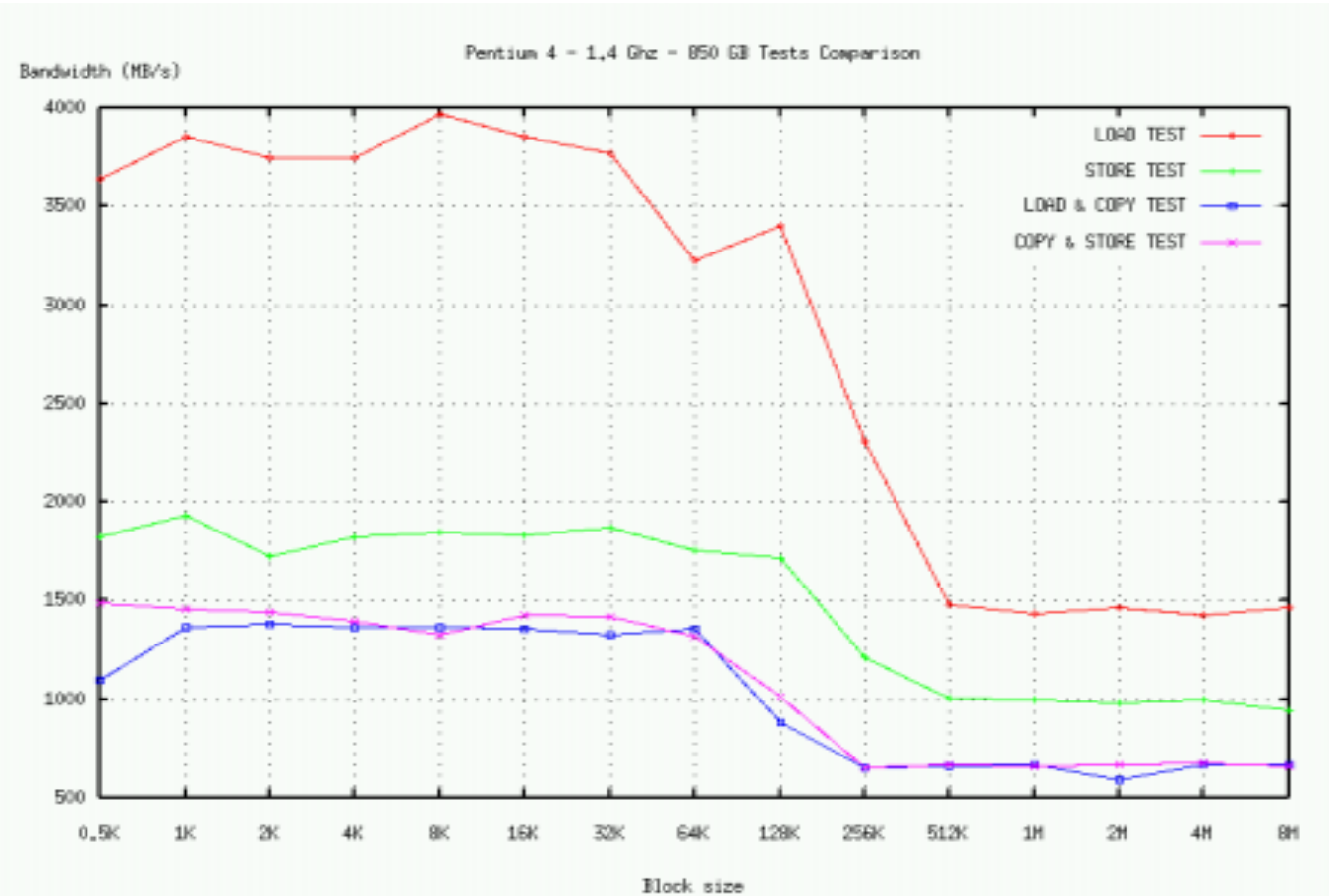
Hardware challenges

- Processor bus
- Memory bus
- I/O bus
- Network technologies

Processor / Memory bus

- Processor bus :
  - current technology 1 - 2 GB/s
  - Athlon/Alpha are pt-2-pt / IA32 multi-pt
  - e.g IA32 P4 : 4 x 100 Mhz x 8 bytes = 3.2 GB/s peak, but with many cycles of overhead for each line of cache transferred
- Memory bus :
  - current technology 1-2 GB/s
  - DDR 333 8 bytes wide = 2.6 GB/s th. peak
  - RAMBUS 800 Mhz x 2 bytes x 2 channels = 3.2 GB/s th. peak
  - setup time should be taken into account

Memory b/w


Pentium 4 - 1.4 Ghz - 850 GB Tests Comparison
Bandwidth (MB/s)
4000
3500
3000
2500
2000
1500
1000
500
0.5K
1K
2K
4K
8K
16K
32K
64K
128K
256K
512K
1M
2M
4M
8M
Block size
LOAD TEST
STORE TEST
LOAD & COPY TEST
COPY & STORE TEST

PCI Bus

- PCI32/33 4 bytes@33Mhz=132MBytes/s (on i440BX,...)
- PCI64/33 8 bytes@33Mhz=264Mbytes/s
- PCI64/66 8 bytes@66Mhz=528Mbytes/s (on i840)
- PCI-X 8 bytes@133Mhz=1056Mbytes/s

PCI-X implements split transactions

PCI performance with common chipsets

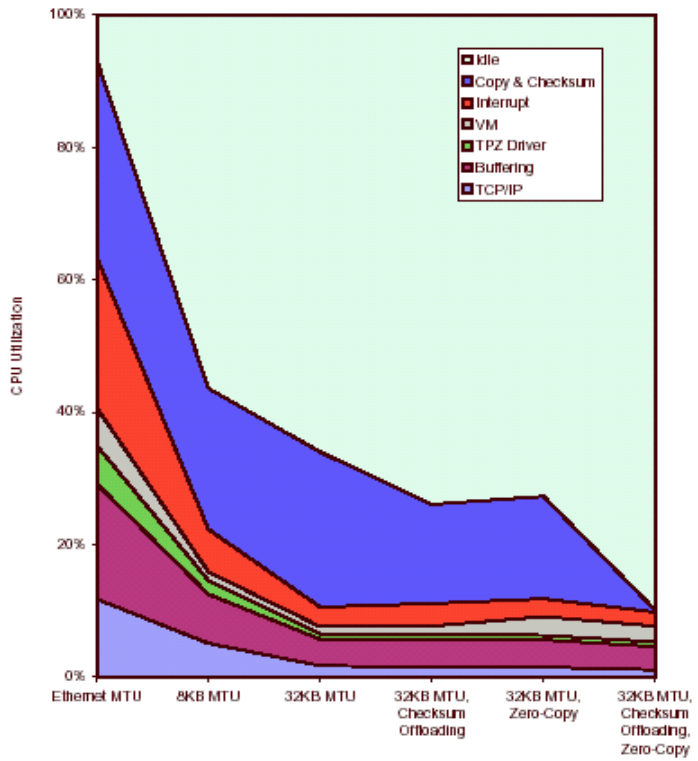
Chipset	Read MB/s	Write MB/s
Supermicro 370DLE, dual PIII, Serverworks Serverset III LE	432	486
Supermicro P4D6, P4 Xeon, Intel e7500 chipset	418	478
Alpha ES45, Titan chipset	387	464
Intel 460GX (Itanium)	353	379
Supermicro 370DEI, PIII, Serverset III HE	299	353
AMD dual, AMD760MPX chipset	299	311
P4 Xeon, Intel 860 chipset	215	299
API UP2000 alpha, Tsunami chipset	183	228

I/O busses

- PCI/PCI-X available, DDR/QDR under development
- Infiniband (HP, Compaq, IBM, Intel): available
  - PCI replacement, memory connected !
  - pt-to-pt , switch based
  - 2.5 Gb/s per wire f/d, widths 1x,4x,12x
- 3rd Generation I/O (HP,Compaq,IBM..)
  - PCI replacement
  - pt-to-pt switched
  - 2.5 Gb/s per wire f/d ,widths 1x,2x,4x,8x,12x..32x
- HyperTransport (AMD,...)
  - PCI and SMP interconnect replacement
  - pt-to-pt switched, synchronous
  - 800 Mb/s- 2Gb/s per wire f/d, widths 2x,4x...32x

Network technologies

- Extended frames :
  - Alteon Jumbograms (MTU up to 9000)
- Interrupt suppression (coalescing)
- Checksum offloading

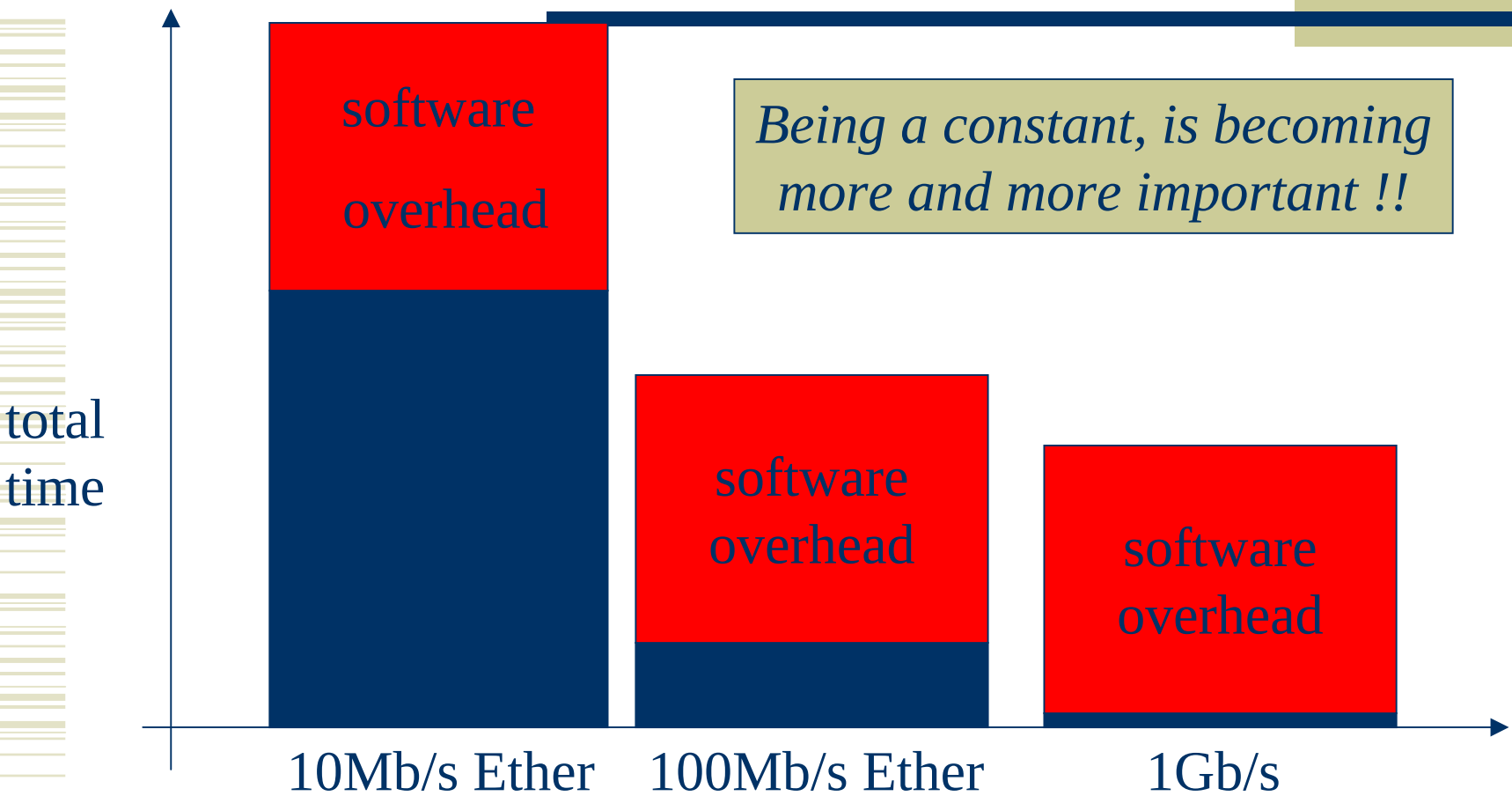


(from Gallatin 1999)

Software challenges

- ULNI (User Level Network Interface), OS-bypass
- Zero copy
- Network Path Pipelining

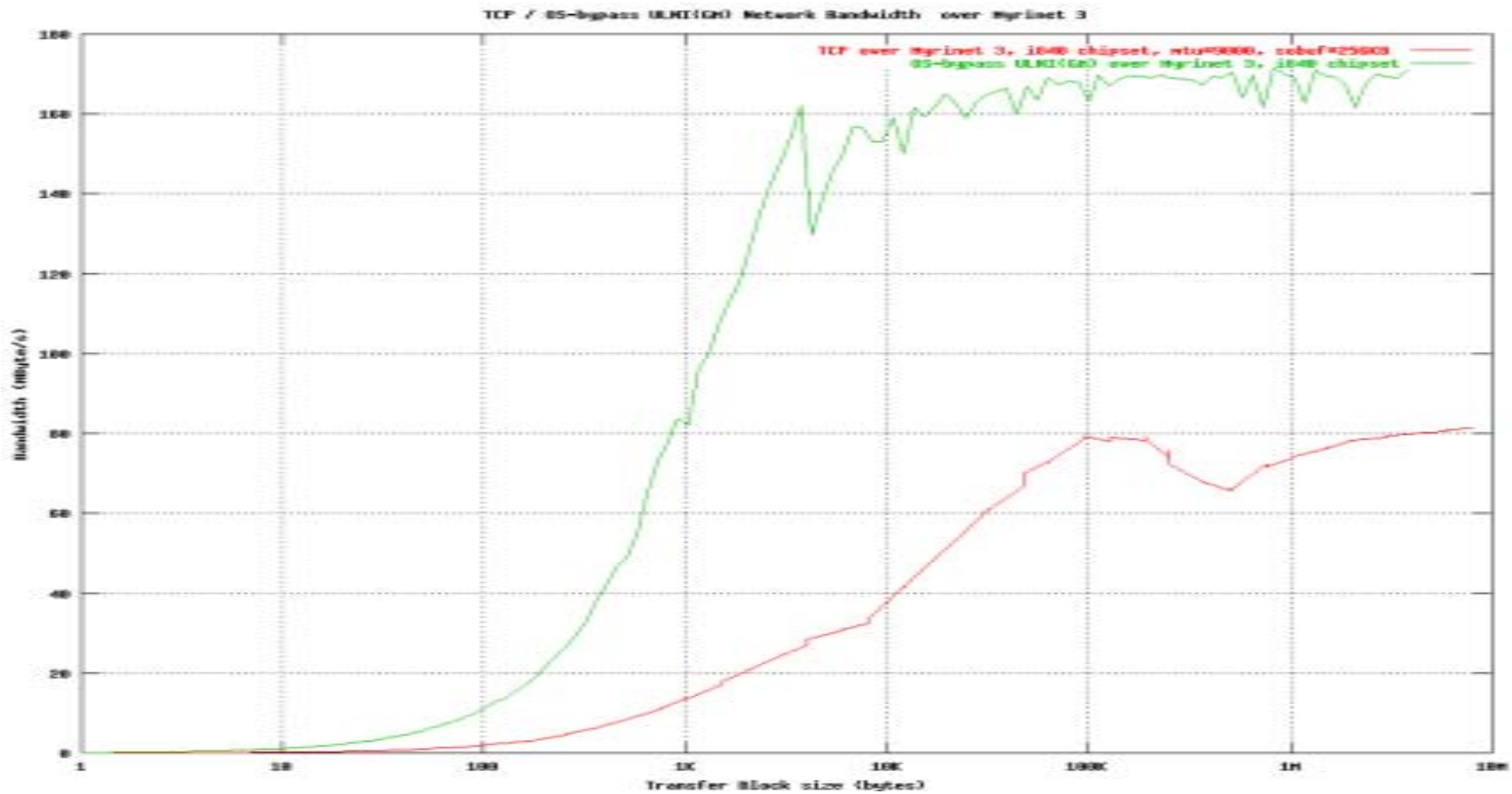
Software overhead



ULNI / OS-bypass

- Traditional networking can be defined as **in-kernel** networking : for each network operation (read / write) a kernel call (trap) is invoked
- This frequently is too expensive for **HPN** (high performance networks)
- Therefore in the last years, in the SAN environment **User Level Network Interface** (**ULNI**) / **OS-bypass** schemes have had a wide diffusion
- These schemes avoid the kernel intervention for read/write operations. The user process directly reads from/writes to the NIC memory using special user space libraries. Examples : U-Net, Illinois Fast Messages, ...
- **VIA** (**Virtual Interface Architecture**) is a proposed ULNI industry standard backed by Intel, Compaq et al. : **VIA over IP** is an internet draft

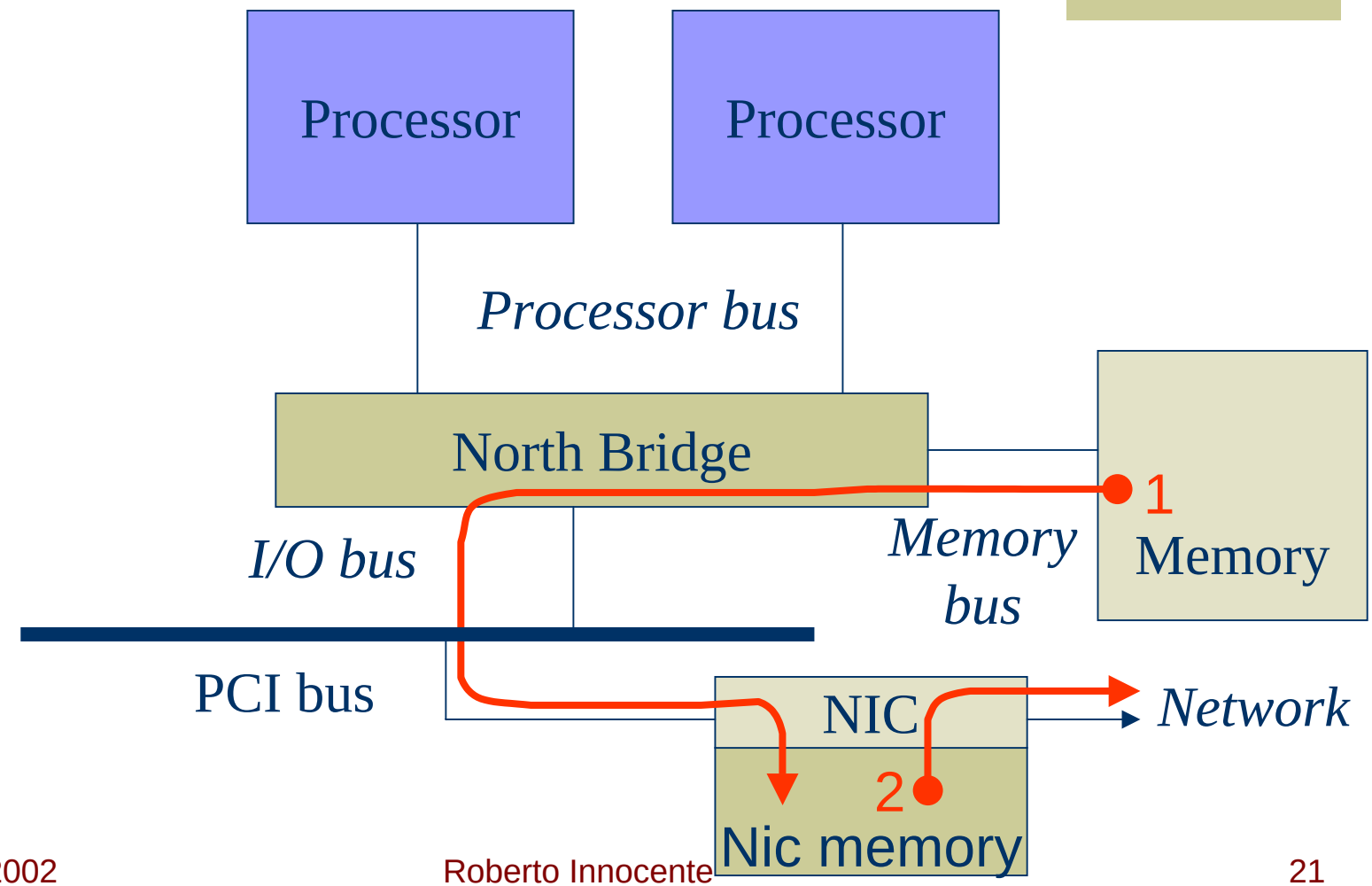
Standard TCP / OS-bypass ULNI(gm) on i840 chipset



Advanced networking

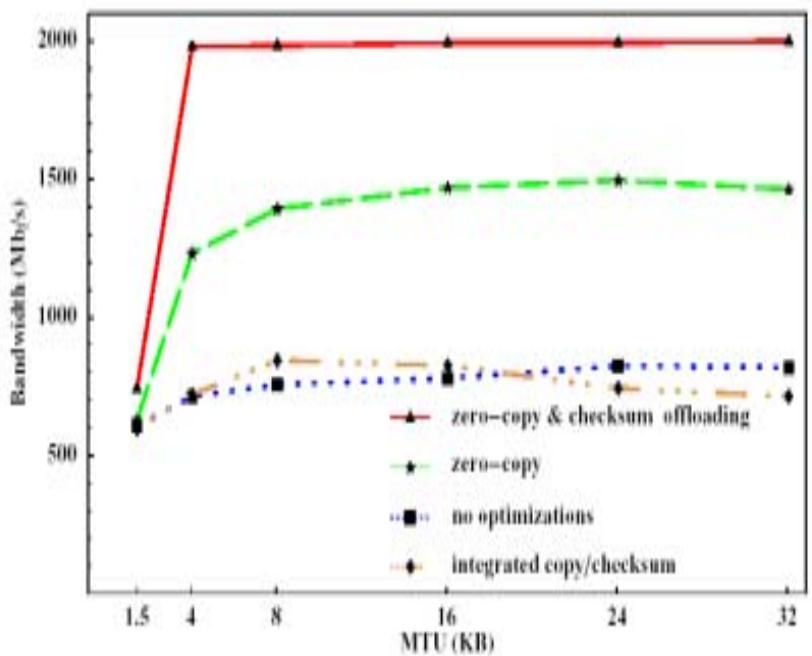
- Traditional UNIX i/o calls have an implicit *copy semantics.* It is possible to diverge from this creating a new API that avoids this requirement.
- Zero-copy can be obtained in different ways :
  - user/kernel page remapping (trapeze bsd drivers) : performance strongly depends on particular h/w, usually a Copy On Write flag preserves the copy semantic
  - fbufs (fast buffers , Druschel 1993) : shared buffers

Zero-copy data flow



Trapeze driver for Myrinet2000 on freeBSD

- zero-copy TCP by page remapping and copy-on -write
- split header/payload
- gather/scatter DMA
- checksum offload to NIC
- adaptive message pipelining
- 220 MB/s (1760 Mb/s) on Myrinet2000 and Dell Pentium III dual processor



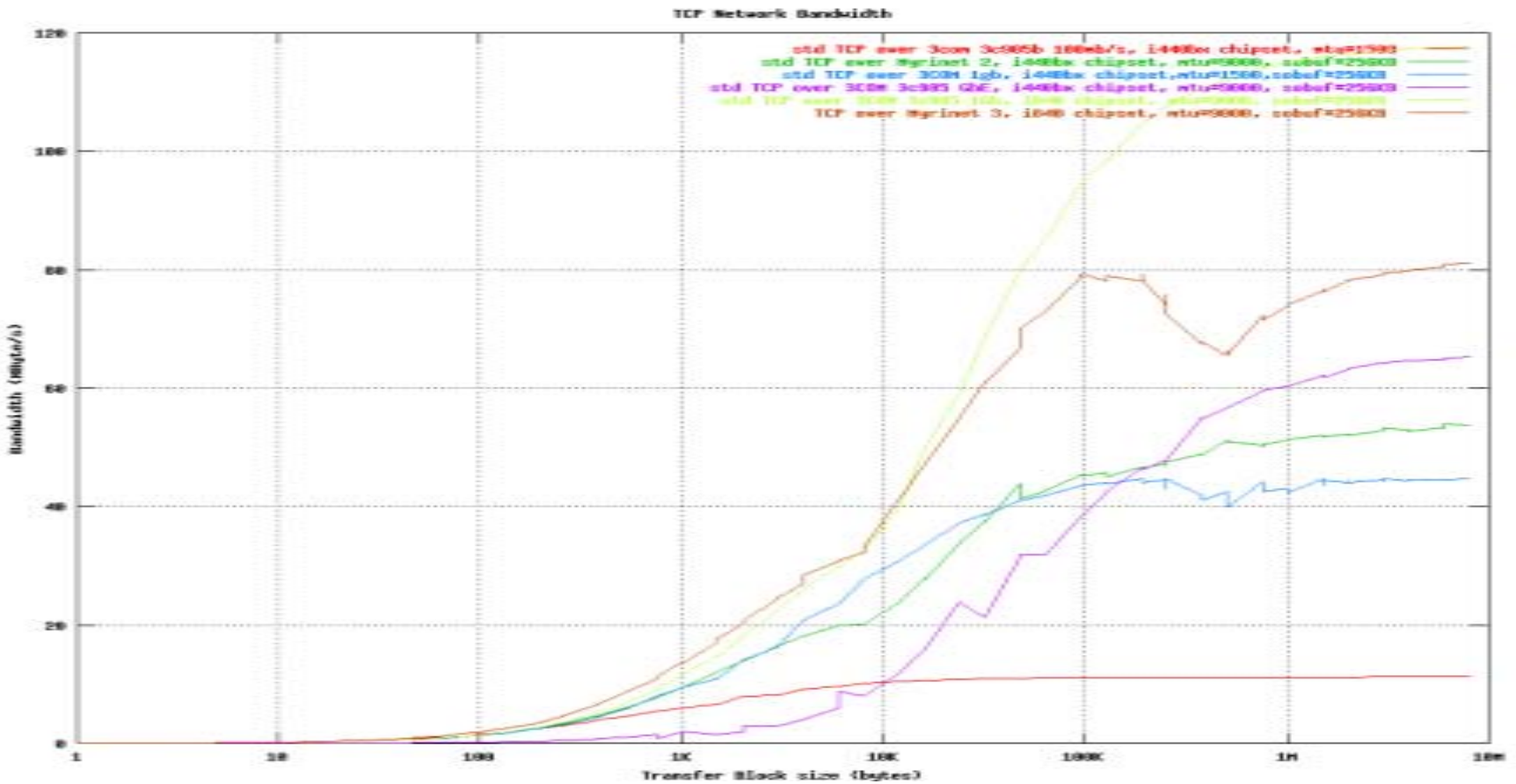
The application does'nt touch the data (from Chase, Gallatin, Yocum 2001)

Fbufs (Druschel 1993)

- Shared buffers between user processes and kernel
- It is a general schemes that can be used for either network or I/O operations
- Specific API without copy semantic
- A buffer pool specific for a process is created as part of process initialization
- This buffer pool can be pre-mapped in both the user and kernel address space

- `uf_read(int fd,void**bufp,size_t nbytes)`
- `uf_get(int fd,void** bufpp)`
- `uf_write(int fs,void*bufp,size nbytes)`
- `uf_allocate(size_t size)`
- `uf_deallocate(void*ptr ,size_t size)`

Gigabit Ethernet TCP



TOE
(TCP Off-loading Engine)

Network Path Pipelining

- It is frequently believed that the overhead on network operations decreases as the size of the frame increases, and therefore in principle an infinite frame would allow the best performance
- This in effect is false (e.g. Prilly 1999) for lightweight protocols : Myrinet, QSnet (does'nt apply to protocols like TCP that require a checksum header)
- Myrinet does'nt have a maximum frame size, but to efficiently use the network path, it is necessary that all the entities (sending CPU, sending **NIC** DMA, sending i/f, receiving i/f,...) along the path could work at the same time on different segments (pipelining). Using long frames inhibits co-working.
- It is a linear programming exercise to find the best frame size : it is different for different packet sizes.
- On PCI frequently the best frame size is the maximum transfer size allowed by the MLT (**maximum latency timer**) of the device : 256 or 512 bytes

Network protocols challenges

- Active Queue Management (AQM):
  - FQ, WFQ, CSFQ, RED
- ECN
- MPLS
- TCP Congestion avoidance/control
- TCP friendly streams
- IP RDMA protocols

AQM (active queue management)

- RFC 2309 Braden et al. :

  Recommandation on Queue Management and Congestion Avoidance in the Internet (Apr 1998)

- “Internet meltdown”, “congestion collapse” first experienced in the mid ’80
- first solution was V.Jacobson *congestion avoidance* mechanism for TCP (from 1986/88 on)
- Anyway, there is a limit to how much control can be accomplished from the edges of the network

AQM /2

- queue disciplines :
  - tail drop
    - a router sets a max length for each queue, when the queue is full it drops new packets arriving until the queue decreases because a pkt from the queue has been txmitted
  - head drop
  - random drop

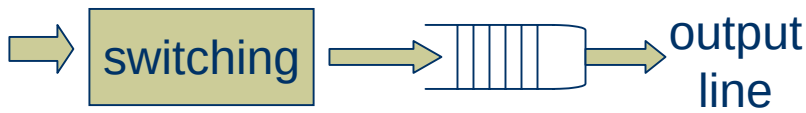
AQM /3

Problems of tail-drop discipline:

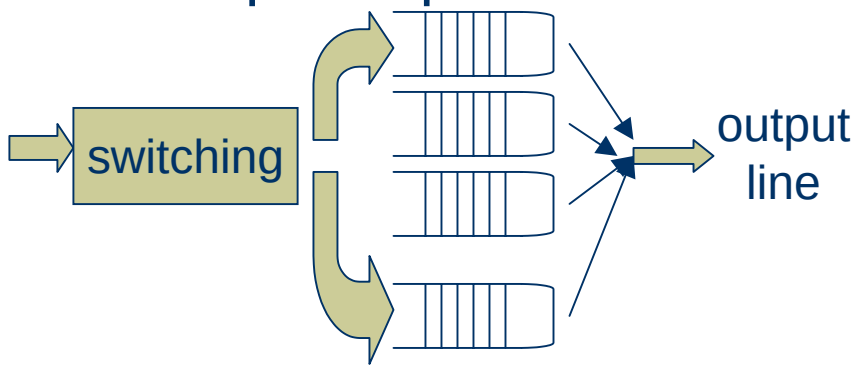
- Lock-Out :
  - in some situations one or a few connections monopolize queue space. This phenomenon is frequently the result of synchronization
- Full Queues :
  - queues are allowed to remain full for long periods of time, while it is important to reduce the steady state size (to guarantee short delays for applications requiring them)

Fair Queuing

- Traditional routers route packets independently. There is no memory, no state in the network.
- Demers, Keshav, Shenker (1990) :
  - Analysis and simulation of a **Fair Queuing** Algorithm
  - Incoming traffic is separated into flows, each flow will have an equal share of the available bandwidth.
  - It approximates the sending of 1 bit for each ongoing flow
- Statistical FQ (hashing flows)



Traditional queuing:
one queue per line



Fair queuing :
one queue per flow

Weighted Fair Queuing (WFQ)

This queuing algorithm was developed to serve real time applications with guaranteed bandwidth.

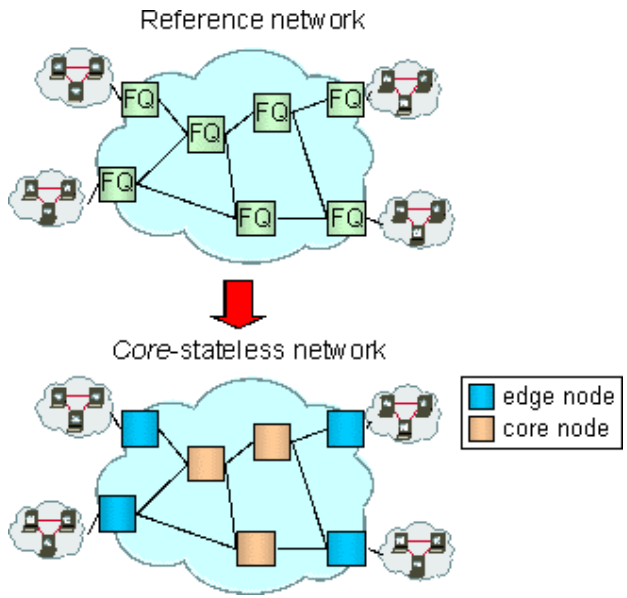
- L.Zhang (1990):

  Virtual clock: A new Traffic Control Algorithm for Packet Switching Networks

  Traffic is classified, and for each class a percentage of the available b/w is assigned. Packets are sent to different queues according to their class. Each pkt in the queue is labelled with a calculated **finish time.** Pkts are transmitted in finish time order.

CSFQ (Core stateless Fair Queuing)

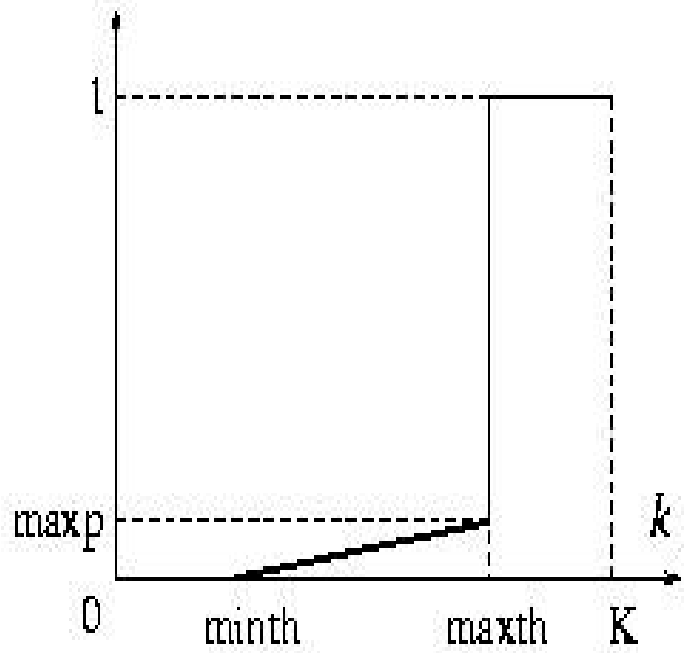
- Only ingress/egress routers perform packet classification/per flow buffer mgmt and scheduling
- Edge routers estimate the incoming rate of each flow and use it to label each pkt
- Core routers are stateless
- All routers from time to time compute the fair rate on outgoing links
- This approximates FQ



(from Stoica,Shenker,Zhang1998)

RED /1 (Random Early Detection)

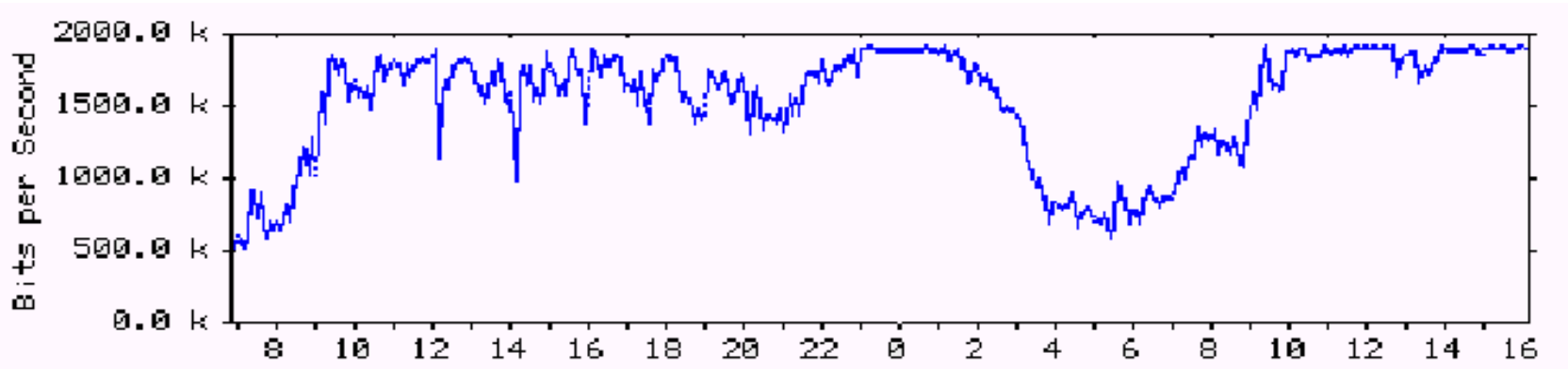
- Floyd, V.Jacobson 1993
- Detects incipient congestion and drops packets with a certain probability depending on the queue length. The drop probability increases linearly from 0 to *maxp* as the queue length increases from *minth* to *maxth*. When the queue length goes above *maxth (maximum threshold)* all pkts are dropped



RED /2

- It is now widely deployed
- There are very good performance reports
- Still an active area of research for possible modifications
- Cisco implements its own WRED (Weighted Random Early Detection) that discards packets based also on precedence (provides separate threshold and weights for different IP precedences)

RED /3



- (from Van Jacobson, 1998) Traffic on a busy E1 (2 Mbit/s) Internet link. RED was turned on at 10.00 of the 2nd day, and from then utilization rised up to 100% and remained there steady.

ECN (Explicit Congestion Notification) /1

- RFC 3168 Ramakrishnan, Floyd, Black :

  The addition of Explicit Congestion Notification (ECN) to IP (Sep 2001)
- Uses 2 bits of the IPv4 Tos (Now and in IPv6 reserved as a DiffServ codepoint). These 2 bits encode the states :
  - ECN-Capable Transport : ECT(0) and ECT(1)
  - Congestion Experienced (CE)
- If the transport is TCP, uses 2 bits of the TCP header, next to the Urgent flag :
  - ECN-Echo(ECE) set in the first ACK after receiving a CE pkt
  - Congestion Window Reduced (CWR) set in the first pkt after having reduced cwnd in response to an ECE pkt
- With TCP the ECN is initialized sending a SYN pkt with ECE and CWR on, and receiving a SYN+ACK pkt with ECE on

ECN /2

How it works:

- senders set ECT(0) or ECT(1) to indicate that the end-nodes of the transport protocol are ECN capable
- a router experiencing congestion, sets the 2 bits of ECT pkts to the CE state (instead of dropping the pkt)
- the receiver of the pkt signals back the condition to the other end
- the transports behave as in the case of a pkt drop (no more than once in a RTT)

Linux adopted it on 2.4 ... many connection troubles (the default now is to be off, can be turned on/off using : */proc/sys/net/ipv4/tcp_ecn*). Major sites are using firewalls or load balancers that refuse connections from ECT (Cisco PIX and Load Director, etc).

MPLS (Multi Protocol Label Switching)

This technology is a marriage of traffic engineering as on ATM and IP :

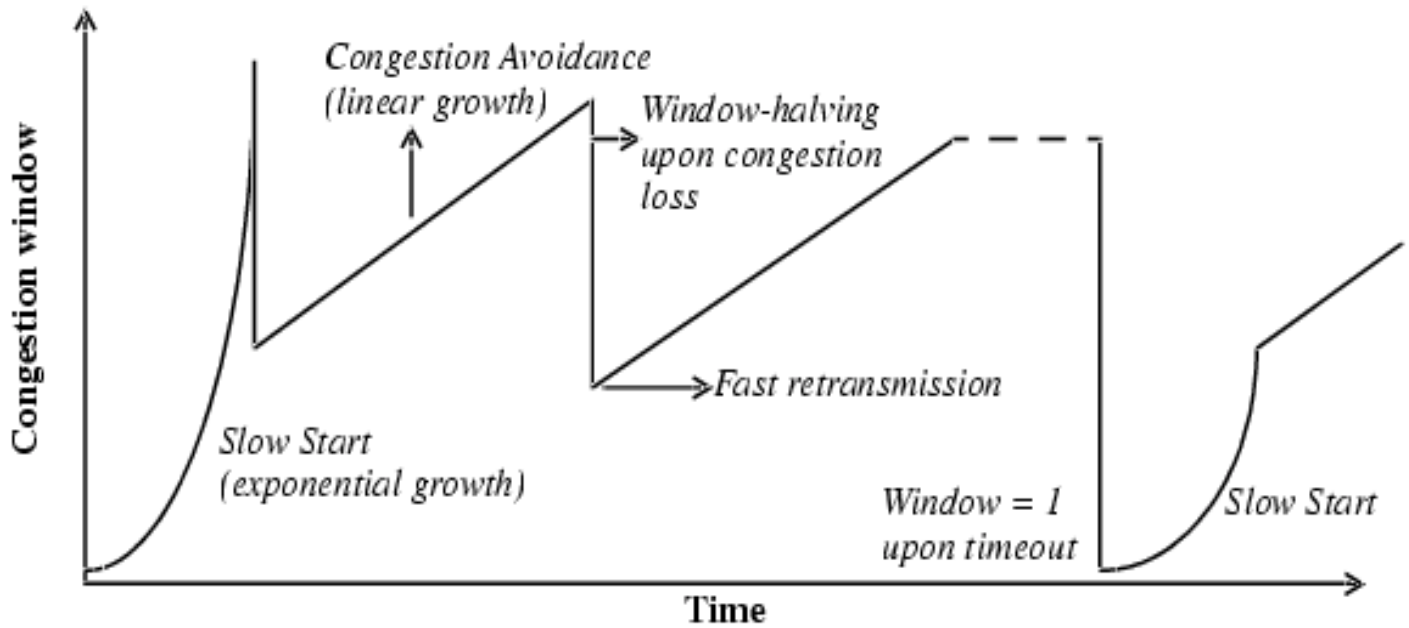
- Special routers called **LER (Label Edge Routers)** mark the packets entering the net with a label that indicates a class of service (**FEC or Forward Equivalence Class**) or priority : the 32 bit header (a Shim header) is inserted between the OSI level 2 header and upper level headers (after the ethernet header, before the IP header): 20 bits for the label, 3 experimental, 1 bit for stack function and 8 bit for TTL
- The IP packet becomes an MPLS pkt
- Internal routers work as **LSR (Label Switch Routers)** for MPLS pkts : will look up and follow the label instructions (usually swapping label)
- Between MPLS aware devices **LSP (Label Switch Paths)** are established and designed for their traffic characteristics

Congestion control in TCP/1

Sending rate is limited by a congestion window (cwnd): the maximum # of pkts to be sent in a RTT(round trip time). Phases:

1. *Slow start :* cwnd is increased exponentially (# pkts acknowledged in a RTT are summed to cwnd) up to a threshold *ssthresh* or a loss
2. *Congestion avoidance:* AIMD (Additive increase, Multiplicative decrease) phase. cwnd is increased by 1 each RTT. When there is a loss, cwnd is halved.

Congestion control in TCP/2



from [Balakrishnan 98]

AIMD / AIPD Congestion control

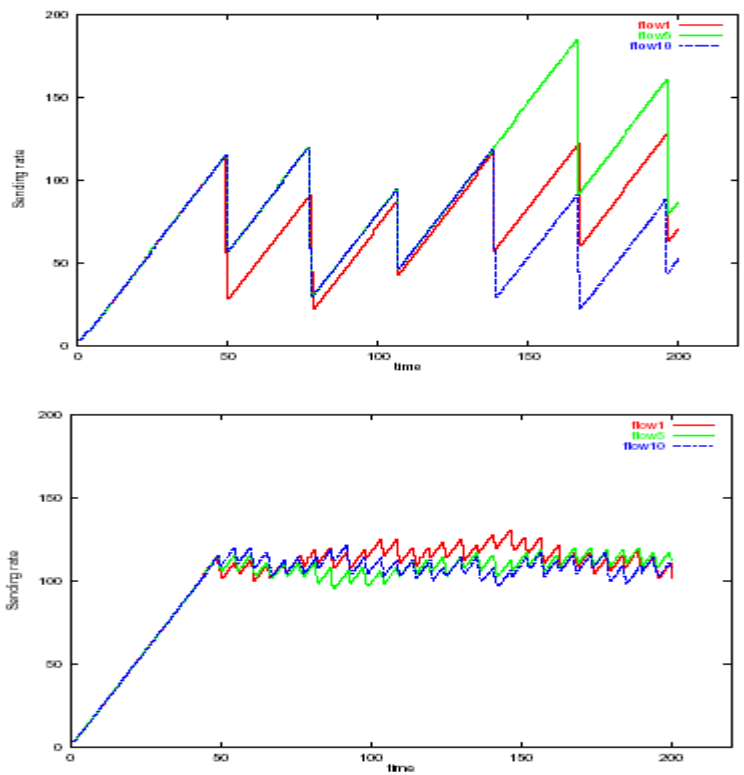
- AIMD (Additive Increase Multiplicative Decrease) :
  - W = W + a (no loss)
  - W = W * (1 – b)

    (L > 0 losses)

  it achieves a b/w ∝1/sqrt(L)
- AIPD (Additive Increase Proportional Decrease):
  - W = W + a (no loss)
  - W = W * (1 –b *L)

    (L > 0 losses)

  it achieves a b/w ∝1/L



(ns2 plot of 3 flows,source Lee 2001)

TCP stacks

- **Tahoe**(4.3BSD,Net/1) : implements Van Jacobson slow start/congestion avoidance and fast retransmit algorithms
- **Reno**(1990,Net/2) : fast recovery, TCP header prediction
- **Net/3**(4.4BSD 1994): multicasting, LFN (Long Fat Networks) mods
- **NewReno** : fast retransmit even with multiple losses (Hoe 1997)
- **Vegas**: experimental stack (Brakmo, Peterson 1995)

TCP Reno

- It is the most widely referenced implementation, basic mechanisms are the same also in Net/3 and NewReno
- It is biased against connections with long delays (large RTT) : in this case the increase of the *cwnd* happens slowly and so they obtain less avg b/w

LFN Optimal Bandwidth and TCP Reno

- Let's consider a 1 Gb/s WAN connection with a RTT of 100 ms (BW*rtt ~12 MB)
- Optimal congestion window would be about 8.000 segments (1.5k)
- When there will be a loss cwnd will be decreased to 4.000 segments
- It will take 400 seconds (7minutes!) to recover to the optimal b/w. This is awful if the network is not congested !

TCP Vegas

- Approaches congestion but tries to avoid it. Estimates the available b/w looking at variation of delays between acks (CARD: Congestion avoidance by RTT delays)
- It has been shown that it is able to better utilize the available b/w (30% better than Reno)
- It is fair with long delay connections
- Anyway, when mixed with TCP Reno it gets an unfair share (50% less) of the bandwidth

Restart of Idle connections

- The suggested behaviour after an idle period (more than a RTO without exchanges) is to reset the congestion window to its initial value (usually cwnd=1) and apply slowstart
- This can have a very adverse effect on applications using e.g. MPI-G
- **Rate Based Pacing (RBP)**: *Improving restart of idle TCP connections* ,Viesweswaraiah, Heidemann (1997) suggests to reuse the previous cwnd , but to smoothly pace out pkts, rather than burst them out, using a Vegas like algorithms to estimate the rate

TCP buffer tuning and parallel streaming

- The optimal TCP flow control window size is the bandwidth delay product for the link. In these years as network speeds have increased, o.s. have adjusted the default buffer size from 8KB to 64KB. This is far too small for hpn. An OC12 link with 50 ms rtt delay requires at least 3.75 MB of buffers ! There is a lot of reseach on mechanisms to automatically set an optimal flow control window and buffer size, just some examples :
  - **Auto-tuning** (Semke,Mahdavi,Mathis 1998)
  - **Dynamic Right Sizing**(DRS) :
    - Weigl, Feng 2001 : *Dynamic Right-Sizing: A Simulation study .*
  - **Enable** (Tierney et al LBNL 2001) : database of BW-delay products
  - Linux 2.4 **autotuning**/connection **caching**: controlled by the new kernel variables net.ipv4.tcp_wmem/tcp_rmem, the advertised receive window starts small and grows with each segment from the transmitter; tcp control info for a destination are cached for 10 minutes(cwnd,rtt,rttvar,sshthresh)
- University of Illinois ***psocket*** library to support parallel sockets.

TCP friendly streams

- It is essential to reduce the amount of streams unresposive to network congestion to avoid *congestion collapses*
- It is important to develop congestion algorithms that are fair when mixed with current TCP, because this constitutes the vast majority of traffic
- **TCP friendly streams** are streams that exhibit a behavior similar to a TCP stream in the same conditions :

  upon congestion, if L is the loss rate, their bandwidth it's not more than 1.3*MTU/(RTT*sqrt(L))
- Unfortunately real time applications suffer too much for the TCP congestion window halving. For this reason a wider class of congestion control algorithms has been investigated. These algorithms are called **binomial algorithms** and they update their window according to (for TCP k=0,l=1):
  - Increase : W = W + a/(W^k)
  - Decrease: W = W – b*(W^l)

  For k+l=1 these algorithms are **TCP friendly** (Bansal, Balakrishnan 2000)

iWarp

- iWarp is an initial work on RDMA (Remote DMA). Internet drafts :
  - The Architecture of Direct Data Placement (DDP) and Remote Direct Memory Access (RDMA) on Internet Protocols, S.Bailey, February 2002
  - The Remote Direct Memory Access Protocol (iWarp) , S.Bailey, February 2002