# High Speed Low Loss Networking

Where Are My Packets?

Steffen Luitz

SLAC 4/29/02

# Outline

- Motivation
  - Networking in DAQ
- Introduction to Networks
  - The Basics: Protocols, Devices, Flow Control
  - The Real World
- The DAQ Application: Event Building
  - BaBar Event Builder
  - Generations of Switches
- The Gigabit/s Future

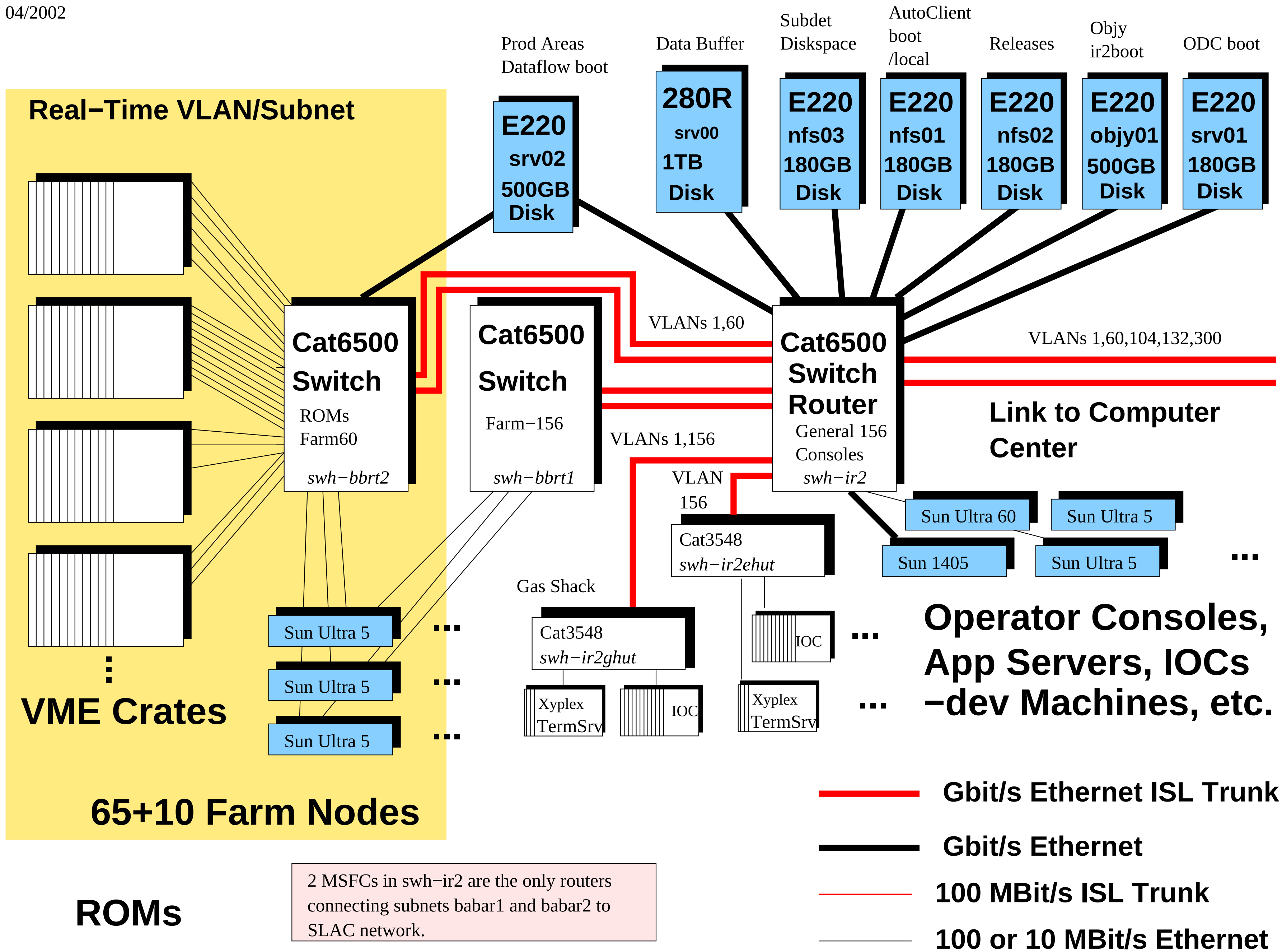
# Thanks

- BaBar Data Flow Group (past and present)
  - Ric Claus, Mike Huffer, Chris O'Grady, Amedeo Perazzo, Matt Weaver
- SCS Networking (past and present)
  - Gary Buhrmaster, Les Cottrell, Charley Granieri, Paola Grosso, Connie Logg, Dave Millsom, Davide Salomoni
- Our Cisco Representatives (past and present)

# Data Acquisition and Networking

- High-performance networking has become an integral part of HEP data acquisition
  - Network traffic on a good day in IR2
    - ~ 45MByte/s event building (real-time)
    - ~ 15Mbyte/s handling of data output (multiple copies) and other traffic (sweeps, backup, etc.)
- → Moving around at least 5 Tbyte/day with IP and Ethernet on IR-2 LAN (Local Area Network)

04/2002

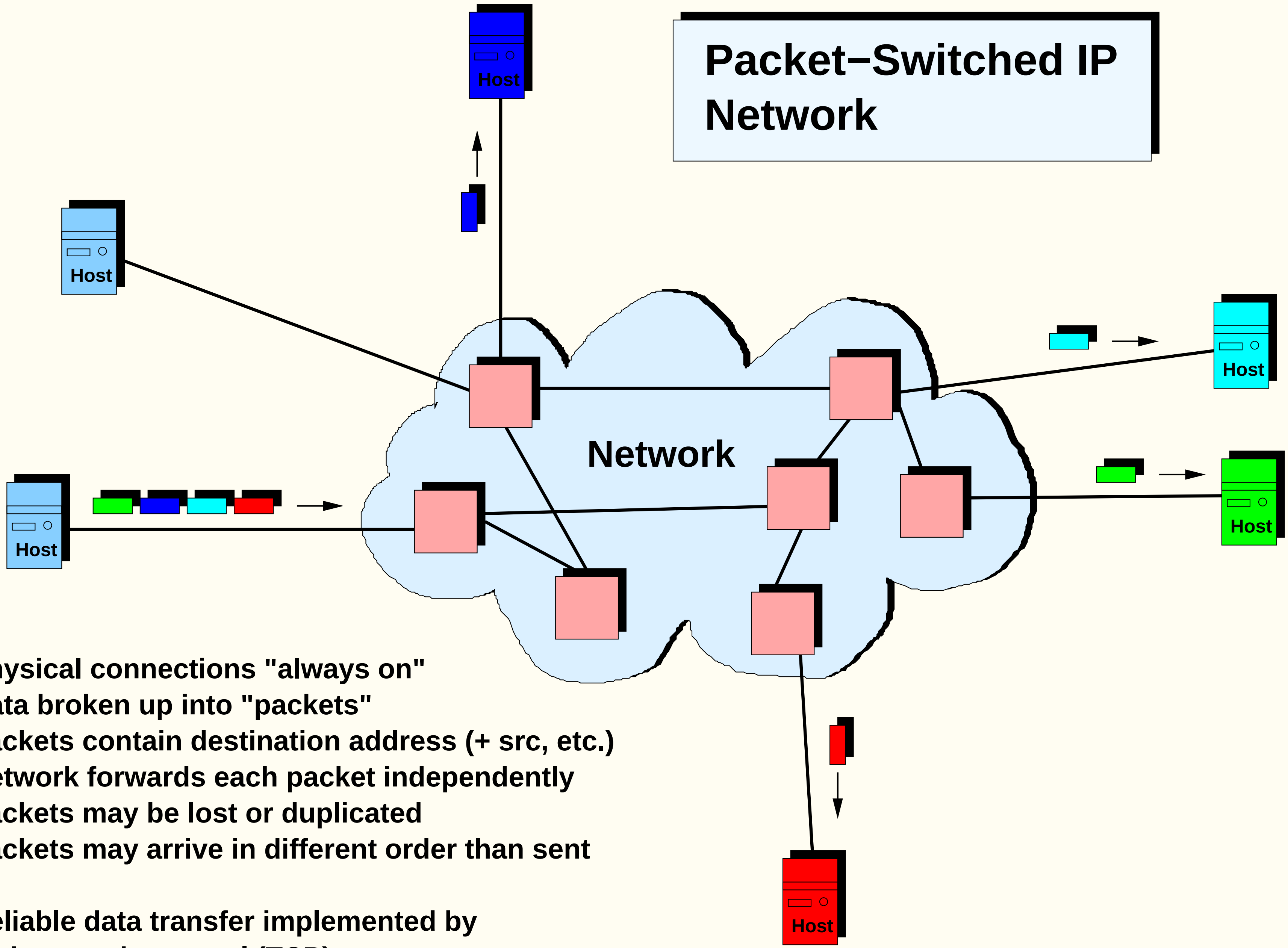



# BaBar Data Acquisition and Controls Networks

# The Basics And The Real World

- Protocols
- Devices
- Buffers
- Flow Control

# Packet−Switched IP Network



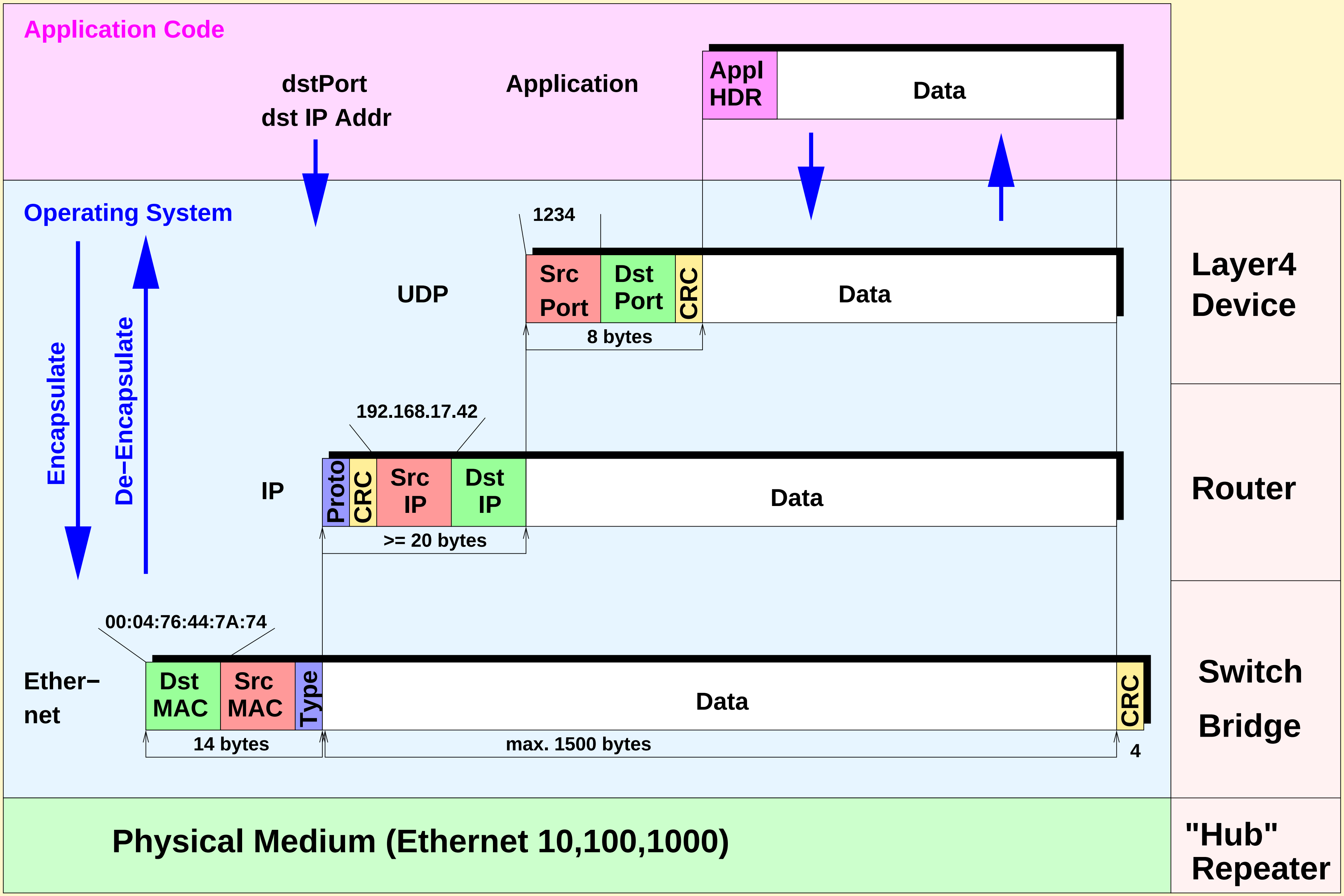

Physical connections "always on"
Data broken up into "packets"
Packets contain destination address (+ src, etc.)
Network forwards each packet independently
Packets may be lost or duplicated
Packets may arrive in different order than sent

Reliable data transfer implemented by
end−to−end protocol (TCP)

# Sending Data from Host A to B

- Split data into pieces
- Wrap data into “protocol”
- Wrap higher level protocols into lower level protocols (“protocol stack”)
- Send packets over “wire” from A to B
- Unwrap and combine data

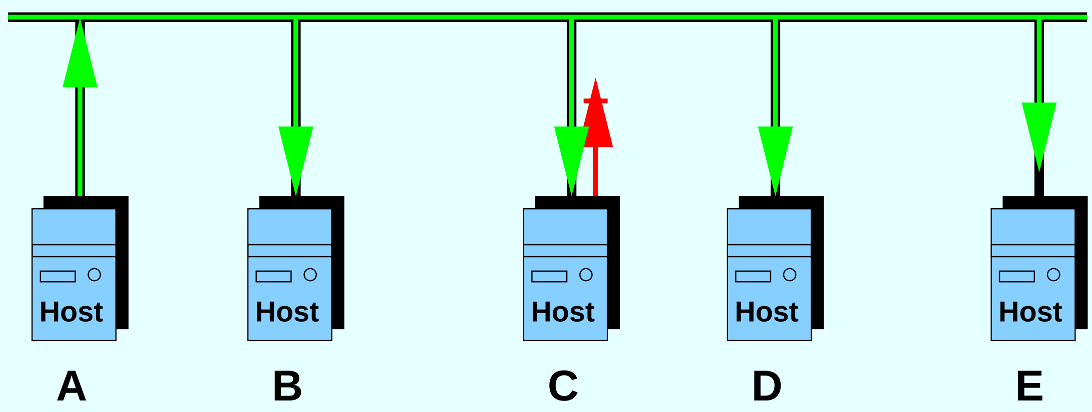
# Protocol Layers and Encapsulation (Ex: UDP)



# Some Protocols in the IP Suite

- ARP (Address Resolution Protocol)
  - Maps Ethernet addresses to IP addresses
- ICMP (Internet Control Message Protocol)
  - Error reporting / diagnostics / control
- IP (Internet Protocol)
- UDP (User Datagram Protocol)
  - Unreliable datagram service
- TCP (Transmission Control Protocol)
  - Reliable end-to-end data transport

## Shared Network

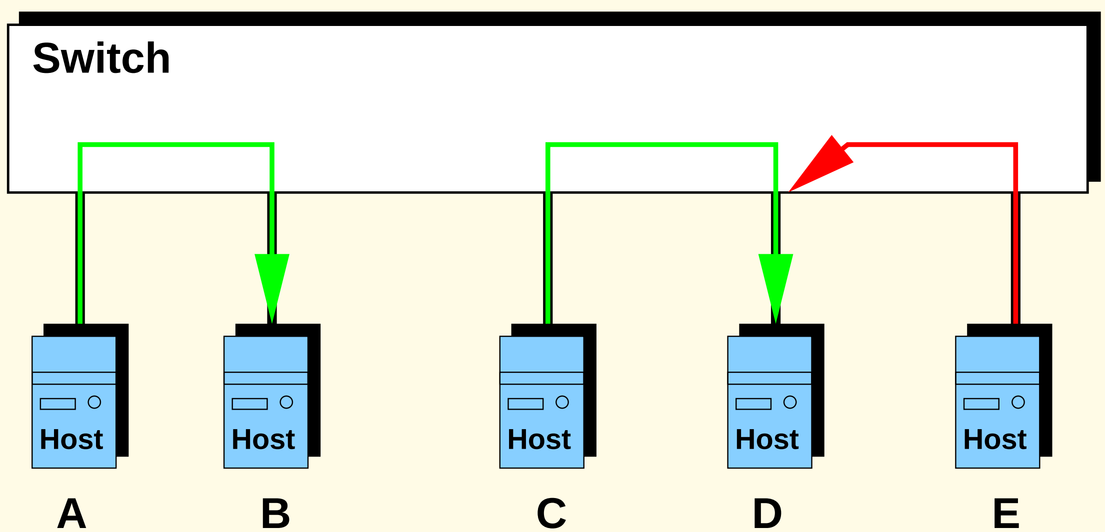



while host A is sending a packet to host B

host C cannot send a packet to host D

−> Arbitration needed

## Switched Network



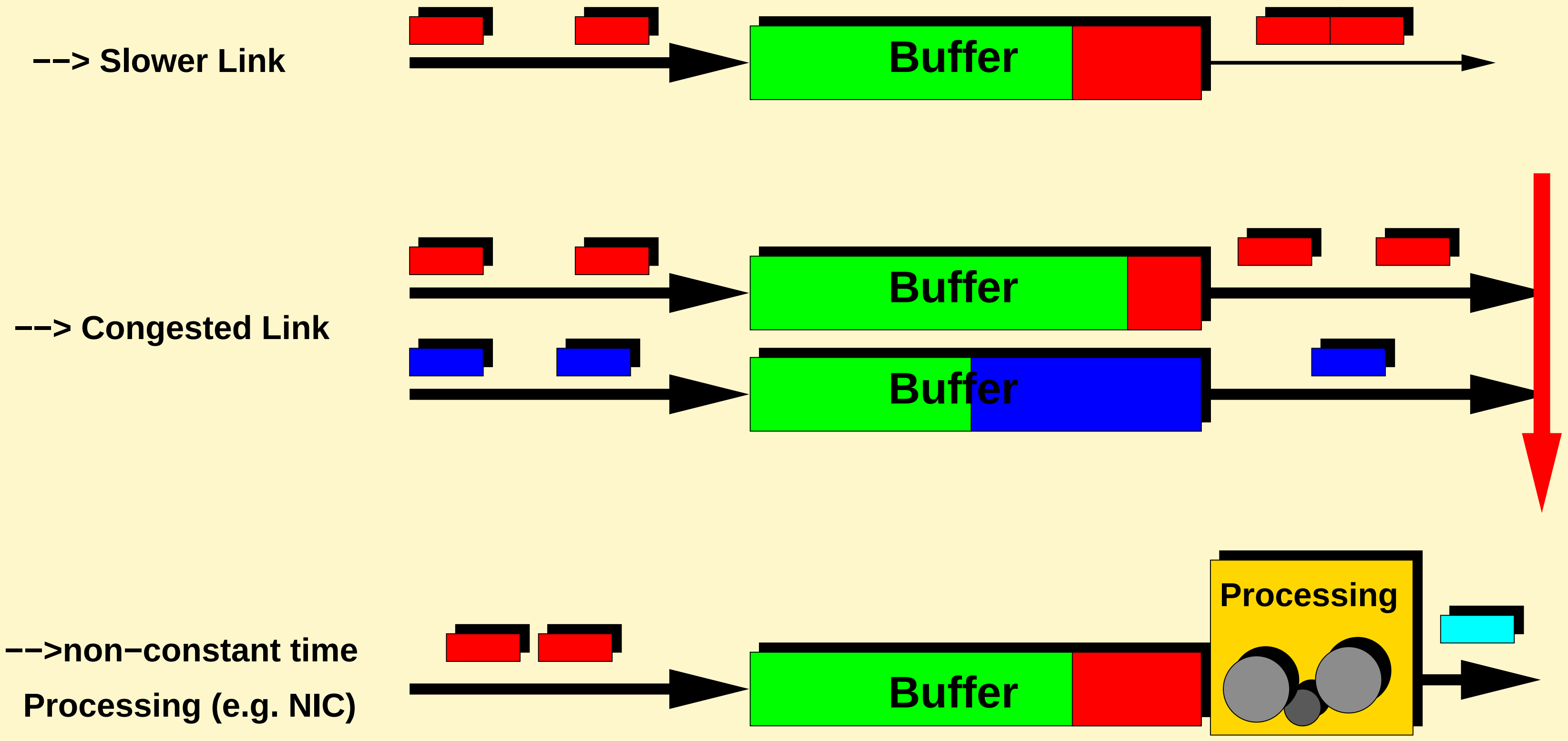

while host A is sending a packet to host B

host C can send a packet to host D at the same time

while switch is sending packet from C−>D it can’t send packet from E−>D at the same time

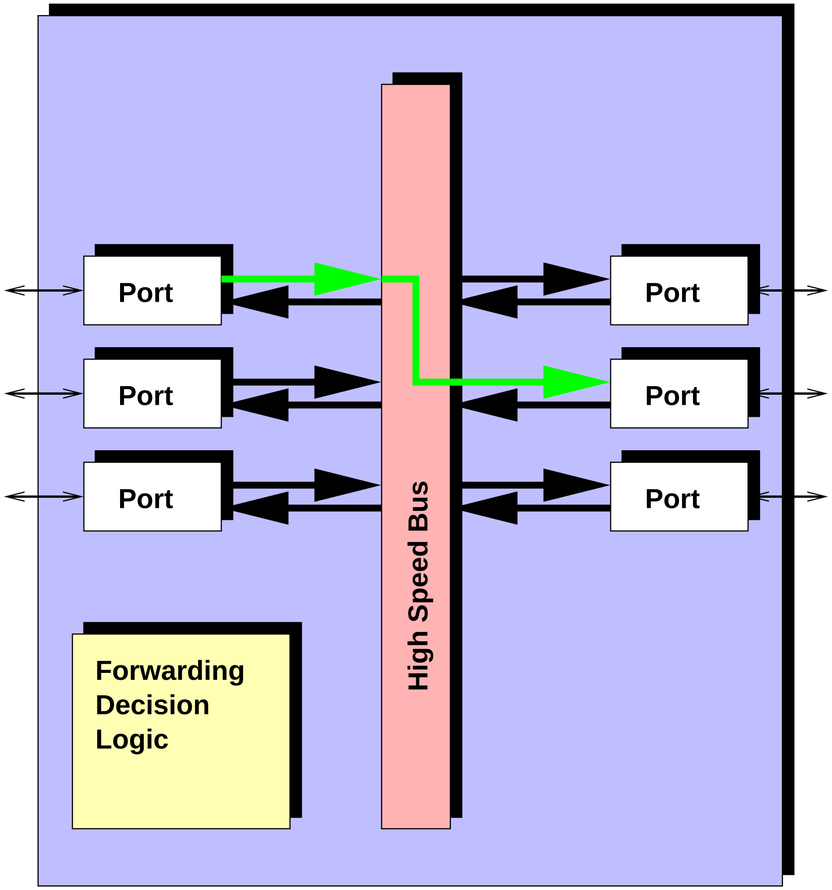
−−> buffer or drop

# Where To Find Buffers

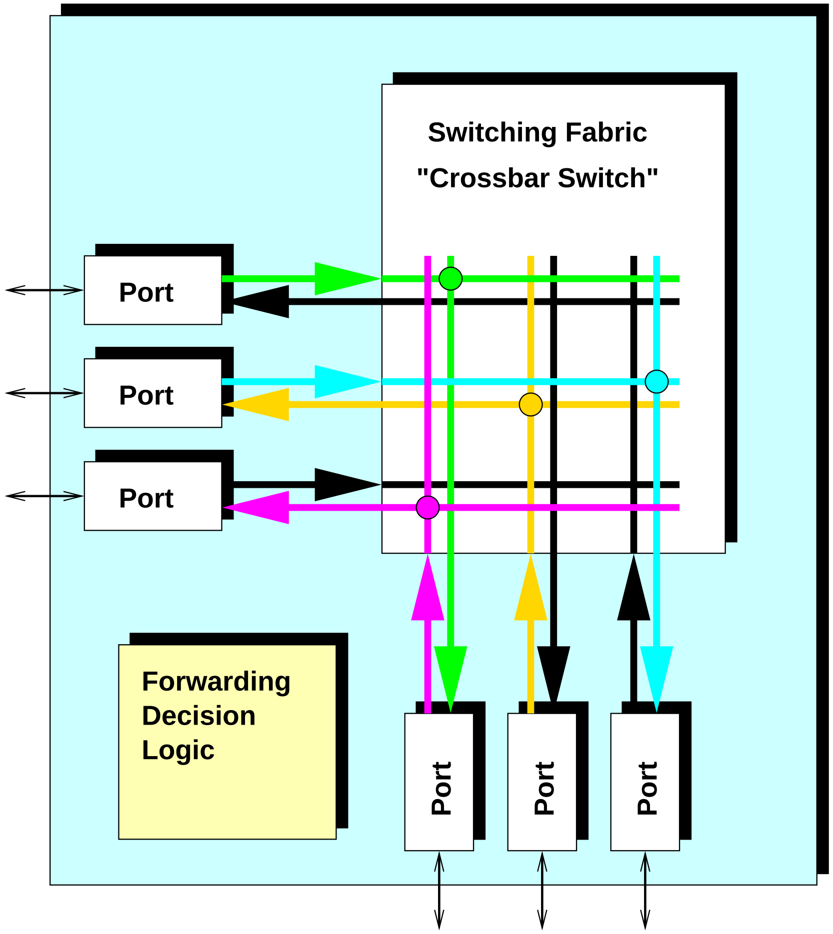


# Fundamental Switch Architectures

## Bus Architecture

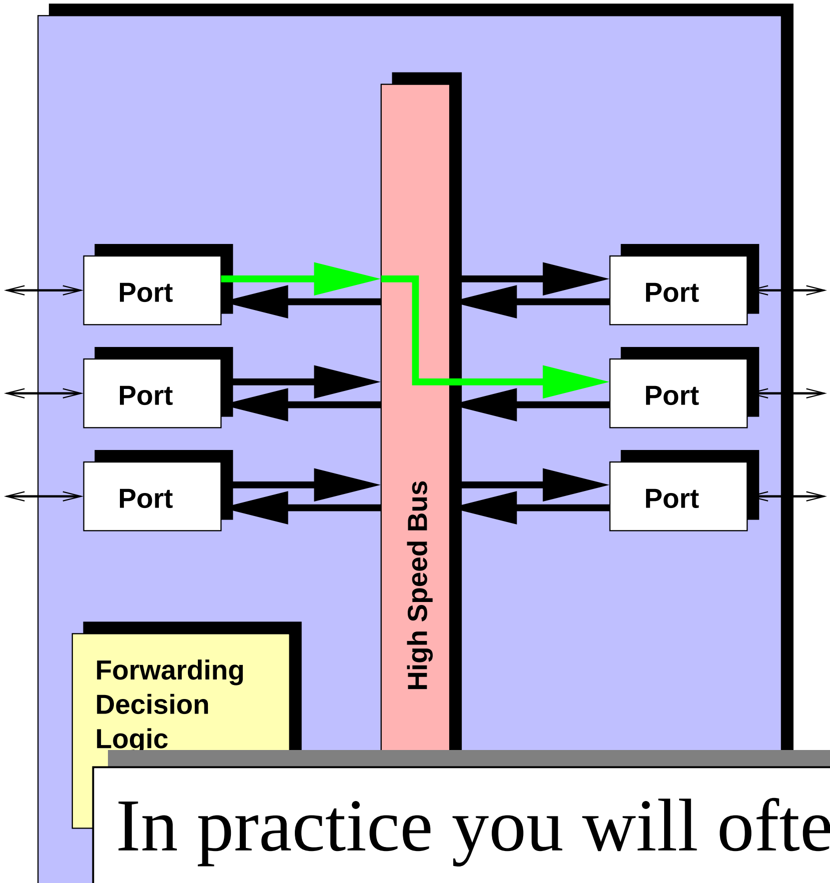



## Fabric Architecture

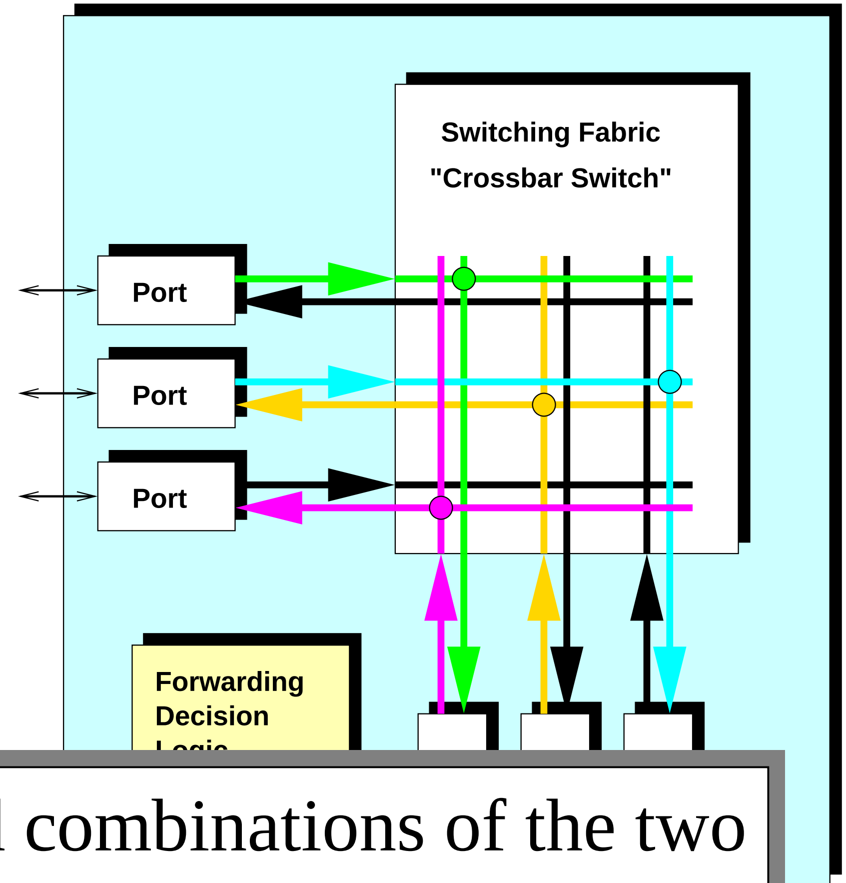


# Fundamental Switch Architectures

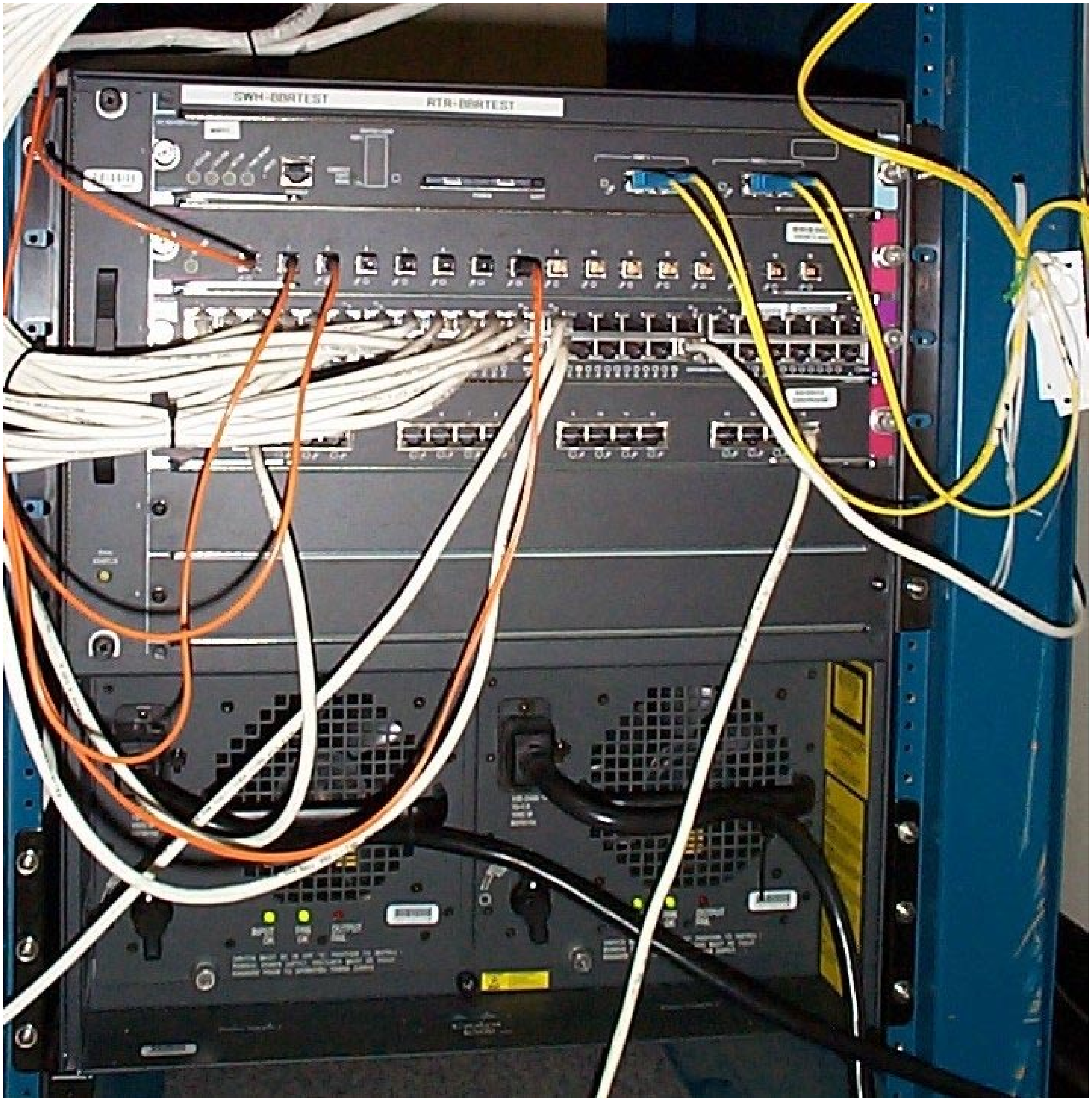
## Bus Architecture




## Fabric Architecture



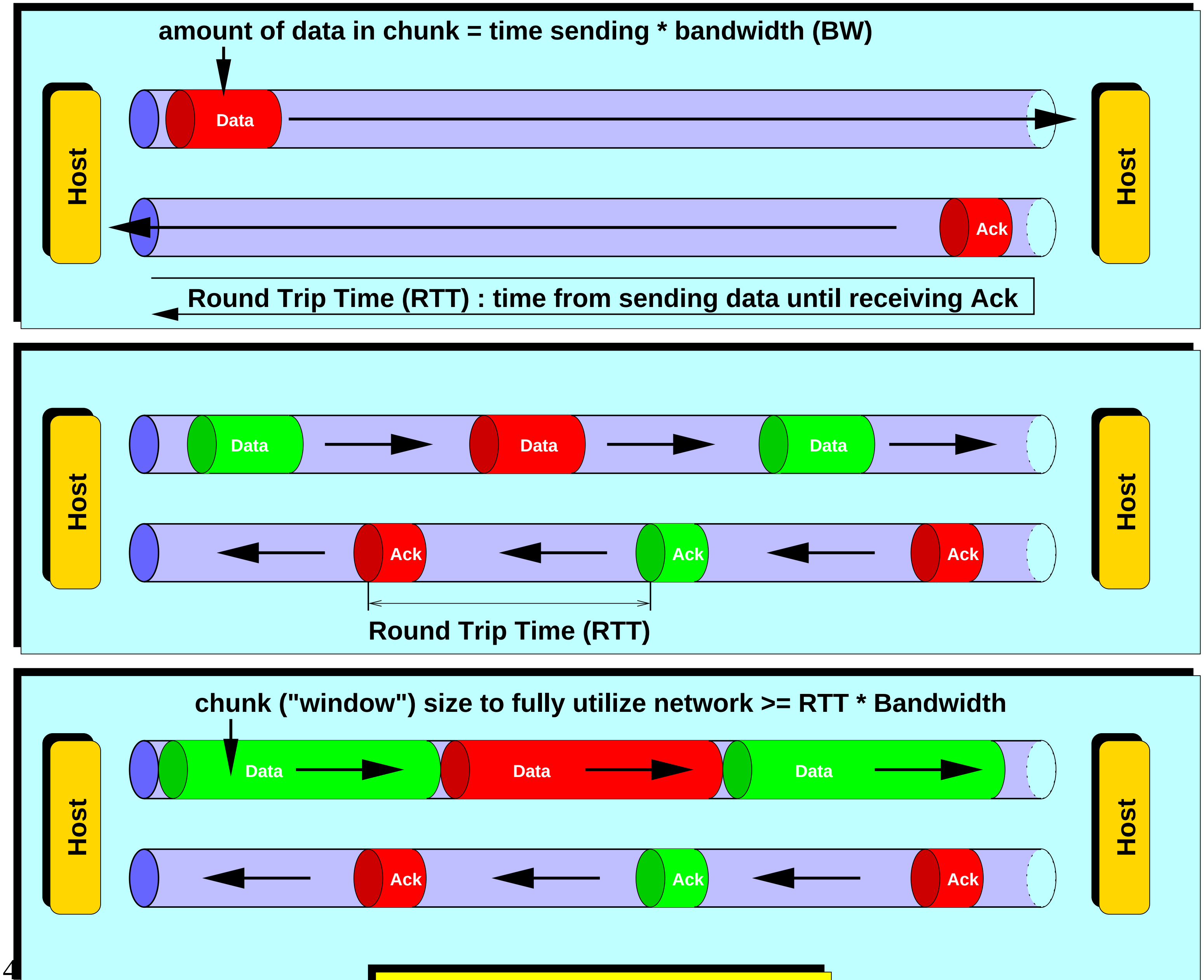

In practice you will often find combinations of the two architectures, e.g. multiple local busses interconnected by a switching fabric



# Simple Flow Control Protocol

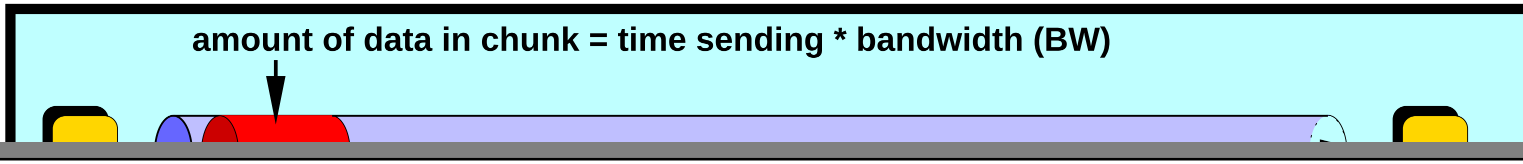
- Make sure the sender doesn't overrun the receiver
  - Send chunk of data to the receiver
  - Wait for acknowledge packet to come back
  - Repeat until all data has been sent
- Doesn't handle packet loss but illustrates the idea

# Bandwidth, Round Trip Time and Network Utilization




**WS >= BW * RTT**

## Bandwidth, Round Trip Time and Network Utilization

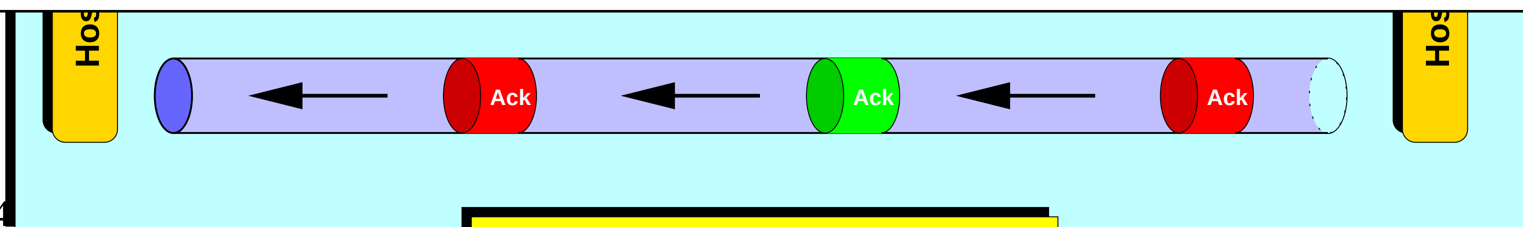



General formula for all window-based flow control schemes.

Links with large BW*RTT need

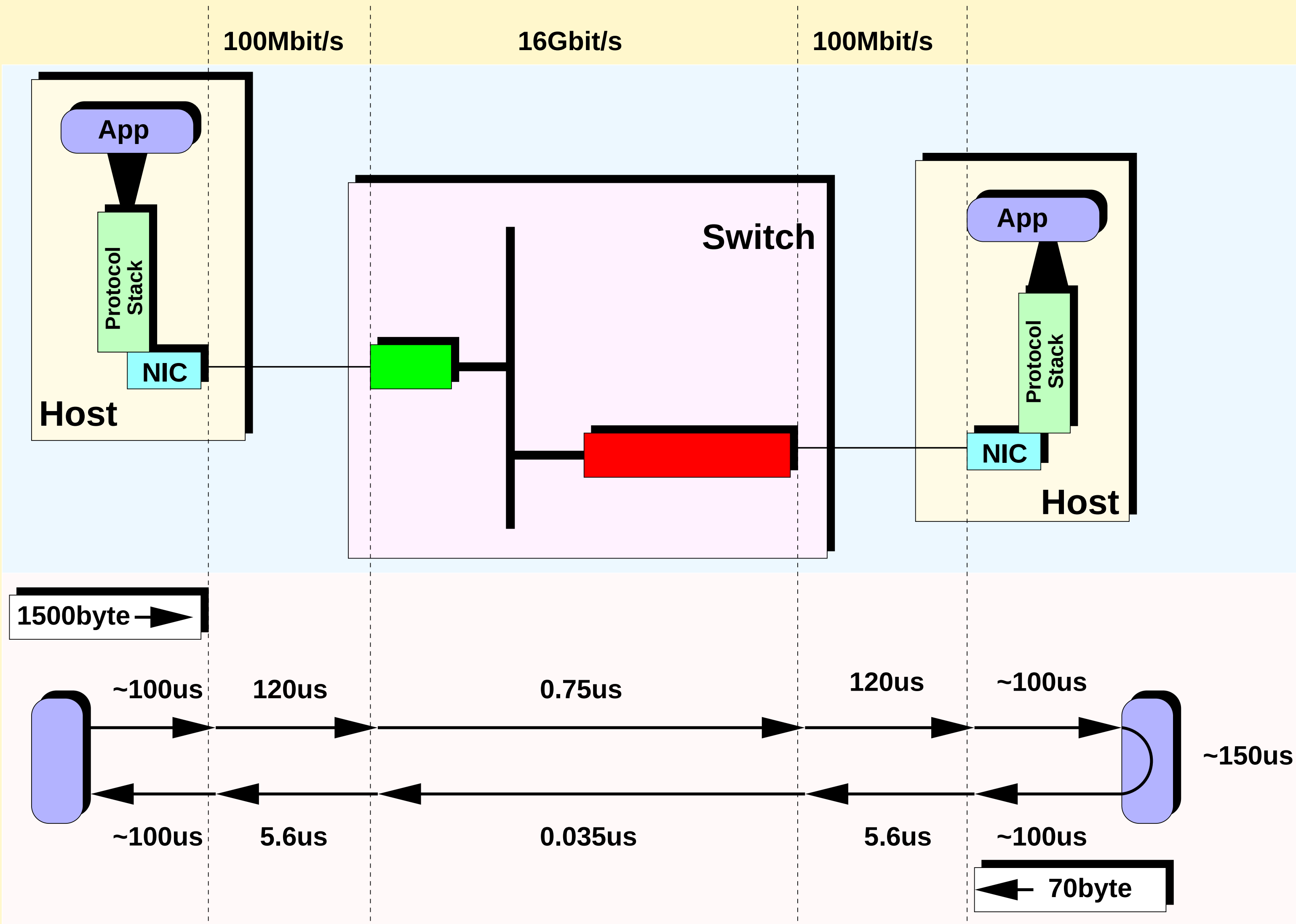
→ Large window sizes

→ Multiple streams (make per-stream bandwidth smaller by sharing the link)



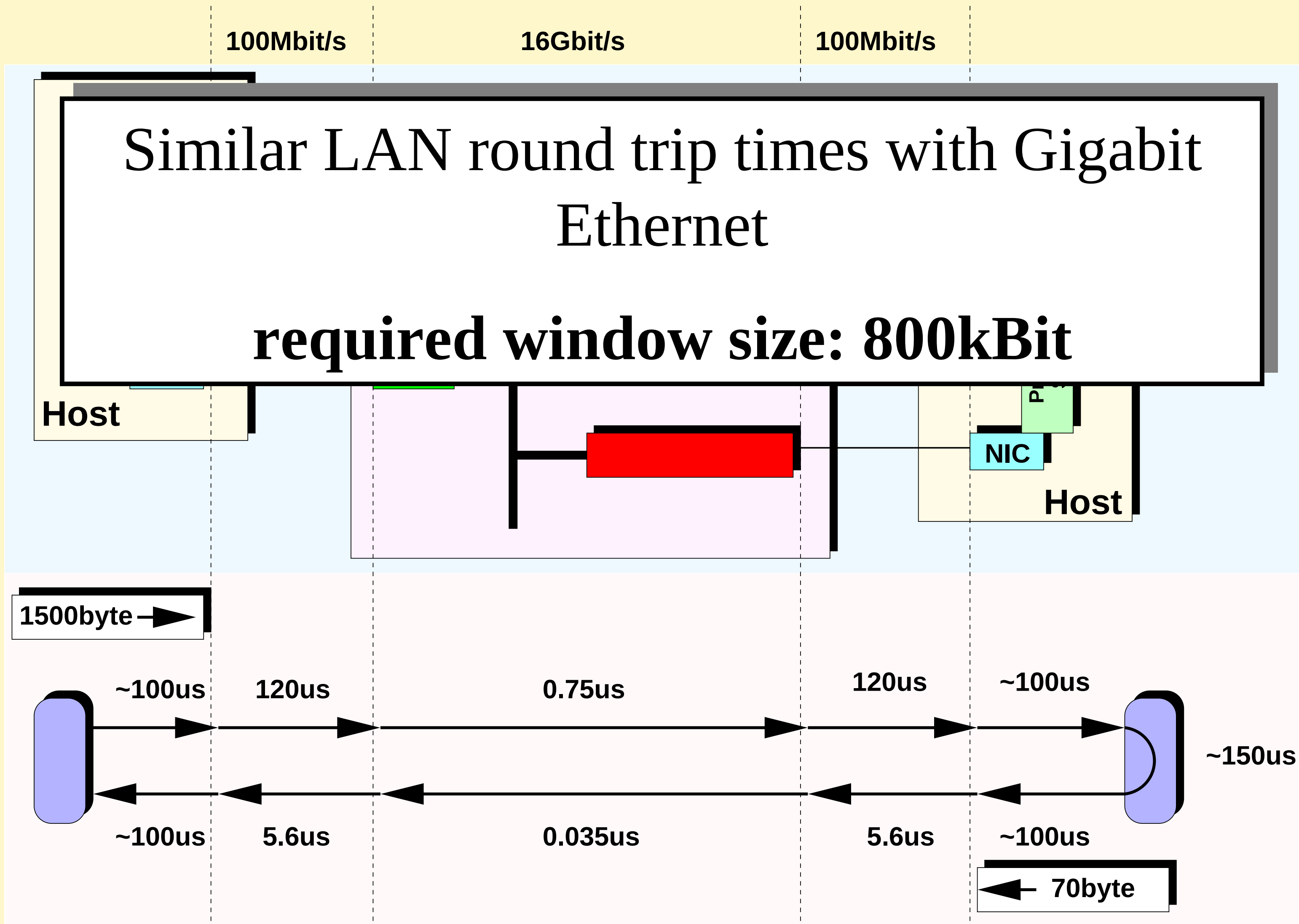

**WS >= BW * RTT**

# Real−World Round Trip Times − The Bad News




**Total RTT: ca 800us Window >= 800us * 100Mbit/s = 80kBit**

# Real−World Round Trip Times − The Bad News




Similar LAN round trip times with Gigabit Ethernet

**required window size: 800kBit**

**Total RTT: ca 800us      Window >= 800us * 100Mbit/s = 80kBit**

# LAN: Round Trip Times

```
luitz@bbt-odf100 9:42am [~] ping -s 1500 bbt-srv00 -c 5
PING bbt-srv00 (134.79.108.26) from 134.79.111.56 : 1500(1528) bytes of data.
1508 bytes from bbt-srv00 (134.79.108.26): icmp_seq=0 ttl=255 time=725 usec
1508 bytes from bbt-srv00 (134.79.108.26): icmp_seq=1 ttl=255 time=505 usec
1508 bytes from bbt-srv00 (134.79.108.26): icmp_seq=2 ttl=255 time=489 usec
1508 bytes from bbt-srv00 (134.79.108.26): icmp_seq=3 ttl=255 time=668 usec
1508 bytes from bbt-srv00 (134.79.108.26): icmp_seq=4 ttl=255 time=481 usec
--- bbt-srv00 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.481/0.573/0.725/0.105 ms
```

Both hosts on Gigabit Ethernet, same network, same switch

Large jitter, long RTT (fast, remote side kernel-only)

# Protecting Intermediate Buffers From Overflowing

- RTT dominated by protocol stack
  - 100 MBit/s example
    - 20 senders 8 kByte window each, one destination: 160 kByte of data in transit
- → Make protocol stack and application response faster/deterministic (e.g. user-space protocol and network driver implementation)
- → Increase buffer drain rates or sizes
- → Use more sophisticated protocol

# What´s Wrong Here?

- Send UDP data through 1-Gbit/s interface

```
Luitz@bbt-odf100$ ./ttcp -t -u -l30000 -n10000 bbt-srv100
ttcp-t: buflen=30000,nbuf=10000,align=16384/+0,port=5001 udp->bbt-srv100
ttcp-t: socket
ttcp-t: 300000000 bytes in 0.90 real seconds = 326766.54 KB/sec +++
ttcp-t: 10006 I/O calls, msec/call=0.09, calls/sec=11160.32
ttcp-t: 0.0user 0.8sys 0:00 real 98% 0i+0d 0maxrss 0+7pf 0+0csw
```

# What´s Wrong Here?

- Send UDP data through 1-Gbit/s interface

```
Luitz@bbt-odf100$ ./ttcp -t -u -l30000 -n10000 bbt-srv100
ttcp-t: buflen=30000,nbuf=10000,align=16384/+0,port=5001 udp->bbt-srv100
ttcp-t: socket
ttcp-t: 300000000 bytes in 0.90 real seconds = 326766.54 KB/sec +++
ttcp-t: 10006 I/O calls, msec/call=0.09, calls/sec=11160.32
0+0csw
```

326,766 KByte/s !!!

(1Gbit/s = 125MByte/s max)

The protocol stack is dropping packets!!!

# UDP Protocol Stack Complications

- "Slow" in terms of LAN RTTs
- Timing jitter
- No hard timing guarantees
  - Data can get batched up while CPU is busy otherwise (this is a "legal" optimization)
- "Lossy"
  - Data may be dropped silently before even reaching the physical interface ("best-effort")
- But it's low-overhead and nice and simple!
  - Allows broadcast and multicast

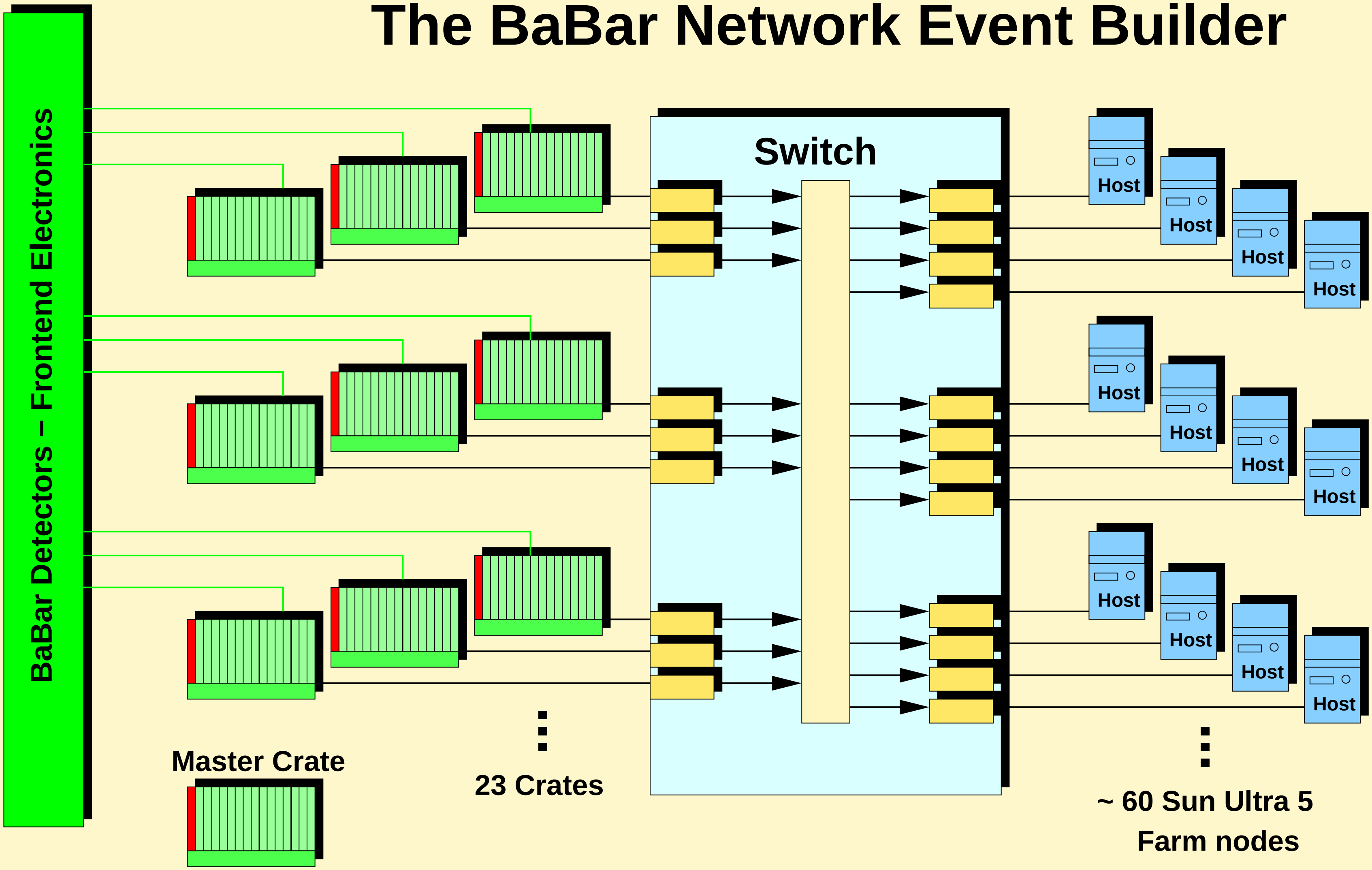
# TCP – Transmission Control Protocol (www, email, ssh, ...)

- Reliable end-to-end transfer
- Timeout and retransmit
- RTT estimate
- Negotiation of maximum window sizes and other parameters
- Dynamic flow control algorithm uses packet loss information as feedback
  - Slow start
  - Congestion avoidance

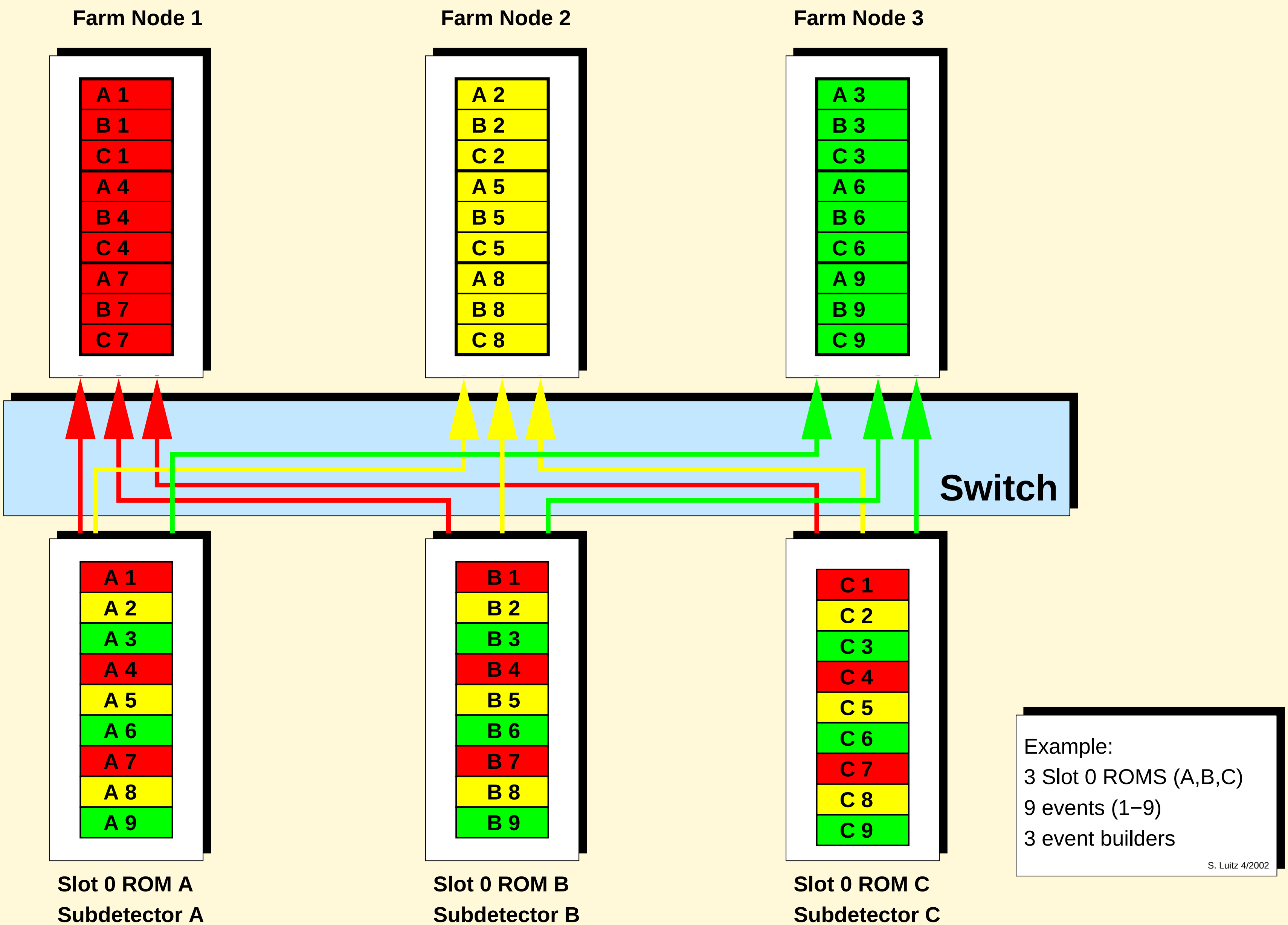
# Event Building

- "Event"
  - Snapshot of the detector at a certain time
  - Contributions from multiple sub-detectors
- Event Building
  - Combine the various event contributions into a complete event
  - Distribute to trigger farm
- Data re-ordering problem

# The BaBar Network Event Builder



# Event Building in a Switched Network



# The BaBar Event Builder (1)

- 23 100MBit/s sources
  - “Slot-0-ROMs”
  - VME single board computers with VxWorks
  - Can generate peak rate of 2.3 Gbit/s
  - Highly synchronized by “Trigger System”
- 60 100MBit/s destinations
  - “Farm Nodes”
  - Sun Ultra-5 (333MHz)
  - Could absorb 6 Gbit/s

# The BaBar Event Builder (2)

- UDP based transport
  - Simple ("send and forget")
  - Naturally non-blocking
  - Scalable (no connections)
  - Possible to optimize (e.g. multicast)
  - Simple failure modes – doesn't hide problems
- Simple flow control
- No retransmission
  - Lost packet → incomplete event(s)

# The BaBar Event Builder (3)

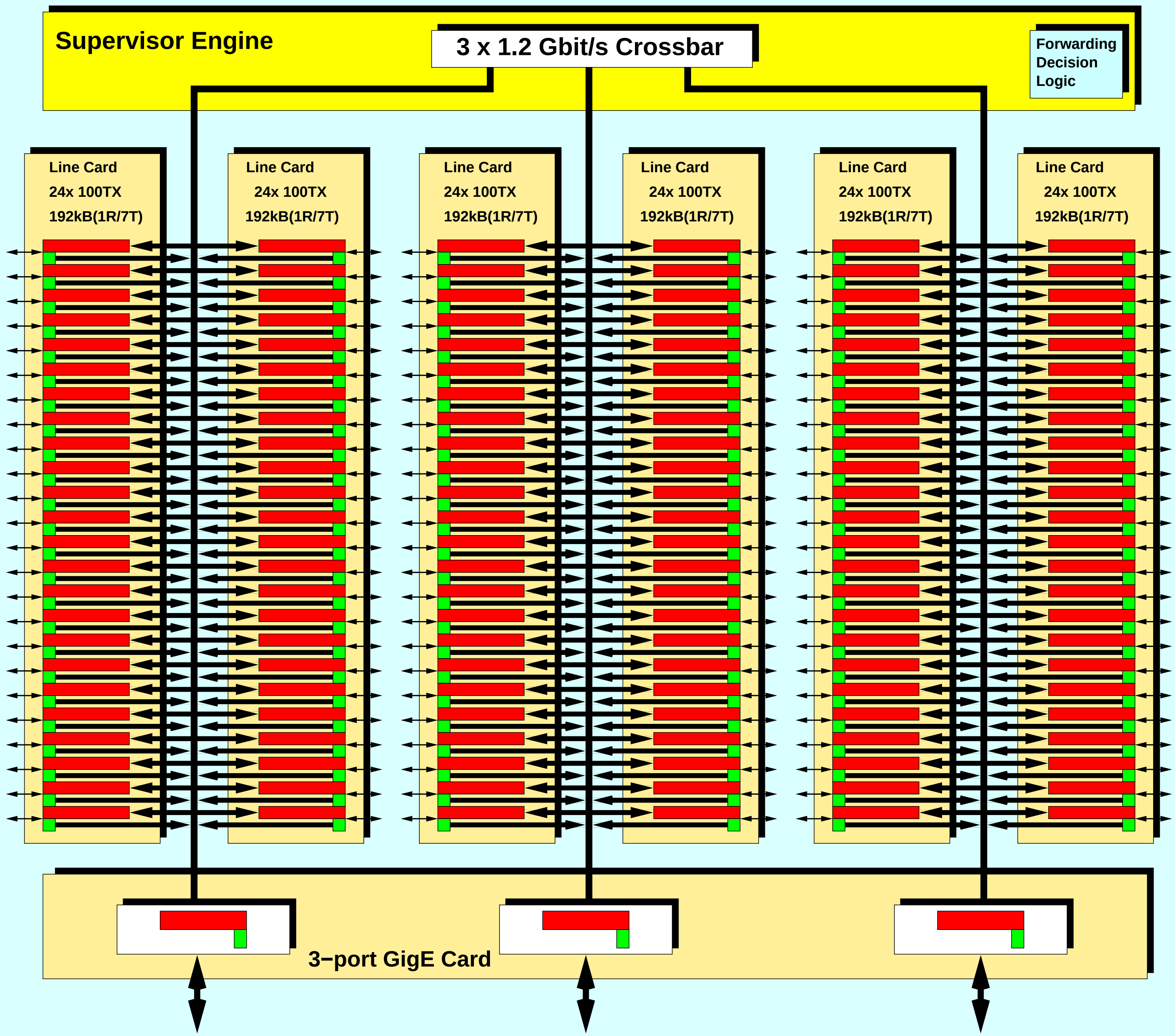
- All buffers in the system must be large enough to not overflow at peak data rate
- Average event: ~30 kByte
  - Largest events >100kByte
  - Trigger rate spikes
  - Two subsequent events can be sent to same farm node
- Output buffer sizes >> 100kByte

# 1999: Cisco Catalyst 5500

- First BaBar event building switch
  - "3.6 Gbit/s" capacity
  - 10 usable slots
  - 24-port 10/100MBit/s Ethernet line cards
    - 192kByte per-port buffer (split into 24kByte for receiving and 168kByte for transmitting)
  - 3-port 1Gbit/s Ethernet line cards

# Problems with Catalyst 5500

- Capacity seemed lower than the 3.6 GBit/s naively expected
- Unaccounted packet loss
- → Start to investigate
  - Re-read promotional material
  - Look at hardware
  - Test & measure
    - Don’t have special test equipment (e.g. traffic generators)
  - Ask Cisco …

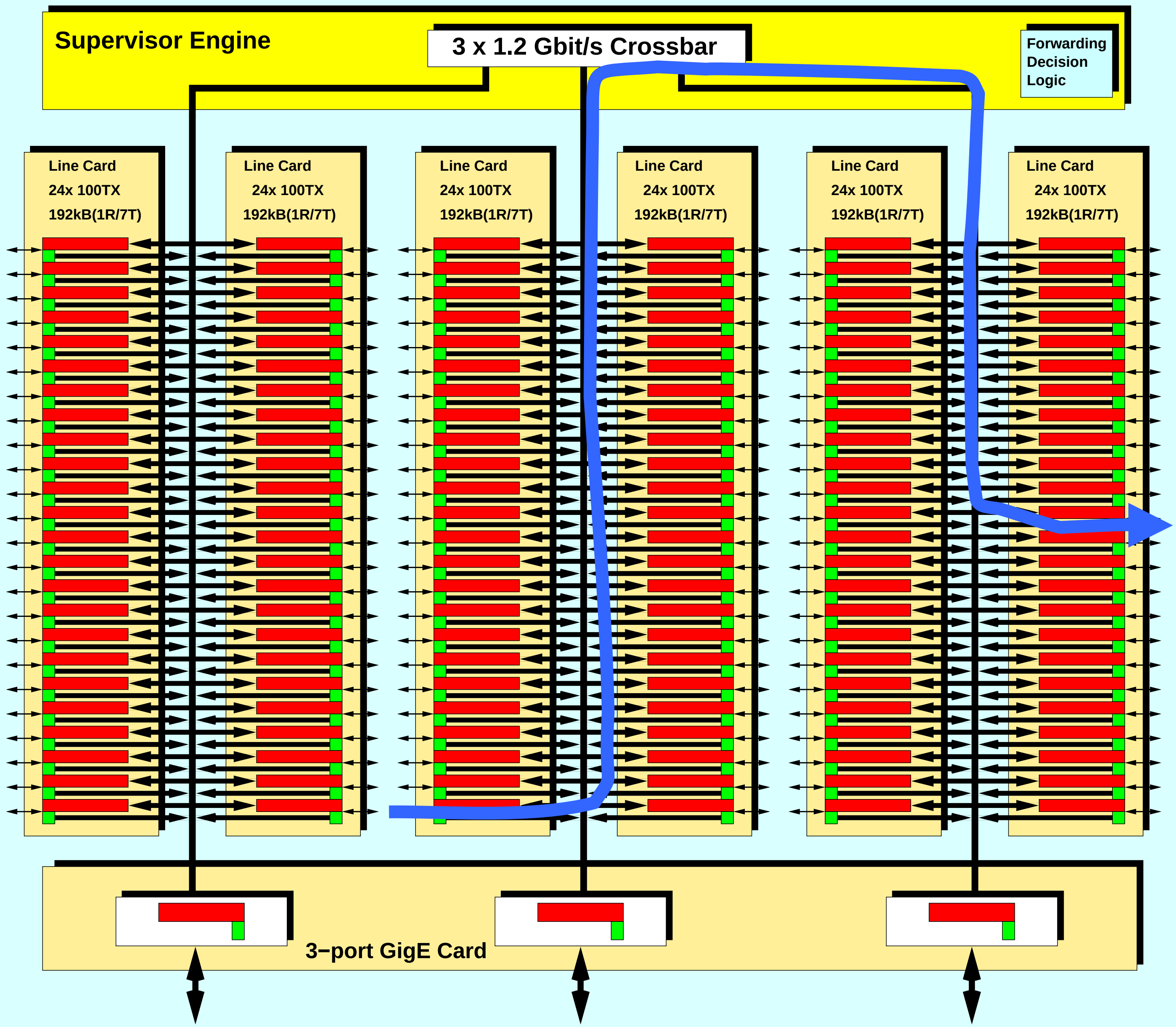



# Cisco Catalyst 5500

**3 1.2 Gbit/s backplanes connected by "crossbar"**

**10 slots assigned to 3 backplanes 4−3−3**

**192 kByte Port Buffers 168:24kByte tx:rx**

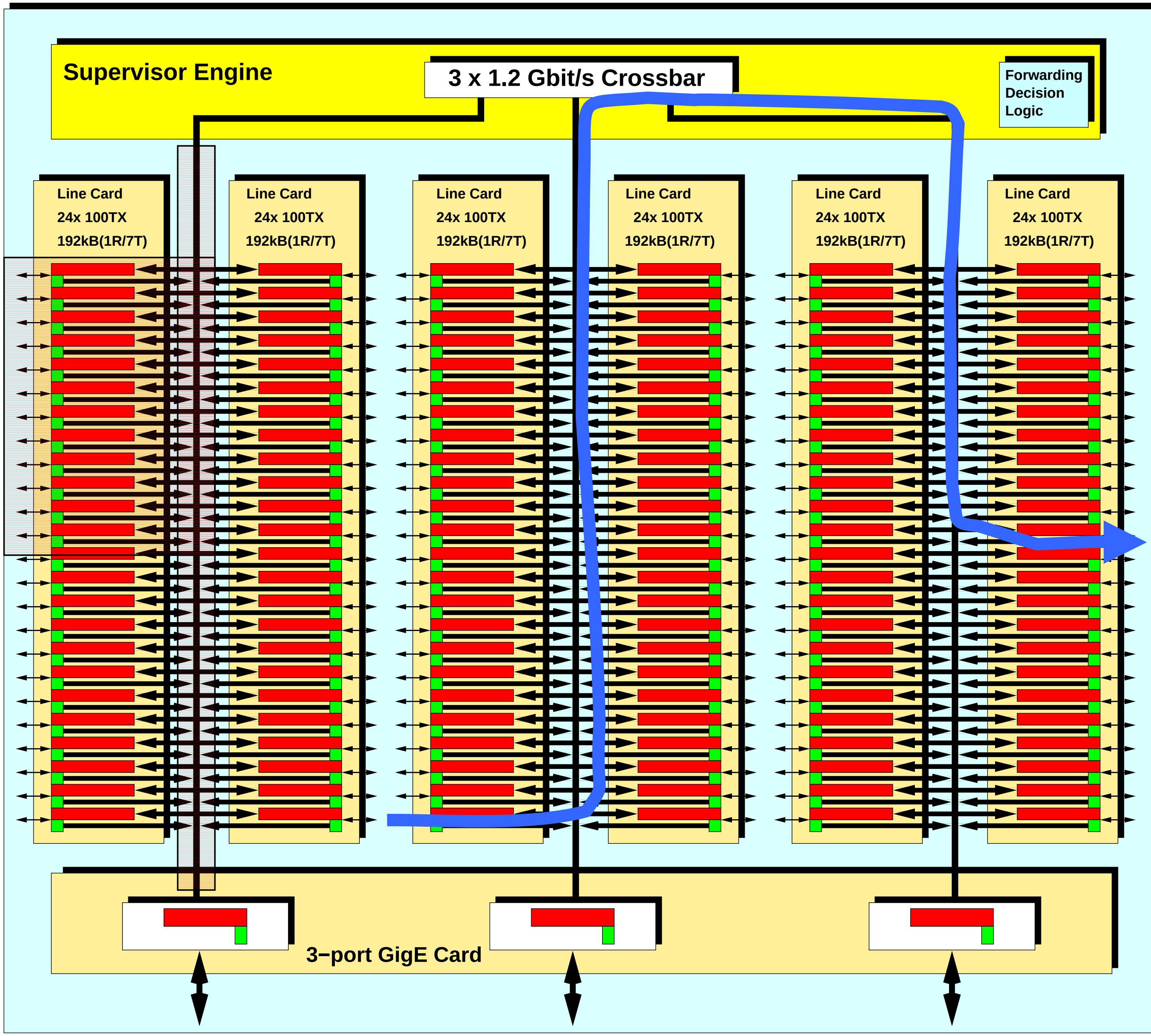



# Cisco Catalyst 5500

3 1.2 Gbit/s backplanes connected by "crossbar"

10 slots assigned to 3 backplanes 4−3−3

192 kByte Port Buffers 168:24kByte tx:rx

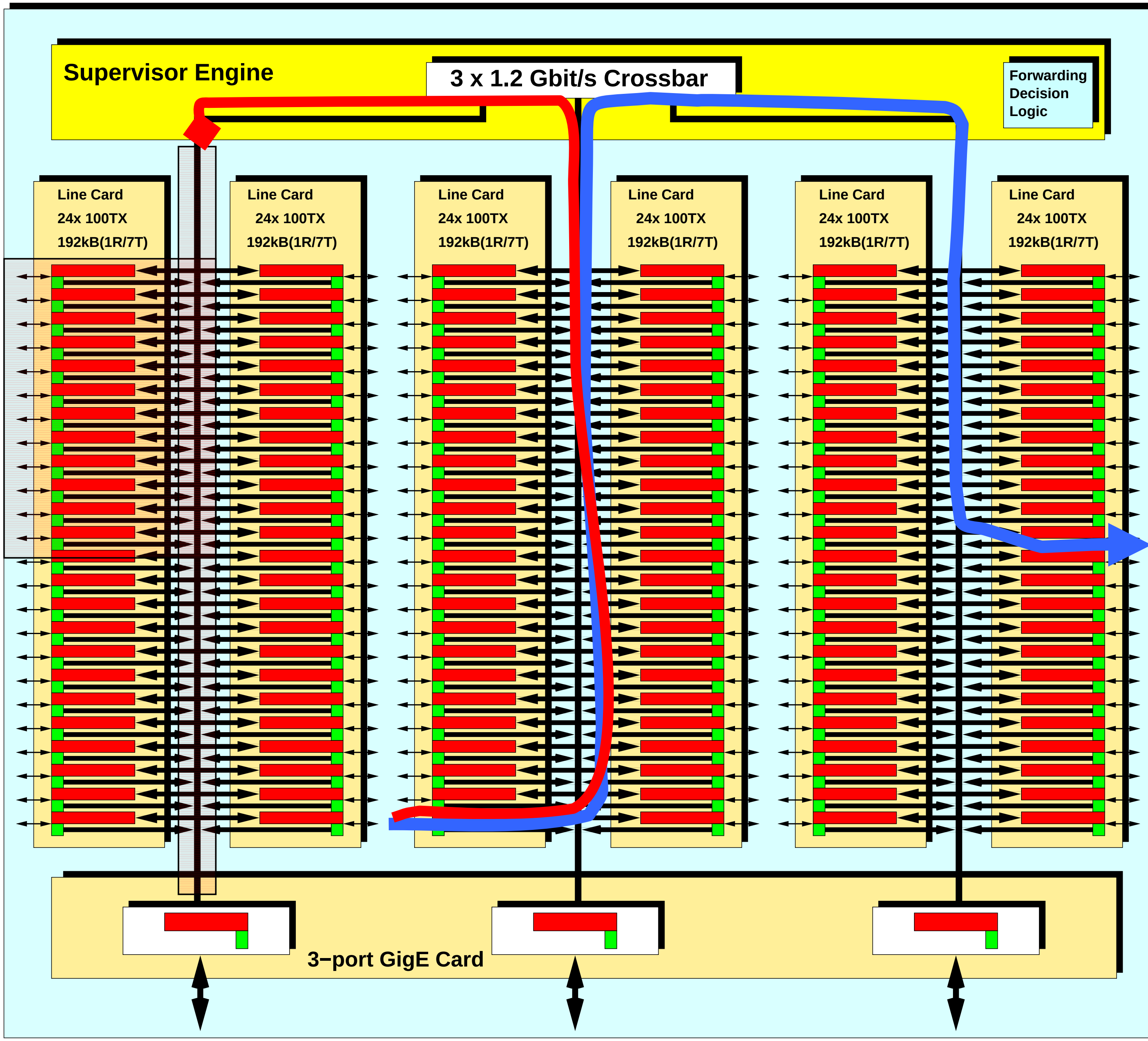



# Cisco Catalyst 5500

3 1.2 Gbit/s backplanes connected by "crossbar"

10 slots assigned to 3 backplanes 4−3−3

192 kByte Port Buffers 168:24kByte tx:rx




# Cisco Catalyst 5500

**3 1.2 Gbit/s backplanes connected by "crossbar"**

**10 slots assigned to 3 backplanes 4−3−3**

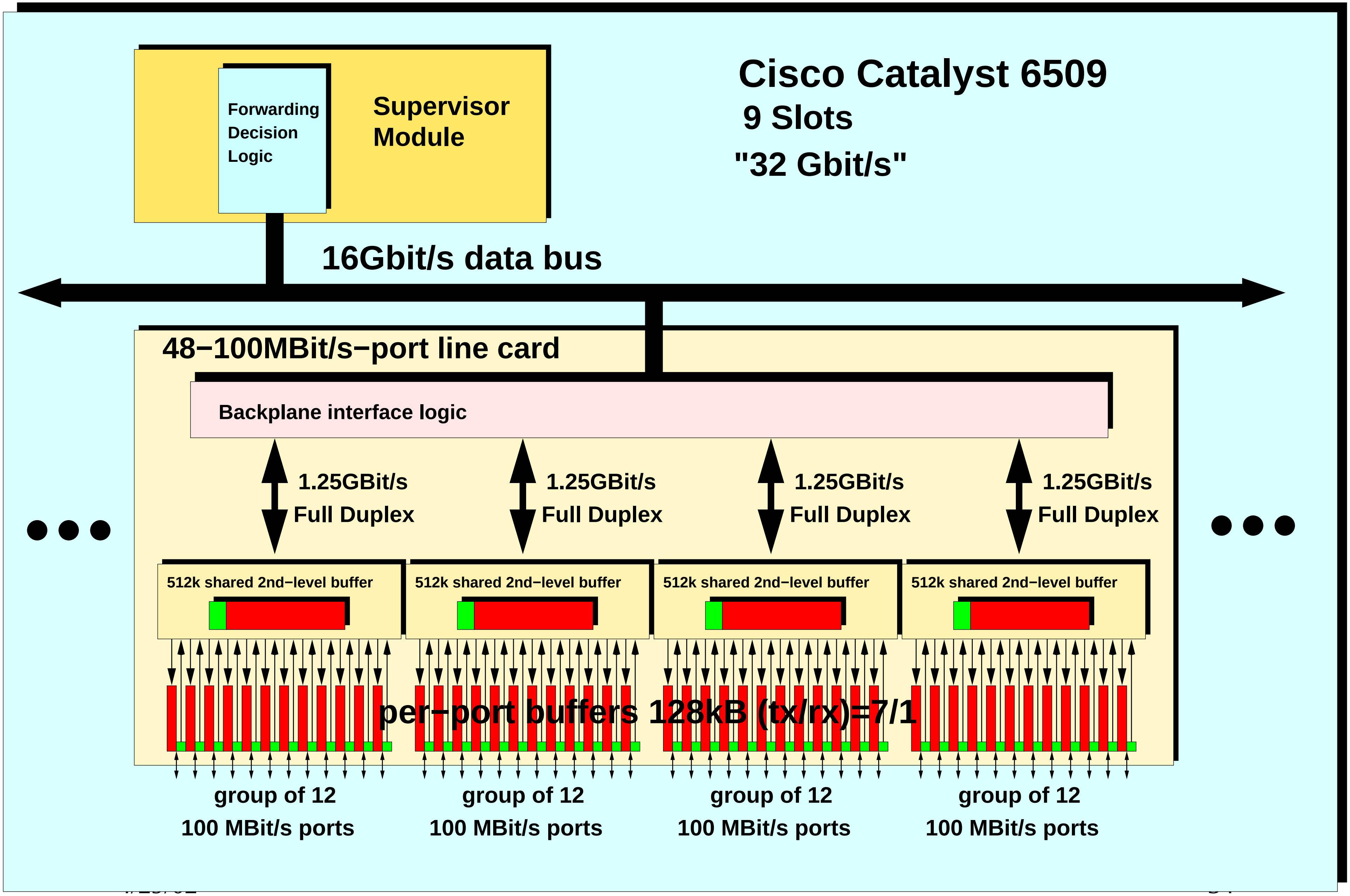
**192 kByte Port Buffers 168:24kByte tx:rx**

# Conclusions

- Catalyst 5500 is not a true 3.6GBit/s switch
    - three interconnected 1.2Gbit/s switches
    - Maximum event building peak capacity: 1.2 Gbit/s
    - Nasty: Input bandwidth (2.3 Gbit/s) exceeded switching bandwidth (1.2 Gbit/s). Overflow small input buffers
- Optimization
  – Rewire so that the slot-0 ROMs and farm nodes are on one backplane (keep backplane clean). Originally wired to distribute event building traffic over all backplanes
  – Improved flow control
- Replace with faster switch … introducing the …

# Cisco Catalyst 6500

- Next generation switch: Catalyst 6500
  - Asked Cisco for detailed specs
  - → really useful „whitepaper“
    - “32 GBit/s” bandwidth
      - Well actually … 16GBit/s backplane – marketing adds up input and output to get a larger number – it’s the “industry standard” to confuse switch bandwidth with network bandwidth
    - 128kB per-port buffers + 512kB shared between 12 ports on 100MBit/s line card
    - 512kB per-port buffers on Gigabit Ethernet line card


Cisco Catalyst 6509
9 Slots
"32 Gbit/s"
Forwarding Decision Logic
Supervisor Module
16Gbit/s data bus
48−100MBit/s−port line card
Backplane interface logic
1.25GBit/s Full Duplex
1.25GBit/s Full Duplex
1.25GBit/s Full Duplex
1.25GBit/s Full Duplex
512k shared 2nd−level buffer
512k shared 2nd−level buffer
512k shared 2nd−level buffer
512k shared 2nd−level buffer
per−port buffers 128kB (tx/rx)=7/1
group of 12 100 MBit/s ports
group of 12 100 MBit/s ports
group of 12 100 MBit/s ports
group of 12 100 MBit/s ports

# „Trouble with Cat 6500“

- During Farm/L3 trigger load test
  - Massive dead time due to packet loss
  - System under-buffered
- Investigate
  - Read whitepaper again
  - Take into account existence of a “hidden” flow control parameter. SCS problem with ´high packet loss in Gigabit→ 10/100Mbit NFS traffic’ (big UDP datagrams)
  - … and measure the switch buffer sizes

# Measuring Buffer Sizes – Basic Idea

Data in Buffer

f : buffer fill rate

d: buffer drain rate

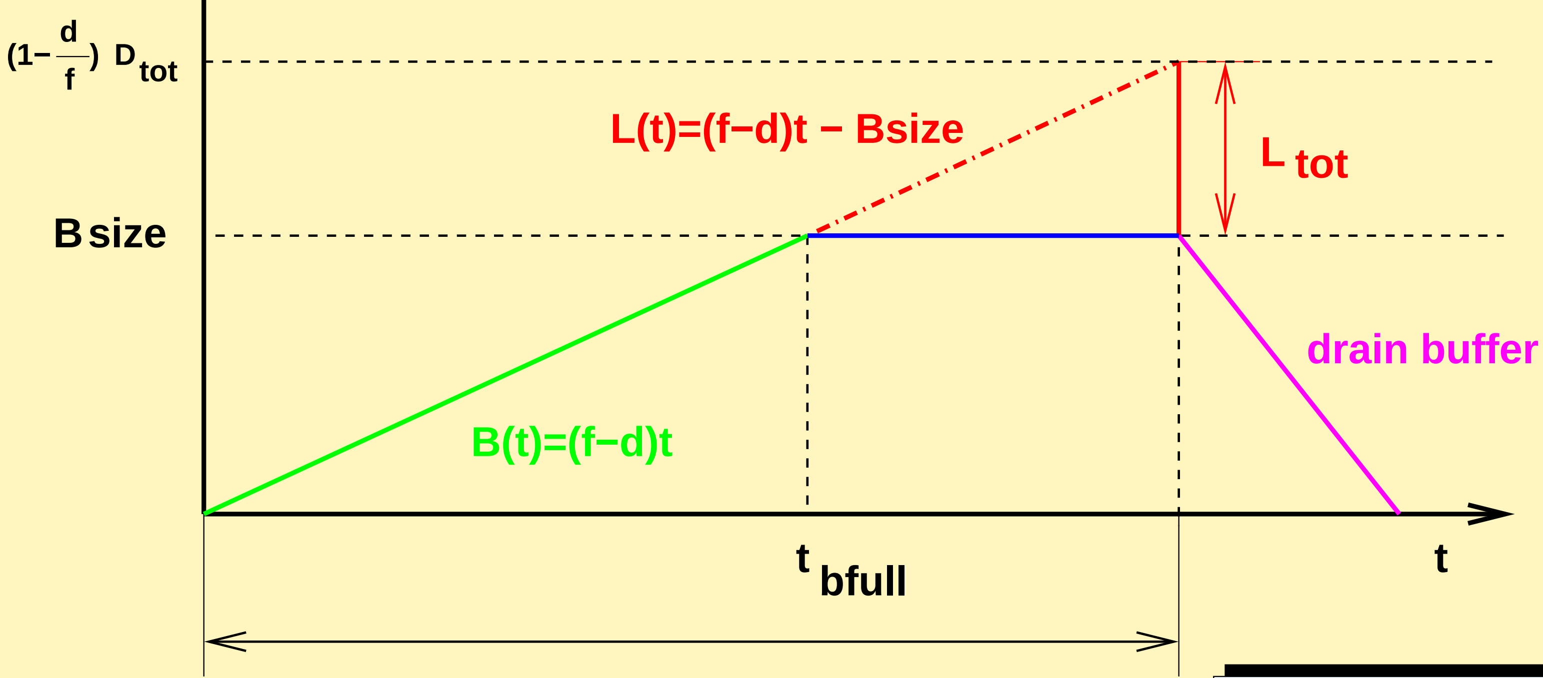
$t_{send}$ : time to send $D_{tot}$ bytes of data

L : data lost

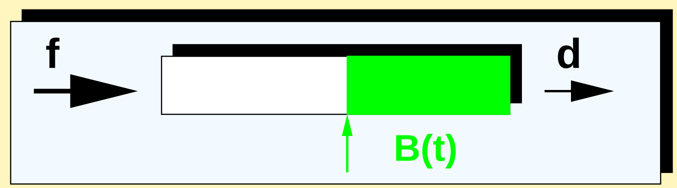
B : data in buffer

Bsize : buffer size

$$L_{tot} = \left(1 - \frac{d}{f}\right) D_{tot} - Bsize$$

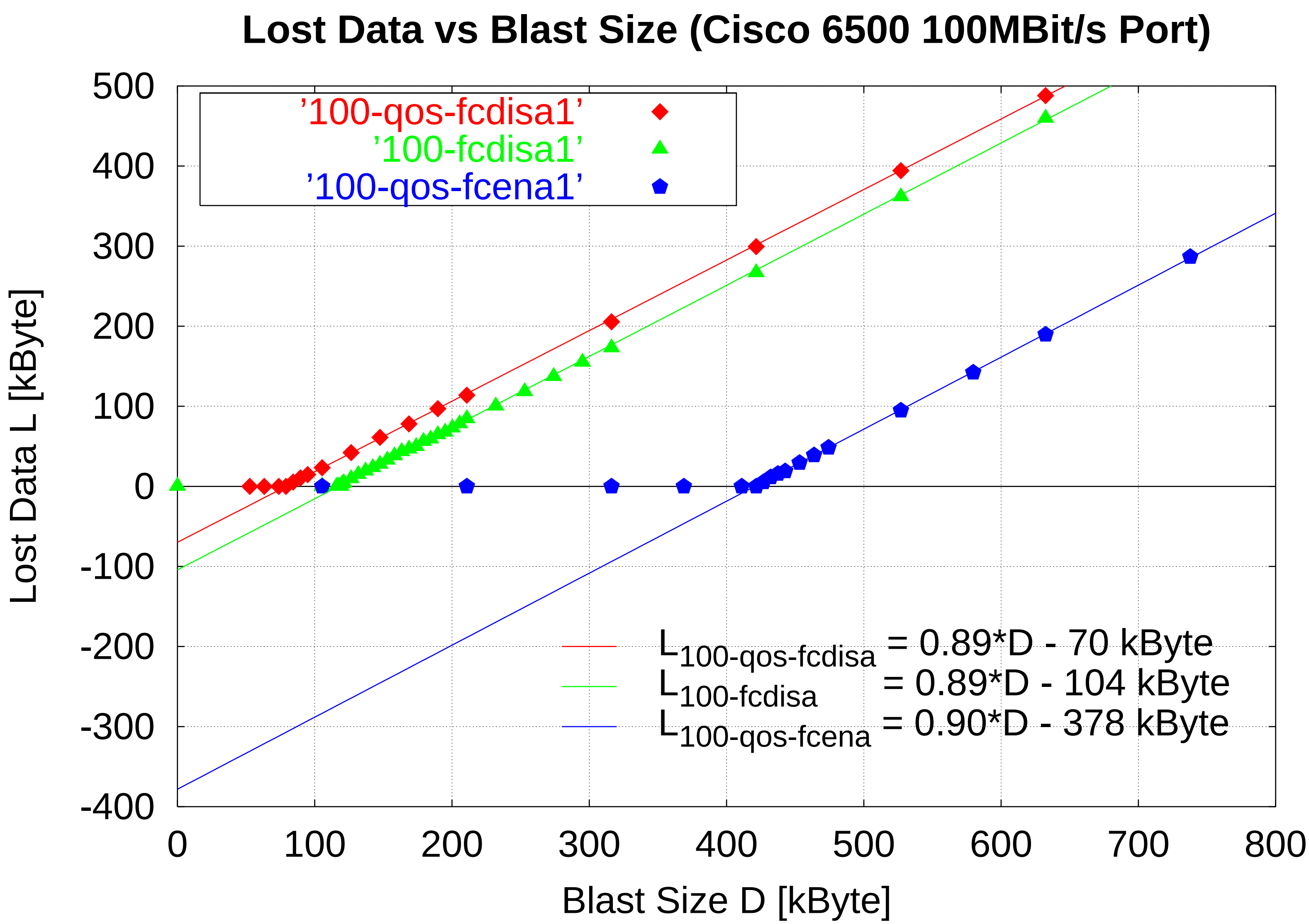


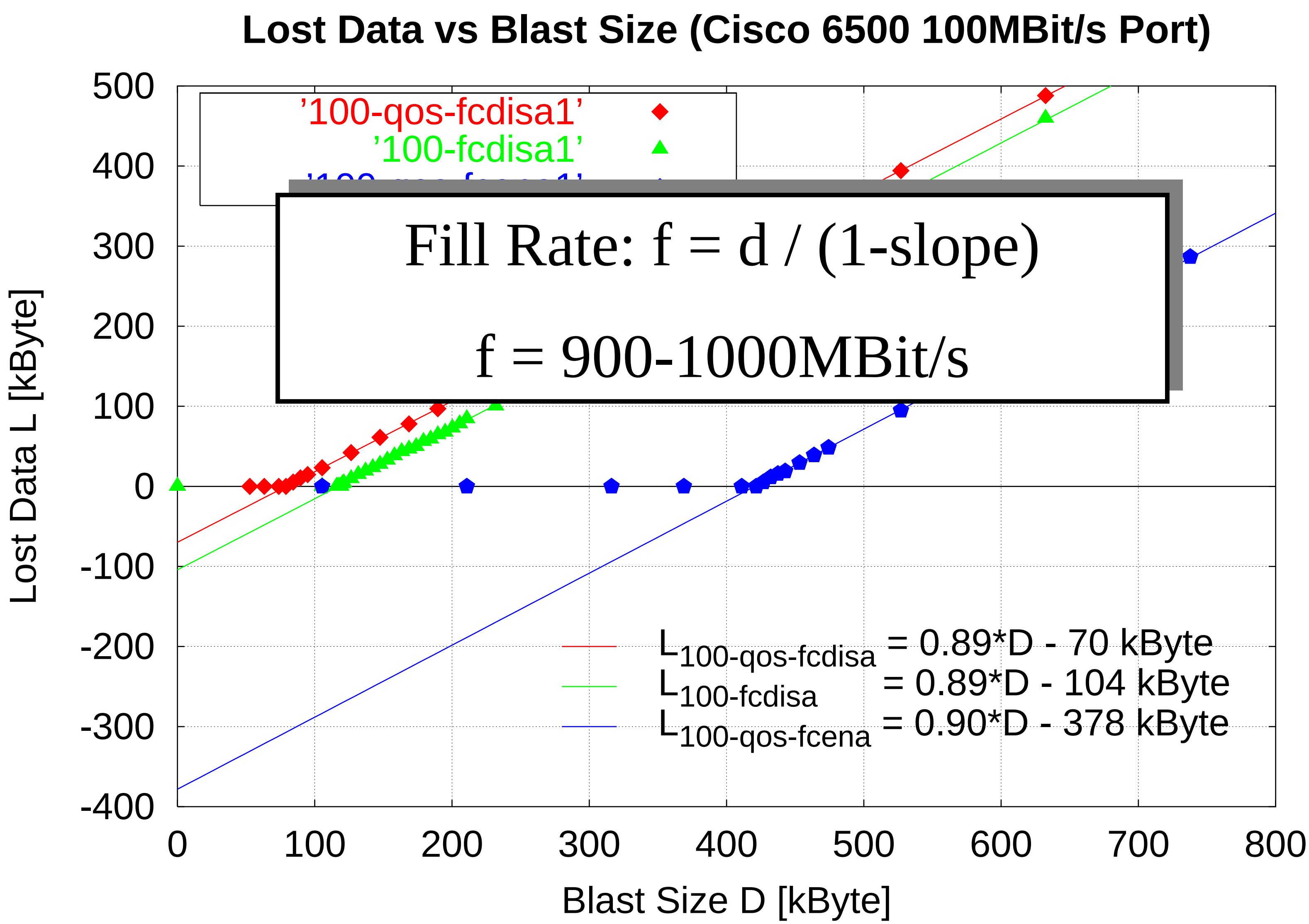

$$t_{send} = \frac{D_{tot}}{f}$$



# Measuring the Buffer Size

- Source: 1GBit/s (Linux machine with Gigabit NIC) blasting a 100MBit/s port
- Using switch dropped packet counters to measure amount of data lost
- “Flow Control” parameter: disabled/enabled
- QoS (Quality of Service) disabled/enabled
  - Enabling QoS reduces buffer sizes available for normal traffic to ~ 80% by reserving buffer space for high priority traffic
  - Has to be on to read dropped packet counters when “Flow Control” turned on


Lost Data vs Blast Size (Cisco 6500 100MBit/s Port)
'100-qos-fcdisa1'
'100-fcdisa1'
'100-qos-fcena1'
$L_{100\text{-}qos\text{-}fcdisa}$ = 0.89*D - 70 kByte
$L_{100\text{-}fcdisa}$ = 0.89*D - 104 kByte
$L_{100\text{-}qos\text{-}fcena}$ = 0.90*D - 378 kByte
Lost Data L [kByte]
Blast Size D [kByte]
500
400
300
200
100
0
-100
-200
-300
-400
0
100
200
300
400
500
600
700
800




Fill Rate: f = d / (1-slope)

f = 900-1000MBit/s

# Conclusion

- Default: 512k buffer disabled
- When enabled switch performs very well
  - ~2 * 10E-6 damaged events due to network losses
  - 30 kByte Event ca. 20 Ethernet packets
  - ~ 1 per 10 million packets lost in the network
  - ~ 250 out of 2.5 billion packets per day
- … but the next steps are …
  - Gigabit Ethernet for the farm (Summer 02)
  - Gigabit Ethernet for the ROMs (later)

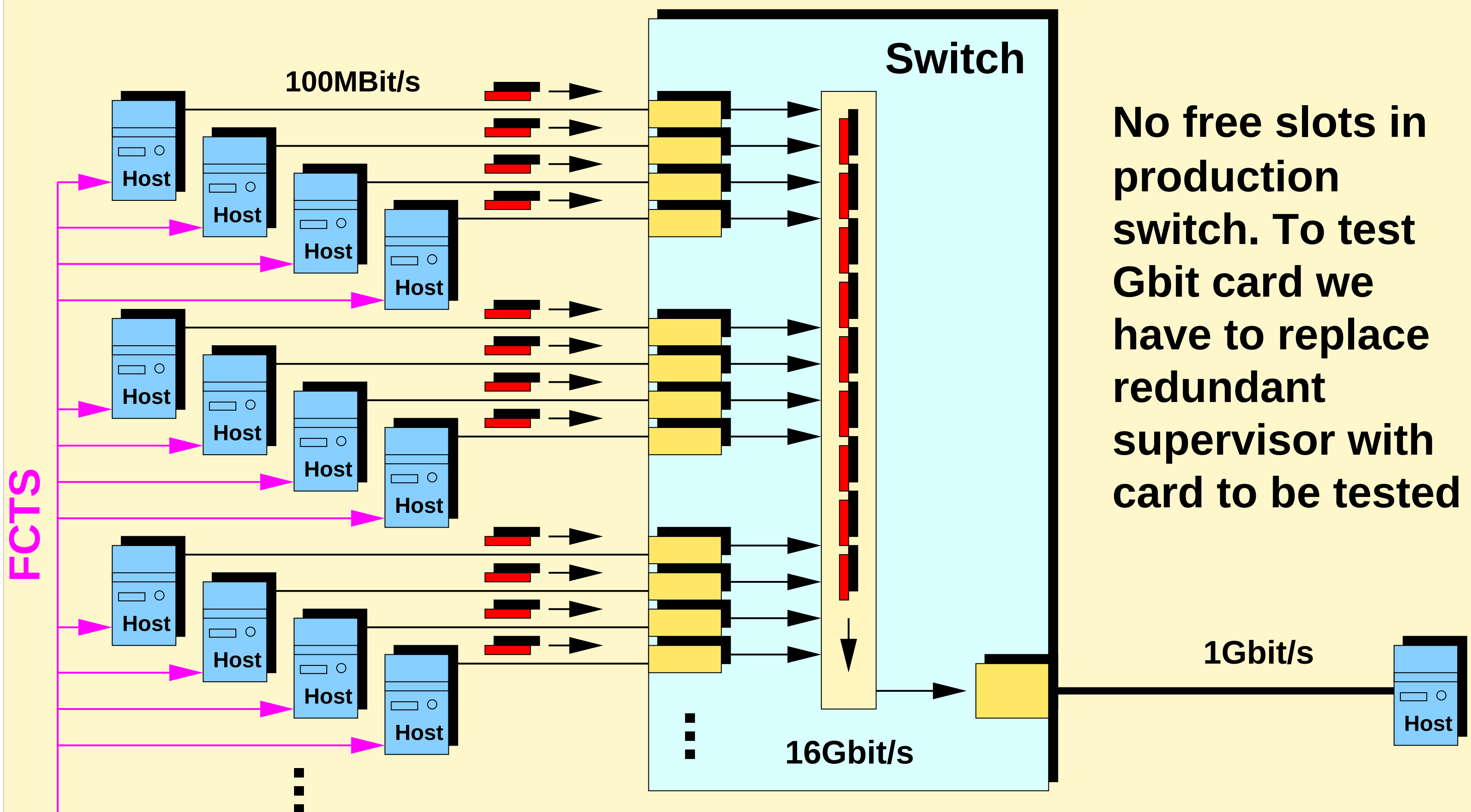
# The Gigabit Future (1)

- Linux Farm Upgrade: Gigabit Ethernet
  - More intelligent cards
    - Lower CPU overhead
  - Higher switch buffer drain rates
- Data Flow ROM Gigabit Ethernet Upgrade
  - More intelligent cards
  - Higher output bandwidth
  - Drive the cards directly (no OS)
    - Probably wouldn’t have been possible with TCP
    - Lower CPU overhead
    - Better control (e.g. traffic shaping)

# The Gigabit Future (2)

- Farm upgrade – no problems expected
  - Academic question: Can we measure the switch gigabit card buffer sizes?
  - For the above technique we need a source faster than 1GBit/s
  - … we tried a few …

# The Big Packet Source: BaBar Data Flow

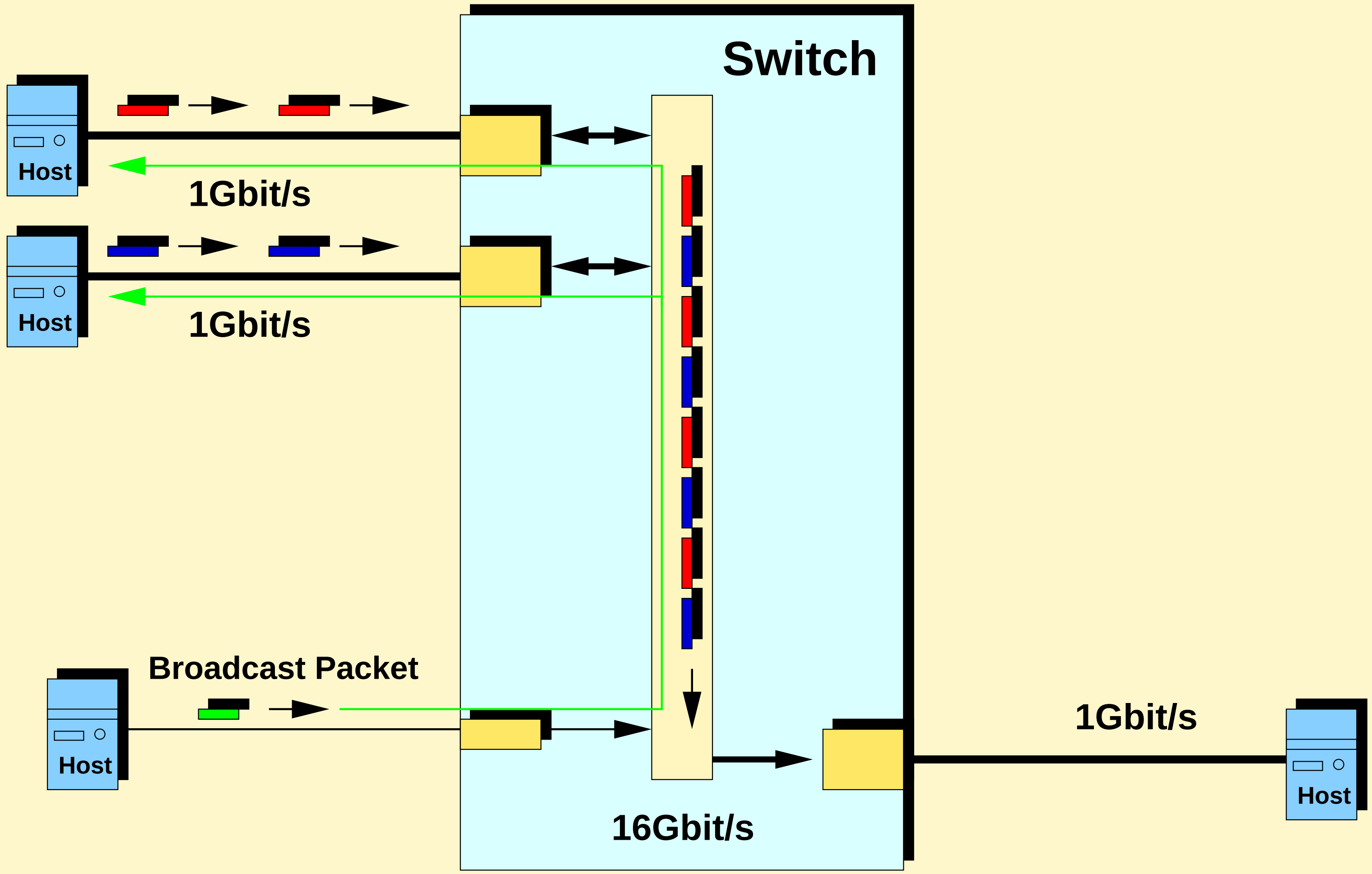



No free slots in production switch. To test Gbit card we have to replace redundant supervisor with card to be tested

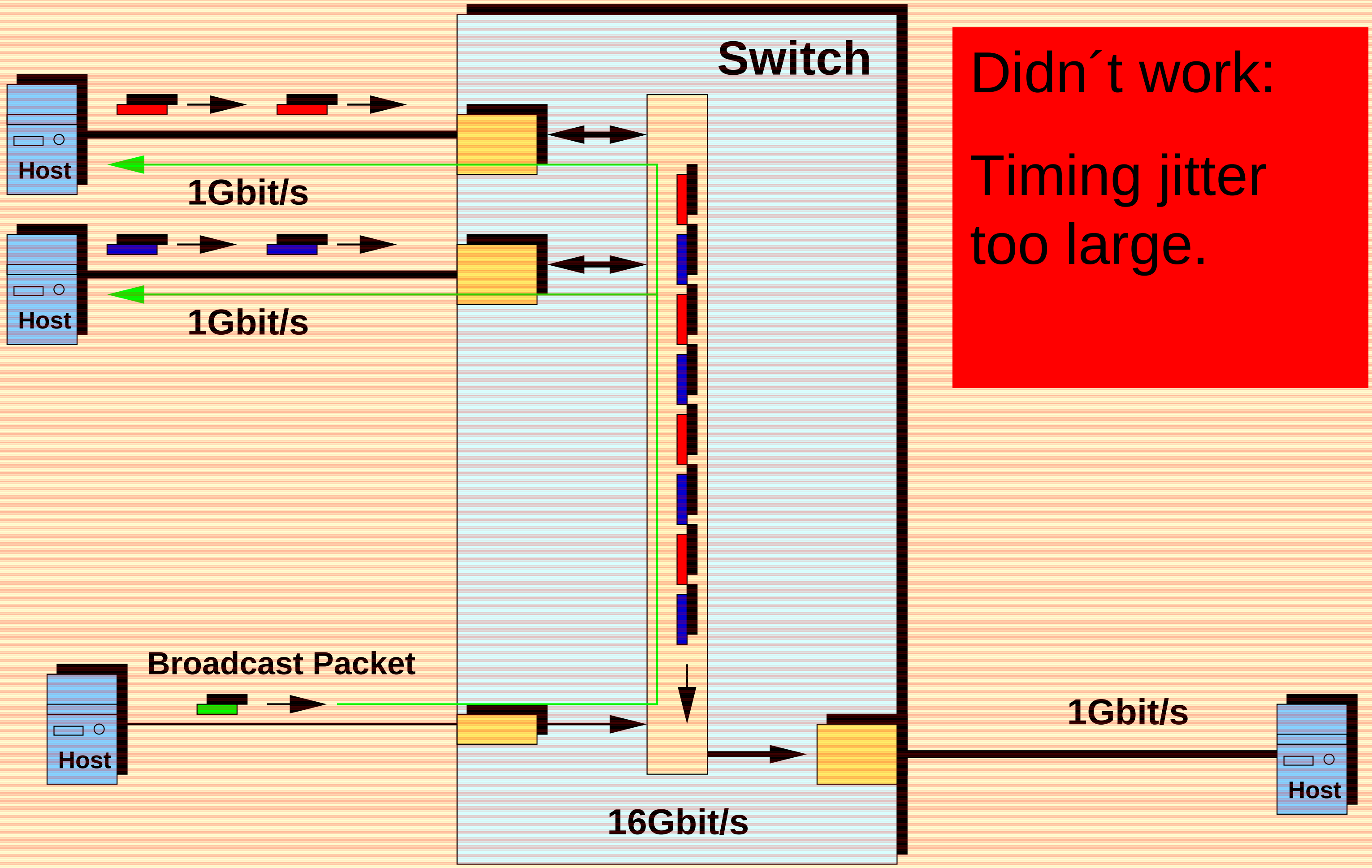
**>150 SBCs**

**>15 Gbit/s Raw Source Bandwidth**

# Using 2 Hosts with Gigabit Ethernet as Source



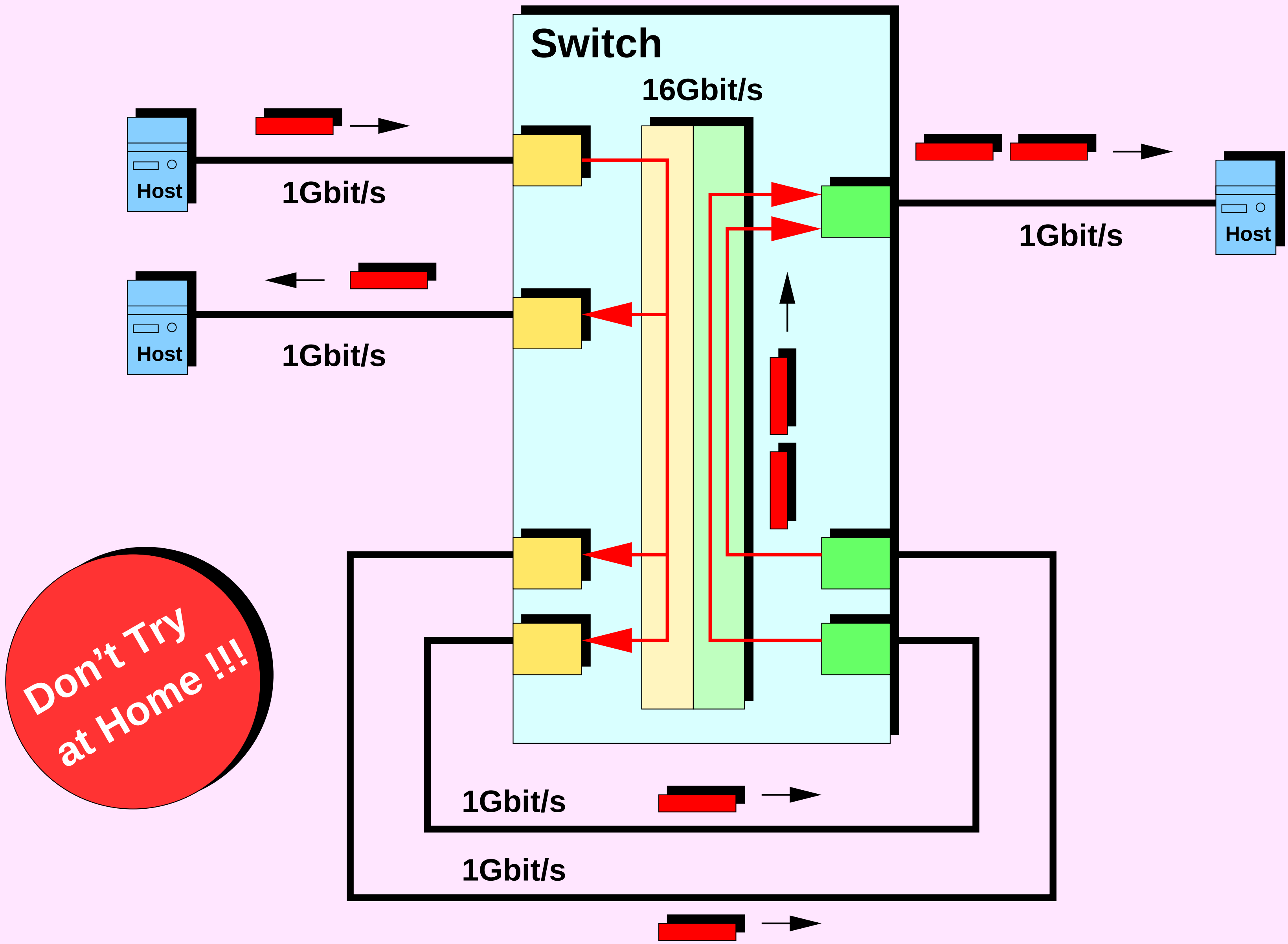
# Using 2 Hosts with Gigabit Ethernet as Source

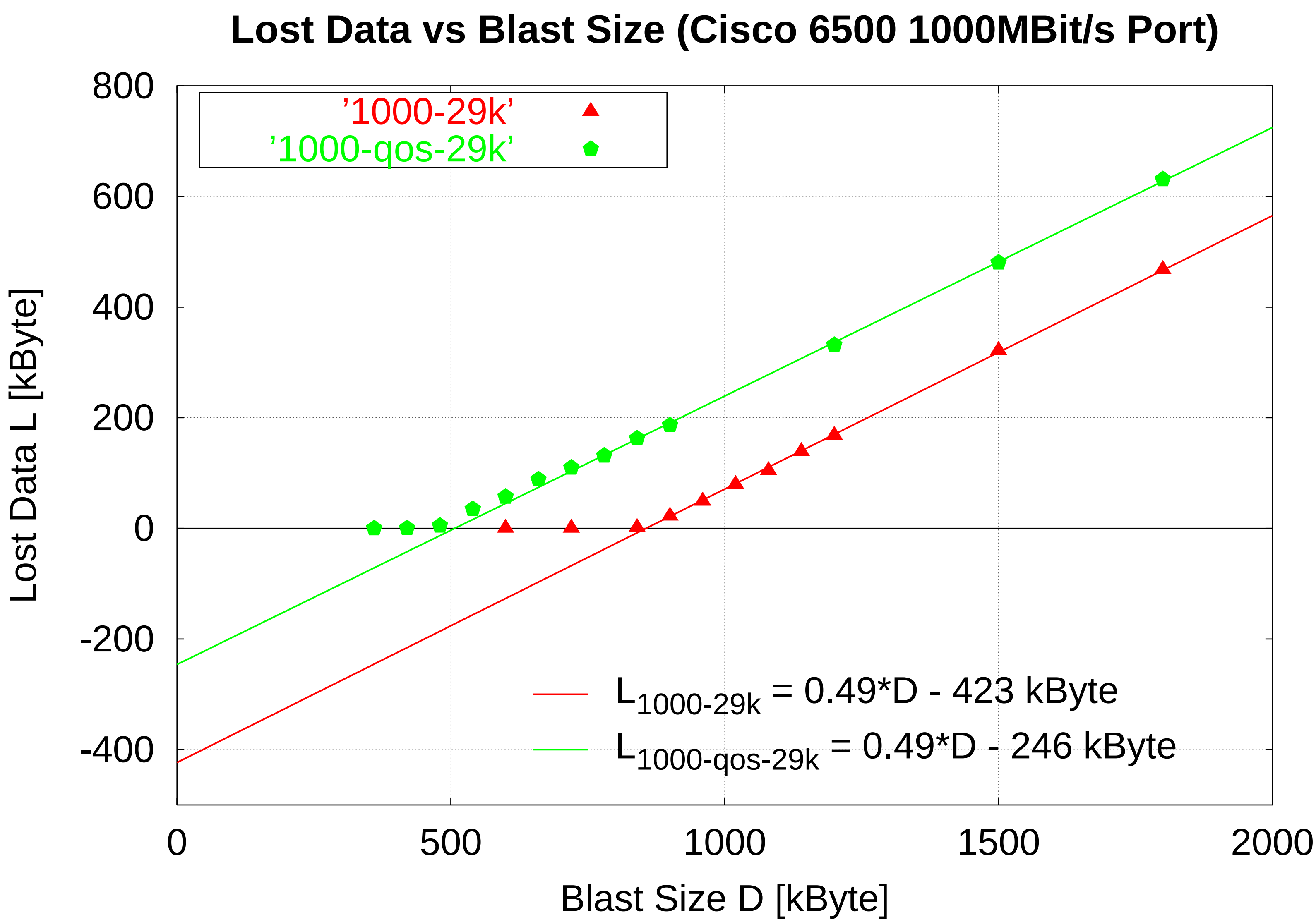



Didn´t work:

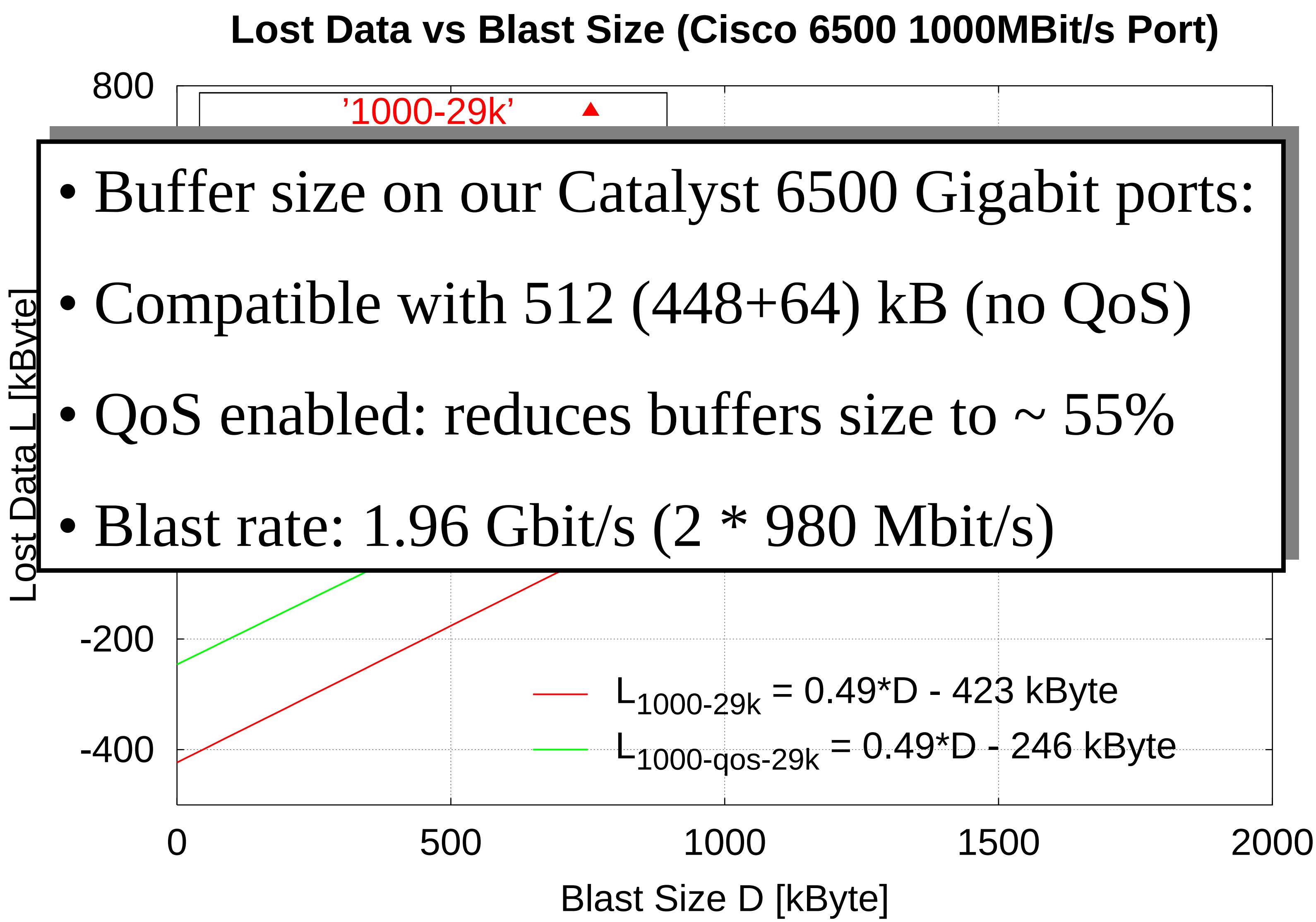
Timing jitter too large.

# Measuring Buffer Sizes – Packet Duplicator





Lost Data vs Blast Size (Cisco 6500 1000MBit/s Port)
'1000-29k'
'1000-qos-29k'
$L_{1000-29k}$ = 0.49*D - 423 kByte
$L_{1000-qos-29k}$ = 0.49*D - 246 kByte
Lost Data L [kByte]
Blast Size D [kByte]
800
600
400
200
0
-200
-400
0
500
1000
1500
2000




- Buffer size on our Catalyst 6500 Gigabit ports:
- Compatible with 512 (448+64) kB (no QoS)
- QoS enabled: reduces buffers size to ~ 55%
- Blast rate: 1.96 Gbit/s (2 * 980 Mbit/s)

# Can We Measure More Things Without Special Test Equipment?

- Backplane speeds?
- Pipelining buffer effects?
- Host NIC send/receive buffering?
- Gigabit Ethernet flow control?
- … ?

Some interesting projects!

Better understanding of our equipment.

# The Gigabit Future (3)

- Data Flow upgrade
  - Gigabit interfaces on Slot-0-ROMs
  - 5 more Slot-0-ROMs (split load)
- “Back” to the Catalyst 5500 situation
  - ~ 29 * 1GBit/s into 16GBit/s switch backplane
  - Gigabit Ethernet defines a NIC-to-NIC flow control protocol, will this work and help?
  - Can we do traffic shaping (high-resolution source flow control?)
- If all this doesn’t work …

We’ll get the next generation switch (256GBit/s) and have the chance to do more exciting and fun experiments to understand and work around its (mis-)features