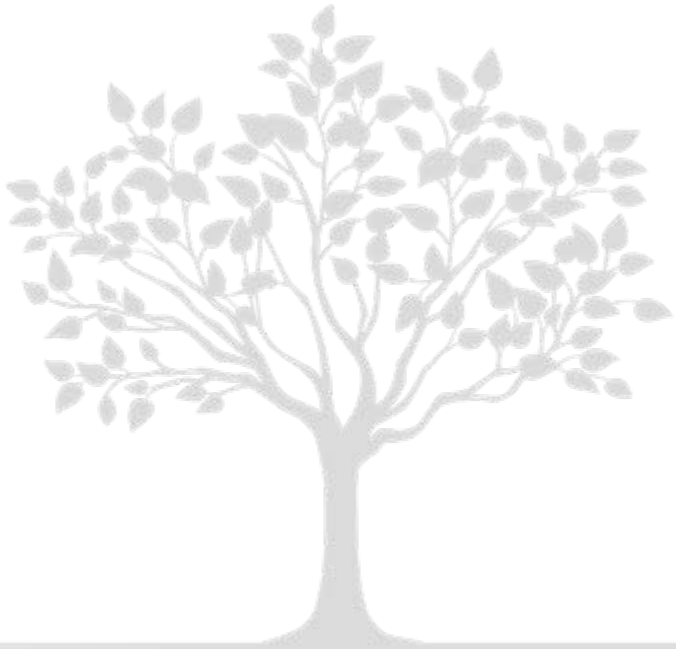
NFS Tuning for Oracle

Kyle Hailey

http://dboptimizer.com

Aug 2011



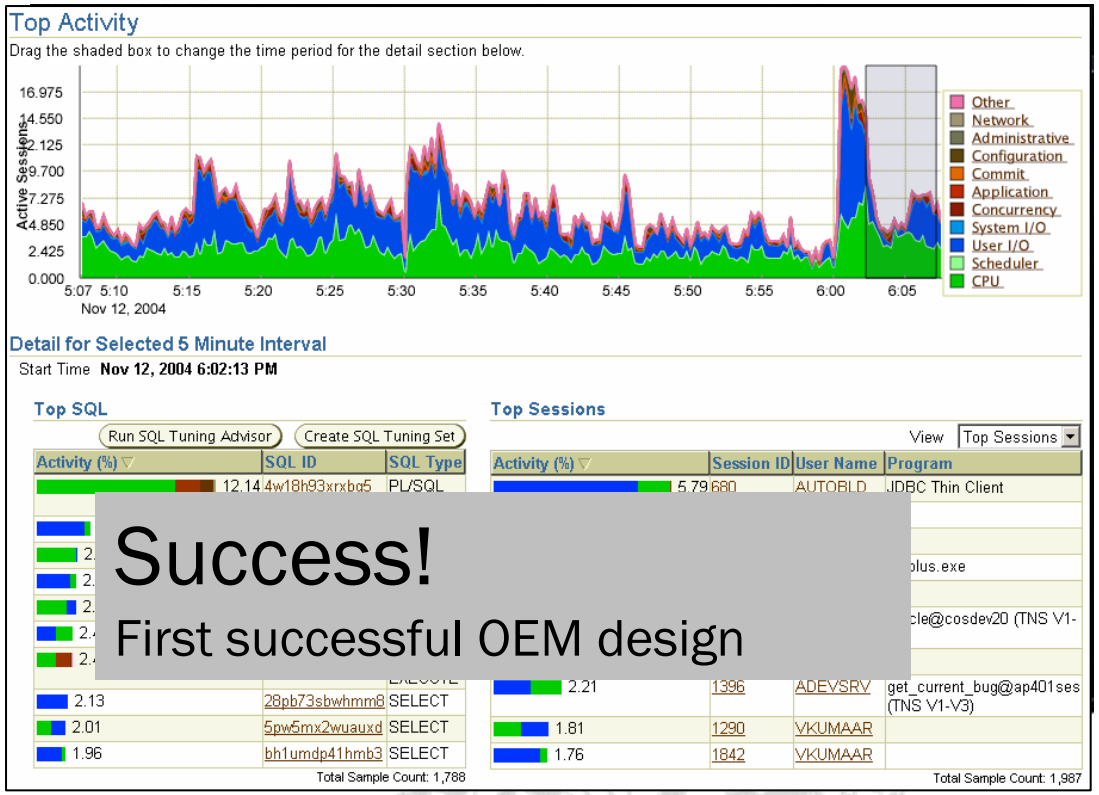
Intro

- Who am I
  - why NFS interests me
- DAS / NAS / SAN
  - Throughput
  - Latency
- NFS configuration issues*
  - Network topology
  - TCP config
  - NFS Mount Options

* for non-RAC, non-dNFS

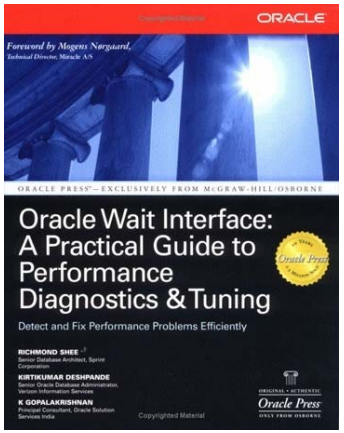
Who is Kyle Hailey

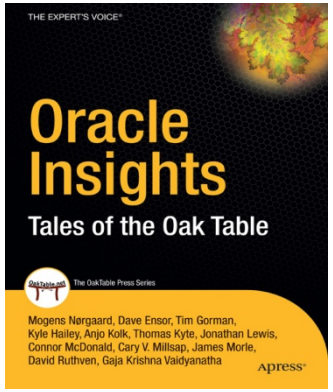
- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g



Who is Kyle Hailey

- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g
- 2005 Embarcadero
  - DB Optimizer
- Delphix

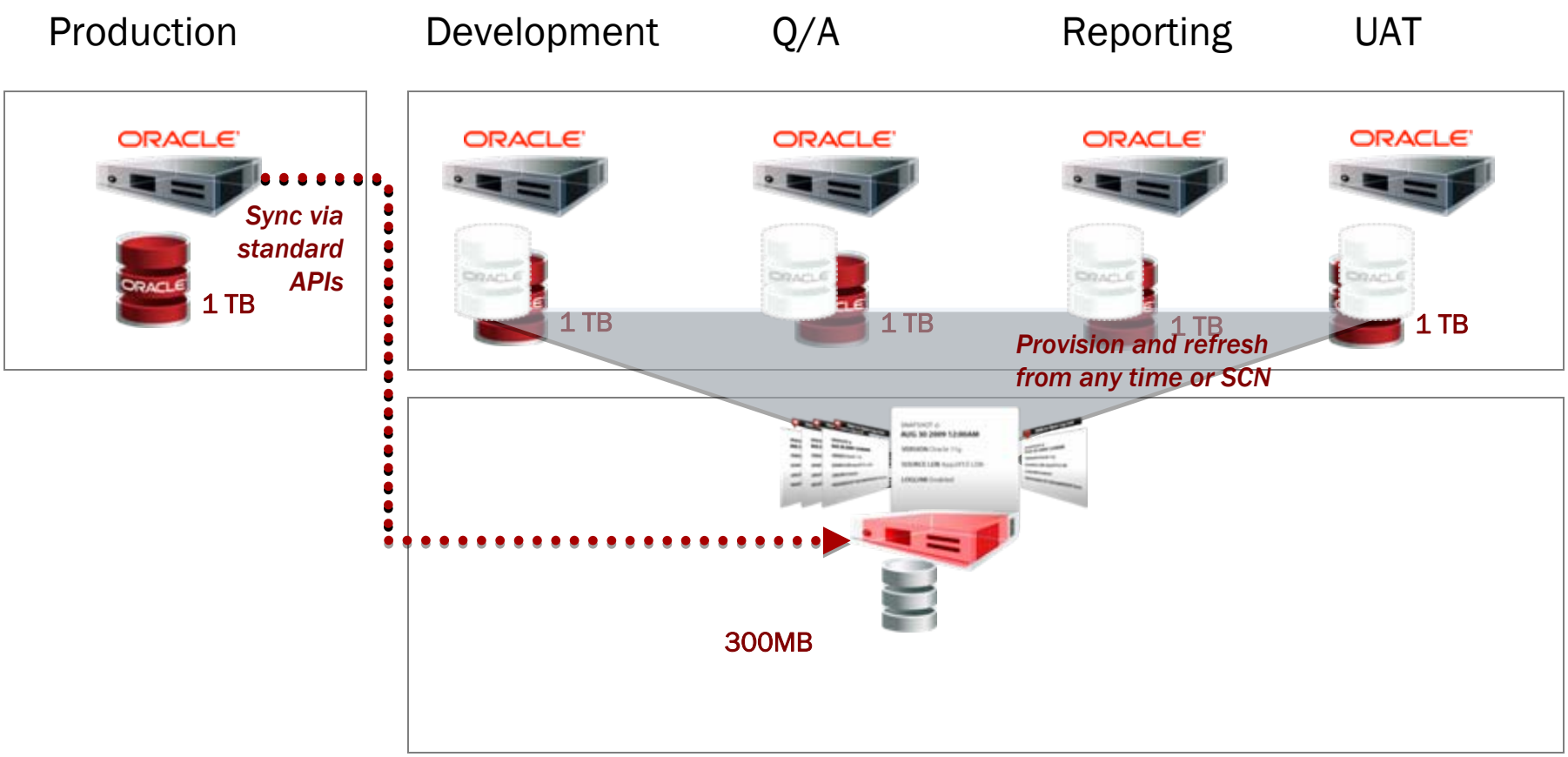




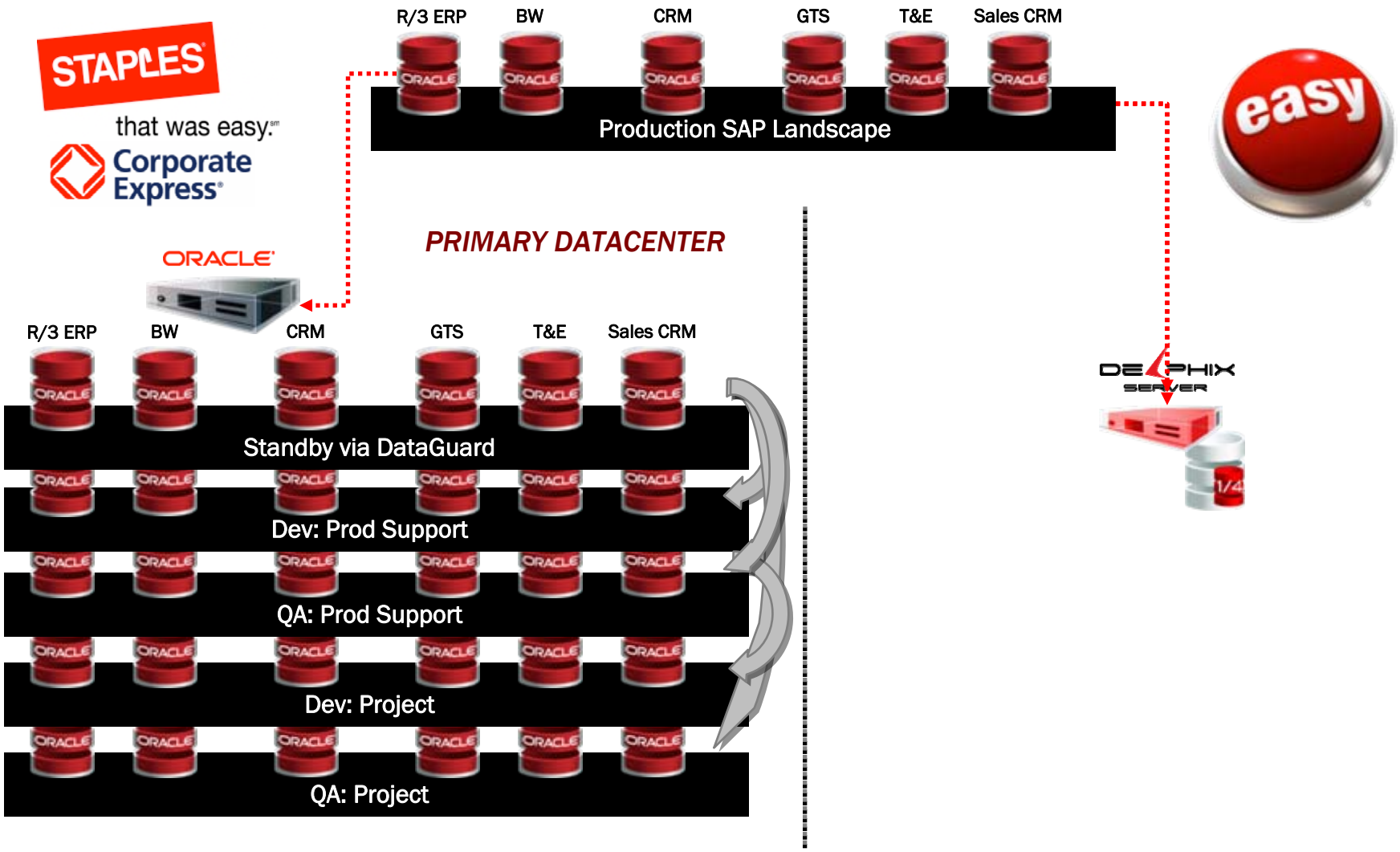
When not being a Geek

- Have a little 2 year old boy who takes up all my time
- and wonder how I missed the dot.com millions

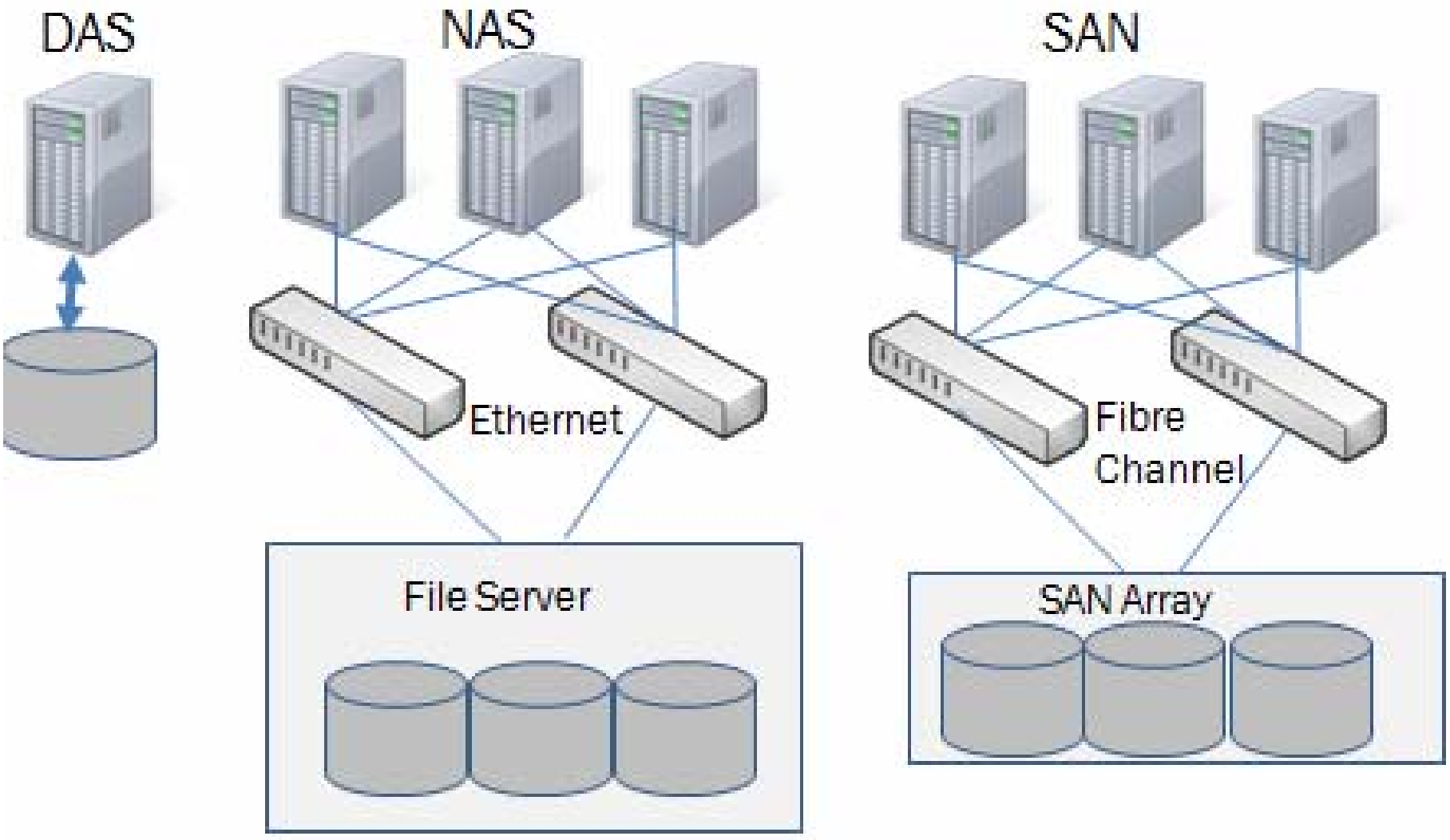
Fast, Non-disruptive Deployment



Combine Prod Support and DR

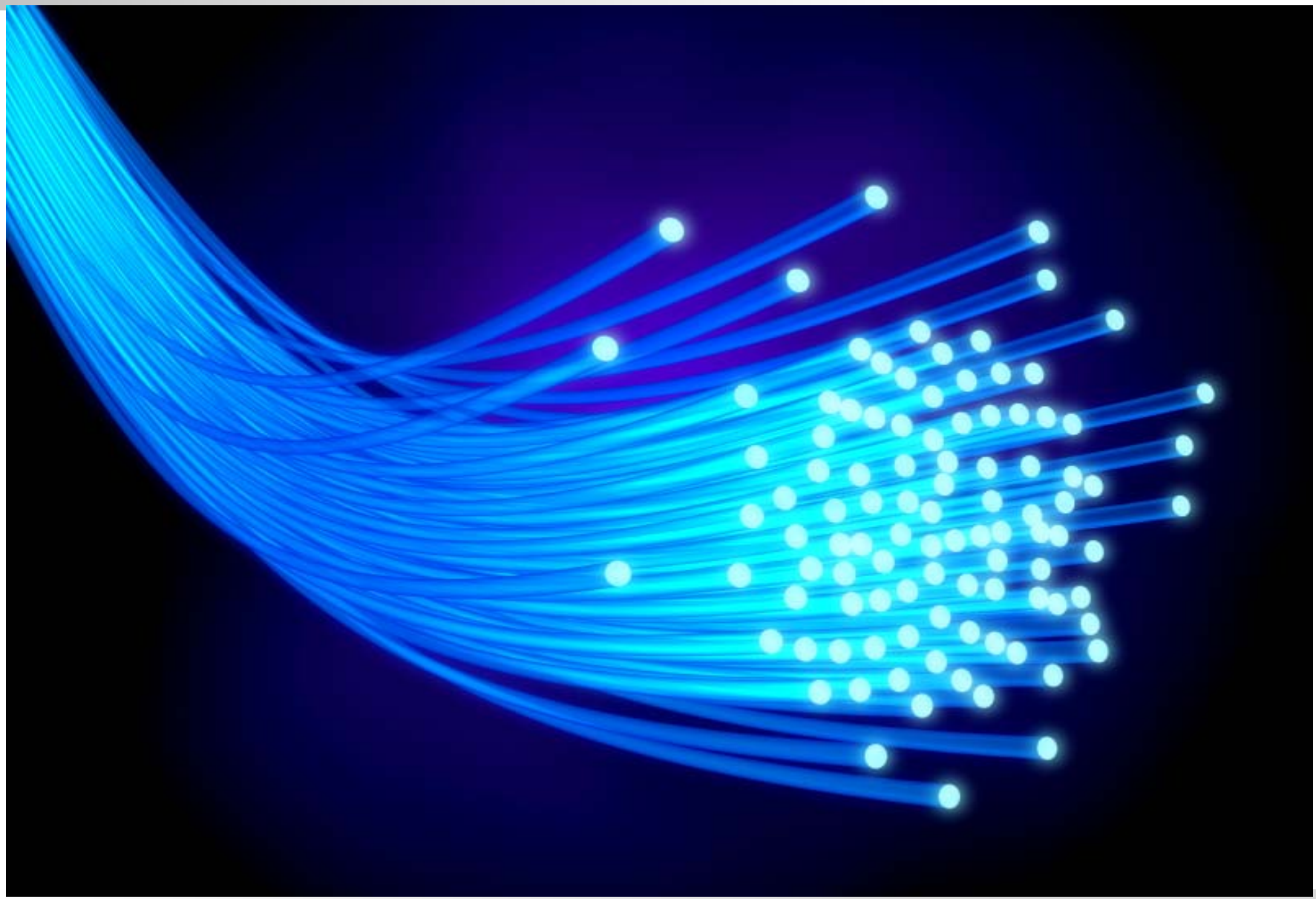


Which to use?



DAS is out of the picture

Fibre Channel



Manly men only use Fibre Channel

NFS - available everywhere

NFS is attractive but is it fast enough?

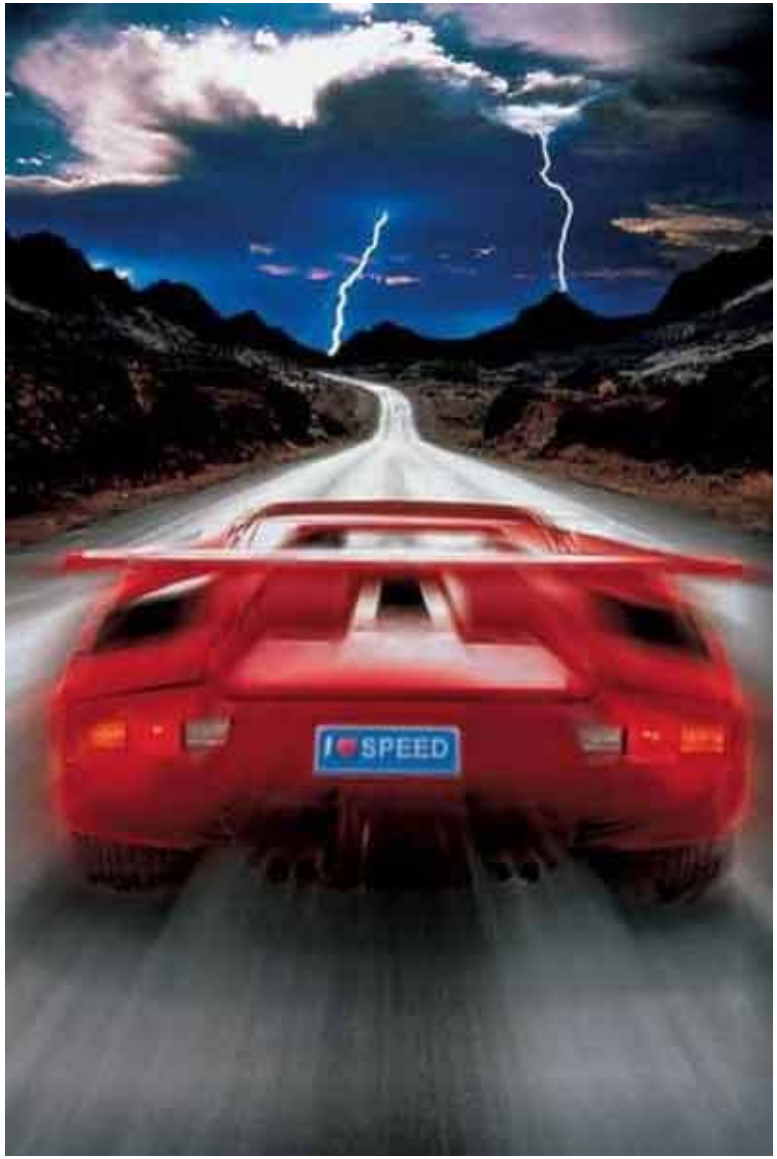
DAS vs NAS vs SAN

	attach	Agile	expensive	maintenance	speed
DAS	SCSI	no	no	difficult	fast
NAS	NFS - Ethernet	yes	no	easy	??
SAN	Fibre Channel	yes	yes	difficult	fast

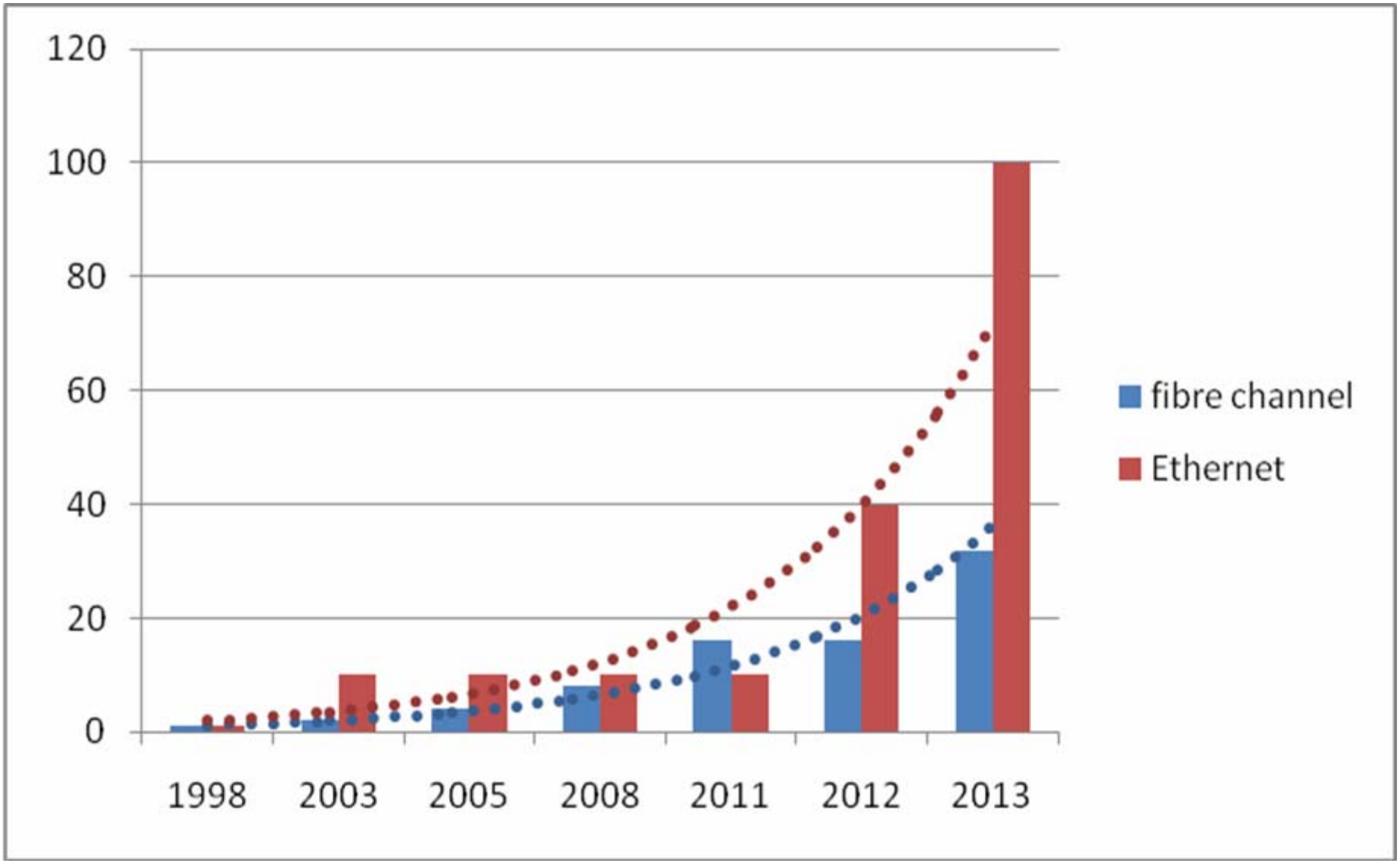
speed

Ethernet
- 100Mb 1994
- 1GbE - 1998
- 10GbE – 2003
- 40GbE – est. 2012
- 100GE –est. 2013

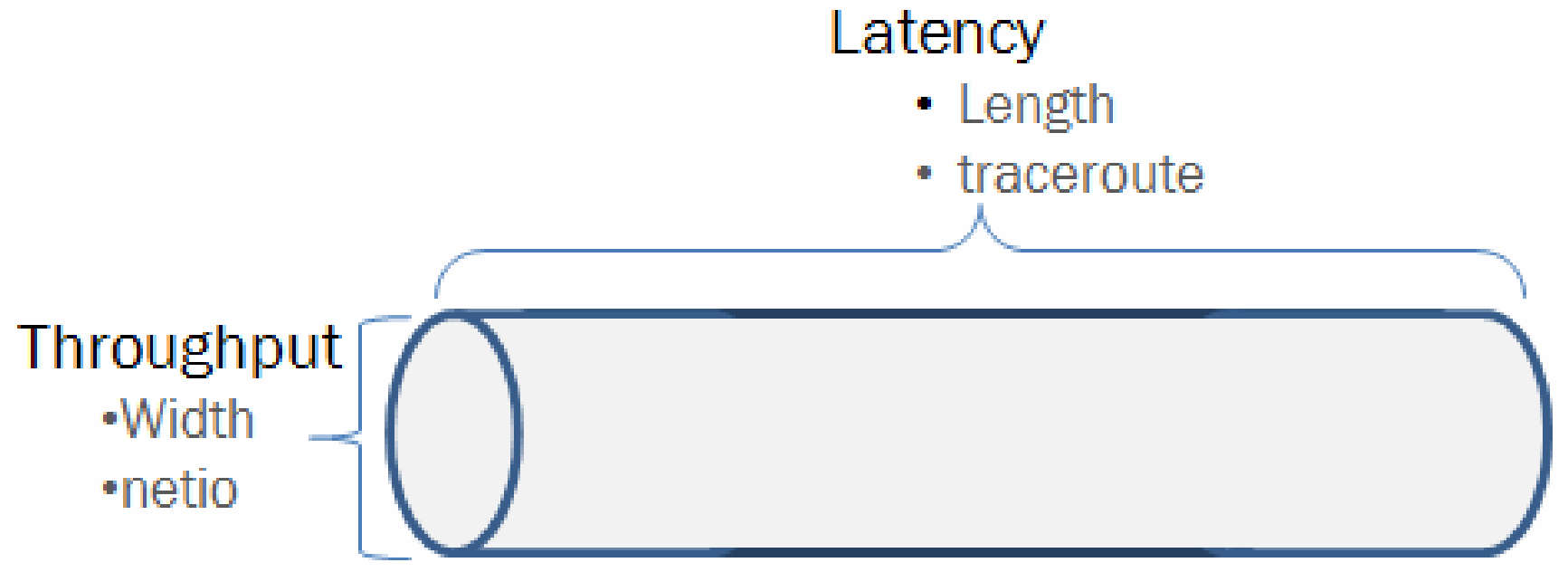
Fibre Channel
- 1G 1998
- 2G 2003
- 4G – 2005
- 8G – 2008
- 16G – 2011



Ethernet vs Fibre Channel



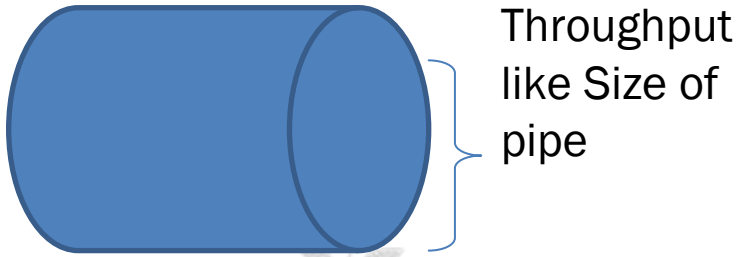
Throughput vs Latency



Throughput : netio

8b = 1Bytes

- 100MbE ~= 10MB/sec
- **1GbE ~= 100MB/sec** (125MB/sec max)
  - 30-60MB/sec typical, single threaded, mtu 1500
  - 90-115MB clean topology
- 10GbE ~= 1GB/sec



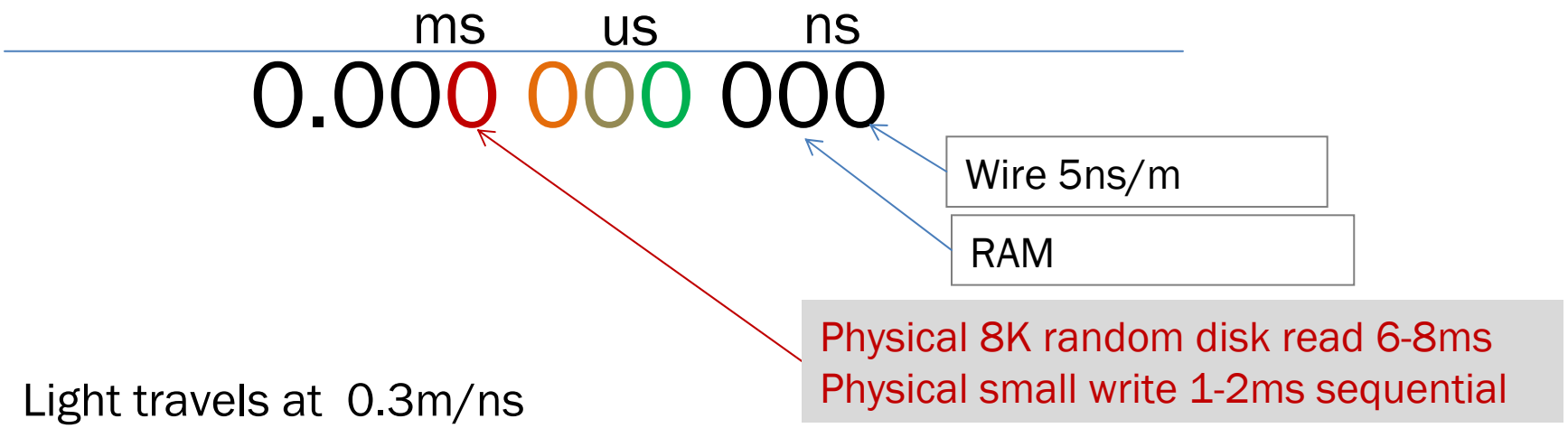
Server machine

```
netio -s -b 32k -t -p 1234
```

Target

```
netio -b 32k -t -p 1234 delphix_machine
Receiving from client, packet size 32k ...  104.37 MByte/s
Sending to client, packet size 32k ...  109.27 MByte/s
Done.
```

Wire Speed – where is the hold up?



Light travels at 0.3m/ns
If wire speed is 0.2m/ns

Data Center 10m = 50ns

LA to London is 30ms
LA to SF is 3ms
(5us/km)

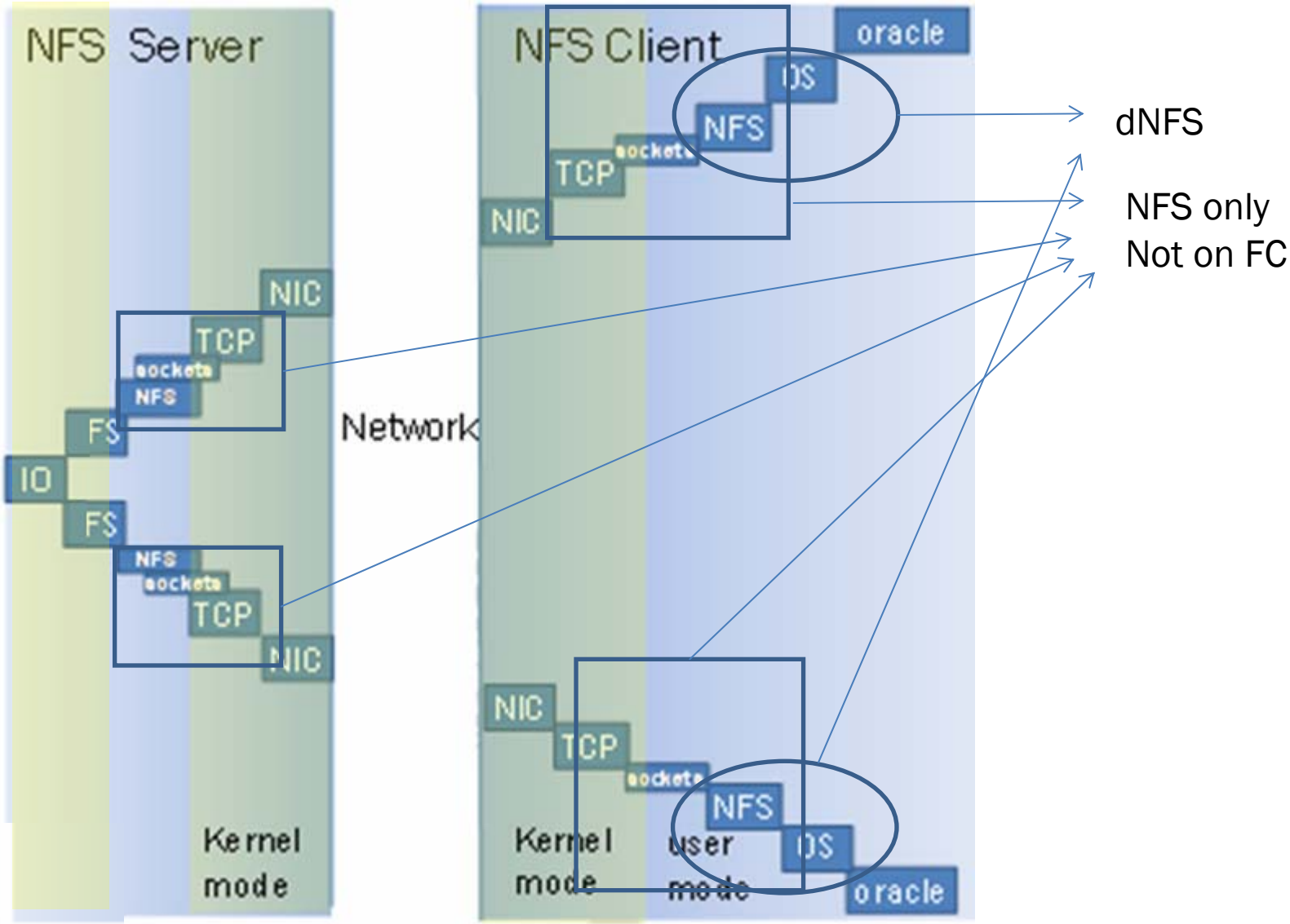
4G FC vs 10GbE

Why would FC be faster?

8K block transfer times

- 8GB FC = 10us
- 10G Ethernet= 8us

More stack more latency



Oracle and SUN benchmark

200us overhead more for NFS

	UFS	NFS
I/O Response Time	7.19 ms	7.39 ms

* I see 350us without Jumbo frames, and up to 1ms because of network topology

8K blocks 1GbE with Jumbo Frames , Solaris 9, Oracle 9.2
Database Performance with NAS: Optimizing Oracle on NFS
Revised May 2009 | TR-3322
http://media.netapp.com/documents/tr-3322.pdf

8K block NFS latency overhead

- 1GbE -> 80us
- 10GbE -> 8us
- 200us on 1GbE = 128us on 10GbE

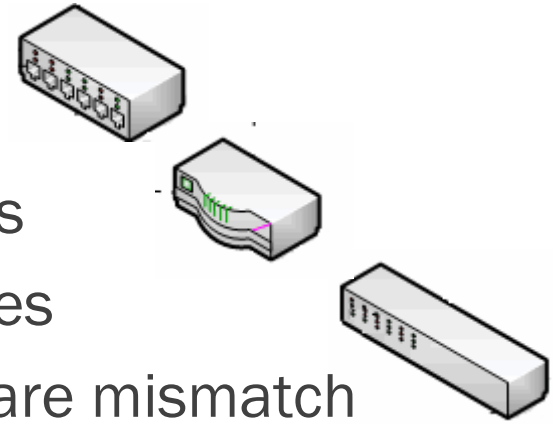
(0.128ms/7ms) * 100 = <u>1.8% latency increase over DAS*</u>

* 7ms as a typical 8K disk read

NFS why the bad reputation?

- Given 1.8% overhead why the reputation?
- Historically slower
- Setup can make a big difference
  1. Network topology and load
  2. NFS mount options
  3. TCP configuration
- Compounding issues
  - Oracle configuration
  - I/O subsystem response

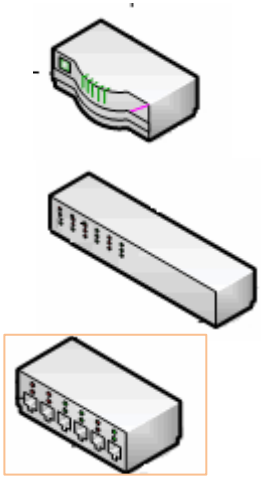
Network Topology



- Hubs
- Routers
- Switches
- Hardware mismatch
- Network Load

HUBs

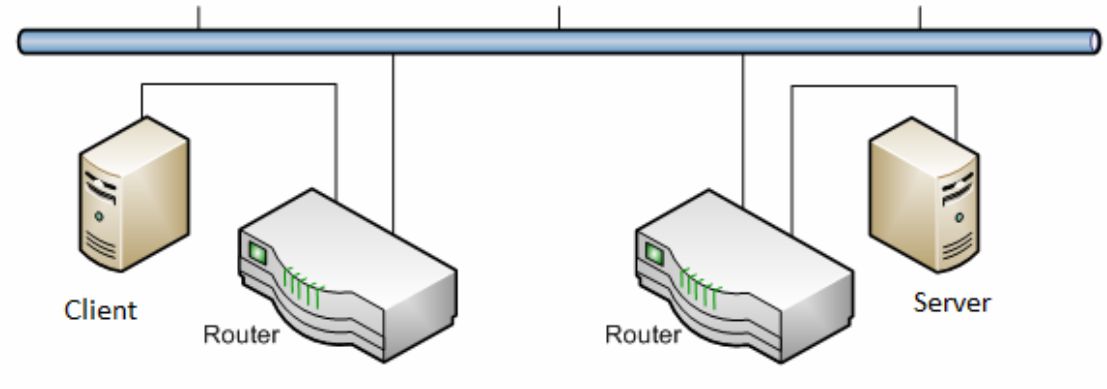
Layer	Name		
7	Application		
6	Presentation		
5	Session		
4	Transport		
3	Network	Routers	IP addr
2	Datalink	Switches	mac addr
1	Physical	Hubs	Wire



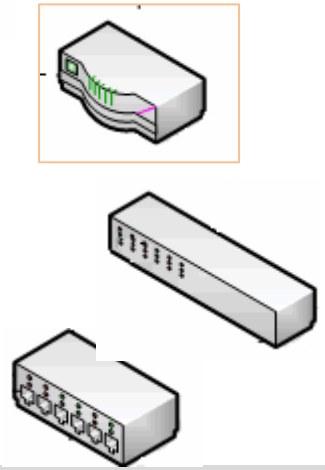
- Broadcast, repeaters
- Risk collisions
- Bandwidth contention

Routers

- Routers can add 300-500us latency
- If NFS latency is 350us (typical non-tuned) system
- Then each router multiplies latency 2x, 3x, 4x etc



Layer	Name		
3	Network	Routers	IP addr
2	Datalink	Switches	mac addr
1	Physical	Hubs	Wire



Routers: traceroute

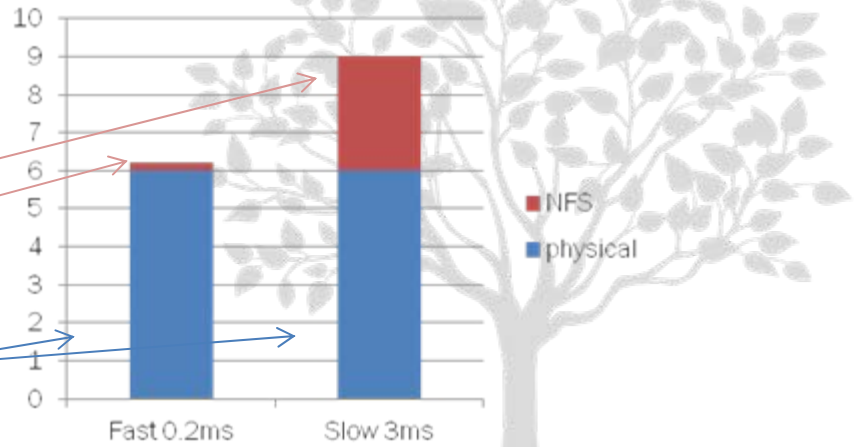
```
$ traceroute 101.88.123.195
 1  101.88.229.181 (101.88.229.181)  0.761 ms  0.579 ms  0.493 ms
 2  101.88.255.169 (101.88.255.169)  0.310 ms  0.286 ms  0.279 ms
 3  101.88.218.166 (101.88.218.166)  0.347 ms  0.300 ms  0.986 ms
 4  101.88.123.195 (101.88.123.195)  1.704 ms  1.972 ms  1.263 ms
```

```
$ traceroute 172.16.100.144
 1  172.16.100.144 (172.16.100.144)  0.226 ms  0.171 ms  0.123 ms
```

3.0 ms NFS on slow network

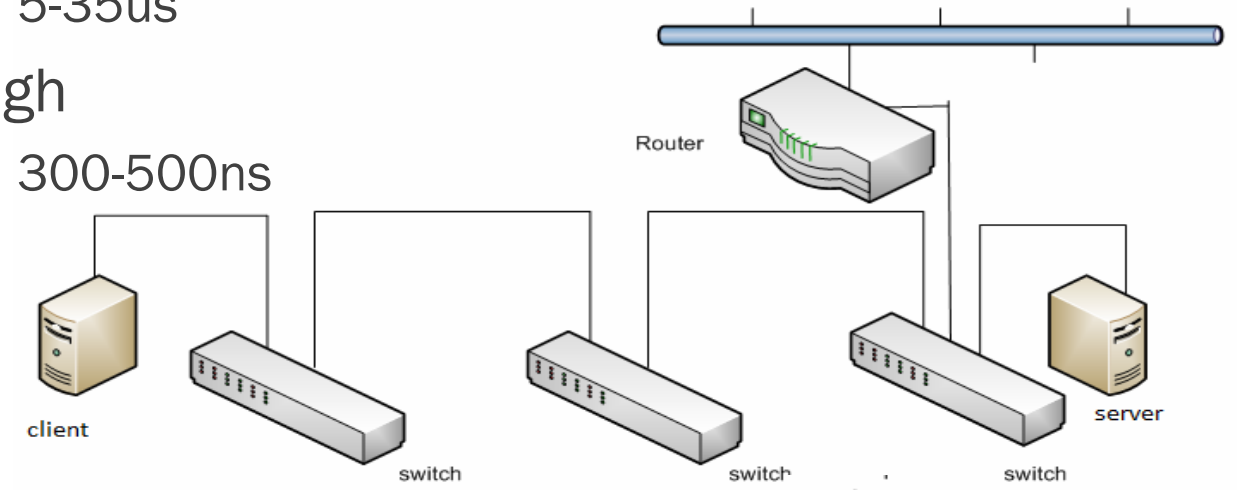
0.2 ms NFS good network

6.0 ms Typical physical read

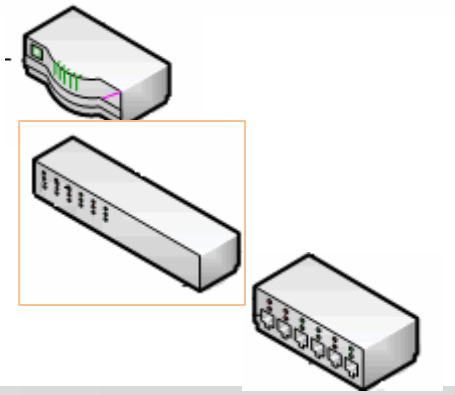


Multiple Switches

- Two types of Switches
  - Store and Forward
    - 1GbE 50-70us
    - 10GbE 5-35us
  - Cut through
    - 10GbE 300-500ns



Layer	Name		
3	Network	Routers	IP addr
2	Datalink	Switches	mac addr
1	Physical	Hubs	Wire



Hardware mismatch

- Speeds and duplex are often negotiated

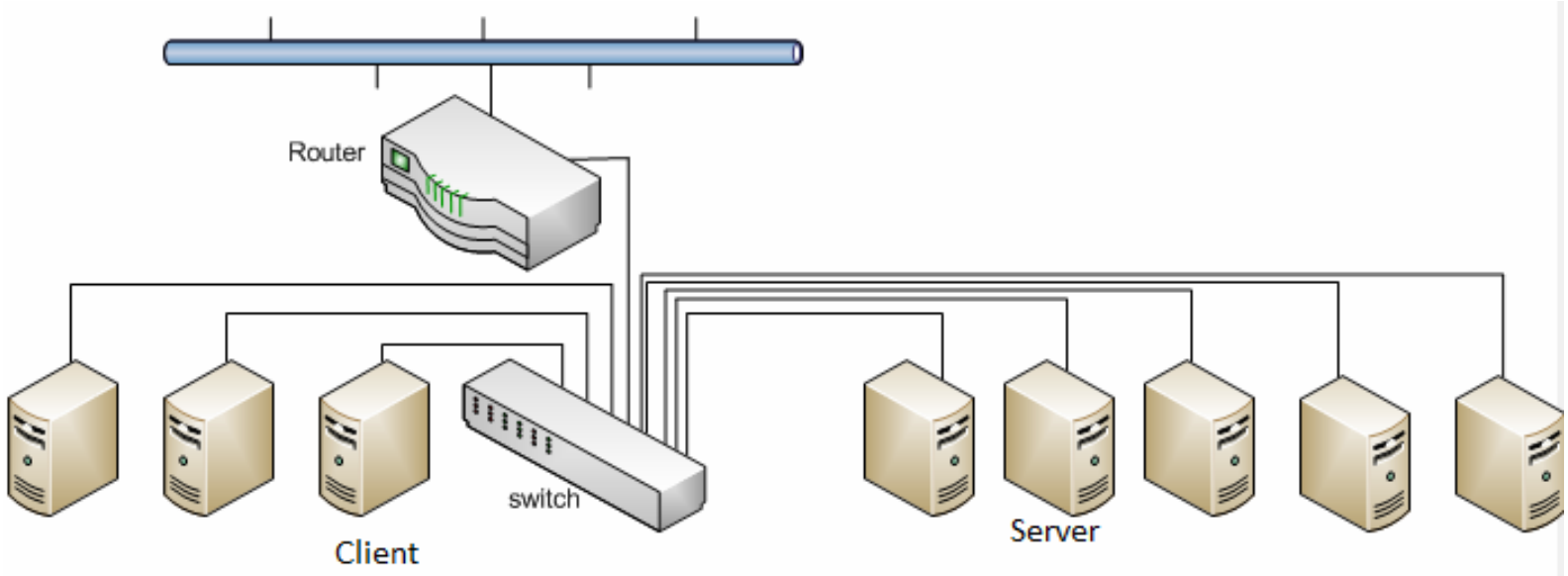
Example Linux:

```
$ ethtool eth0
Settings for eth0:
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
```

- Check that values are as expected

Busy Network

- Traffic can congest network
  - Caused drop packets
  - Out of order packets
  - Collisions on hubs, probably not with switches



Busy Network Monitoring

- Visibility difficult from any one machine
  - Client
  - Server
  - Switch(es)

```
$ nfsstat -cr
Client rpc:
Connection oriented:
badcalls   badxids    timeouts   newcreds   badverfs   timers
89101      6          0          5          0          0          0
```

```
$ netstat -s -P tcp 1
TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
        tcpRetransSegs      =  5986     tcpRetransBytes     =8268005
        tcpOutAck           =49277329   tcpOutAckDelayed    =473798
        tcpInDupAck         =357980     tcpInAckUnsent      =     0
        tcpInUnorderSegs    =10048089   tcpInUnorderBytes   =16611525
        tcpInDupSegs        = 62673     tcpInDupBytes       =87945913
        tcpInPartDupSegs    =    15     tcpInPartDupBytes   =   724
        tcpRttUpdate        =4857114    tcpTimRetrans       =  1191
        tcpTimRetransDrop   =     6     tcpTimKeepalive     =   248
```

Busy Network Testing

Netio is available here:http://www.ars.de/ars/ars.nsf/docs/netio

On Server box

```
netio -s -b 32k -t -p 1234
```

On Target box:

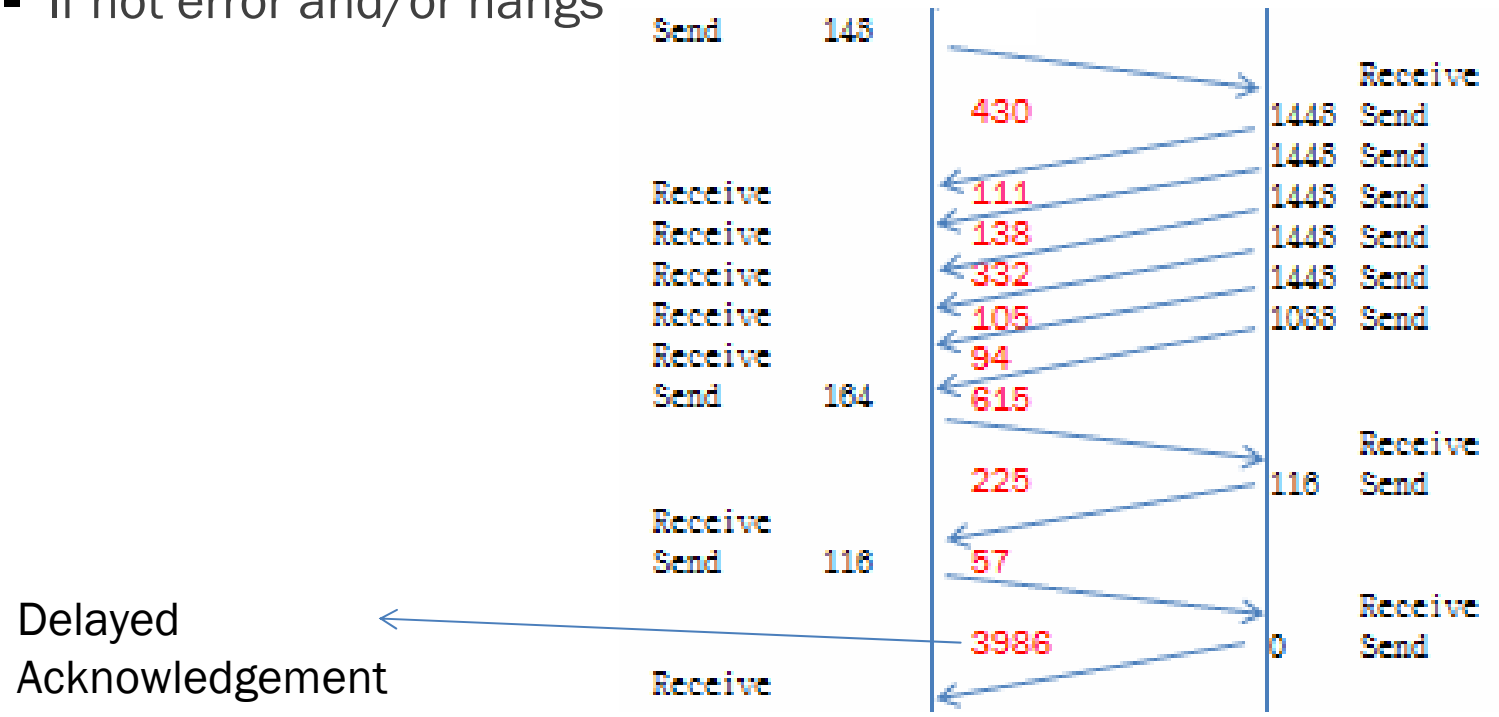
```
netio -b 32k -t -p 1234 delphix_machine
NETIO - Network Throughput Benchmark, Version 1.31
(C) 1997-2010 Kai Uwe Rommel
TCP server listening.
TCP connection established ...
Receiving from client, packet size 32k ...   104.37  MByte/s

Sending to client, packet size 32k ...   109.27  MByte/s
Done.
```

TCP Configuration

1. MTU (Jumbo Frames)
2. TCP window size (subset of Socket buffer sizes)
3. TCP congestion window sizes

MTU 9000 : Jumbo Frames

- MTU – maximum Transfer Unit
  - Typically 1500
  - Can be set 9000
  - All components have to support
    - If not error and/or hangs



Jumbo Frames : MTU 9000

8K block transfer

Test 1

Default MTU 1500

```
   delta    send      recd
                  <-- 164
     152     132  -->
      40    1448  -->
      67    1448  -->
      66    1448  -->
      53    1448  -->
      87    1448  -->
      95     952  -->
   = 560
```

Test 2

Change MTU

```
#  ifconfig eth1 mtu 9000 up
```

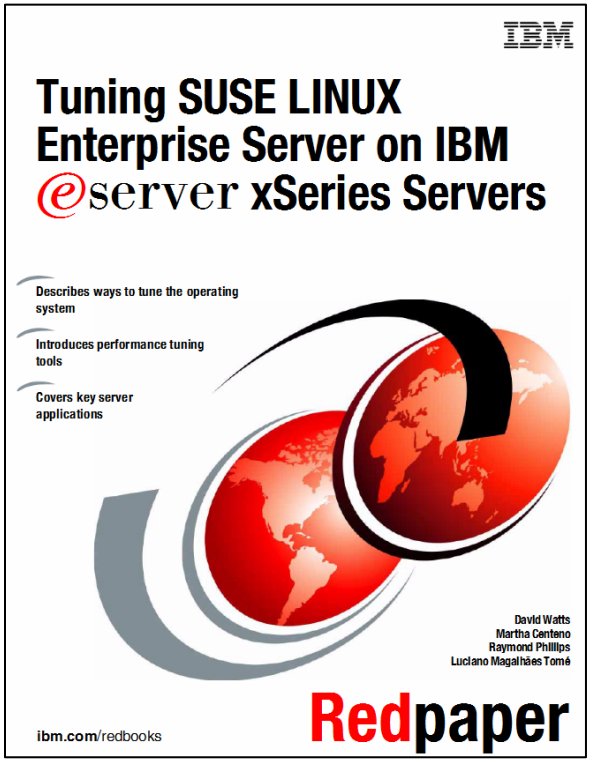
Now with MTU 900

```
 delta    send      recd
                  <-- 164
     273    8324  -->
```

Warning: MTU 9000 can hang if any of the hardware in the connection is configured only for MTU 1500

TCP Sockets

- Set max
  - Socket
  - TCP Window
- If maximum is reached, packets are dropped.



➔ LINUX

- Socket buffer sizes
  - sysctl -w net.core.wmem_max=8388608
  - sysctl -w net.core.rmem_max=8388608
- TCP Window sizes
  - sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
  - sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"

Excellent book

TCP window sizes

- max data send/receive
  - Subset of the TCP socket sizes

Calculating

= 2 * latency * throughput

Ex, 1ms latency, 1Gb NIC

= 2 * 1Gb/sec * 0.001s = 100Mb/sec * 1Byte/8bits= 250KB

Congestion window

delta us	bytes sent		bytes recvd sent	unack bytes	cong window	send window
31	1448	\		139760	144800	195200
33	1448	\		139760	144800	195200
29	1448	\		144104	146248	195200
31		/	0	145552	144800	195200
41	1448	\		145552	147696	195200
30		/	0	147000	144800	195200
22	1448	\		147000	76744	195200
28		/	0	147000	76744	195200
18	1448	\		147000	76744	195200

Unacknowledged bytes
Hits the congestion window
Though less than TCP send win

congestion window size is drastically lowered

NFS mount options

- Forcedirectio
- Rsize / wsize
- Actimeo=0, noac

Sun Solaris	rw,bg,hard,**rsize=32768,wsize=32768**,vers=3,[**forcedirectio** or llock],nointr,proto=tcp,suid
AIX	rw,bg,hard,**rsize=32768,wsize=32768**,vers=3,**cio**,intr,timeo=600,proto=tcp
HPUX	rw,bg,hard,**rsize=32768,wsize=32768**,vers=3,nointr,timeo=600,proto=tcp, suid, **forcedirectio**
Linux	rw,bg,hard,**rsize=32768,wsize=32768**,vers=3,nointr,timeo=600,tcp,**actimeo=0**

Forcedirectio

- Skip UNIX cache
- read directly into SGA
- Controlled by init.ora
  - Filesystemio_options=SETALL (or directio)
  - Except HPUX

Sun Solaris	**Forcedirectio – sets forces directio but not required** **Fielsystemio_options will set directio without mount option**
AIX	
HPUX	**Forcedirectio – only way to set directio** **Filesystemio_options has no affect**
Linux	

Direct I/O

Example query

77951 physical reads , 2nd execution

(ie when data should already be cached)

- 60 secs => direct I/O
- 5 secs => no direct I/O
- 2 secs => SGA

- Why use direct I/O?

Direct I/O

- Advantages
  - Faster reads from disk
  - Reduce CPU
  - Reduce memory contention
  - Faster access to data already in memory, in SGA
- Disadvantages
  - Less Flexible
  - More work
  - Risk of paging , memory pressure
  - Impossible to share memory between multiple databases

Cache	Rows/sec	Usr	sys
Unix File Cache	287,114	71	28
SGA w/ DIO	**695,700**	94	5

http://blogs.oracle.com/glennf/entry/where_do_you_cache_oracle

< 0.2 ms

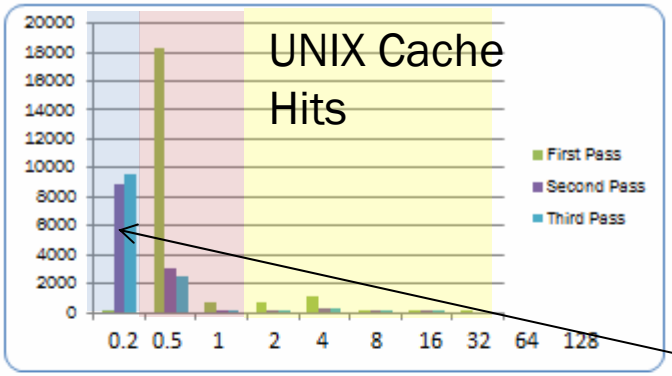
= UNIX Cache

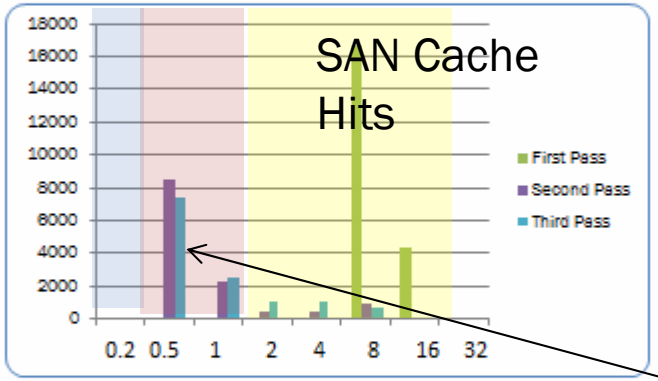
= NAS/SAN

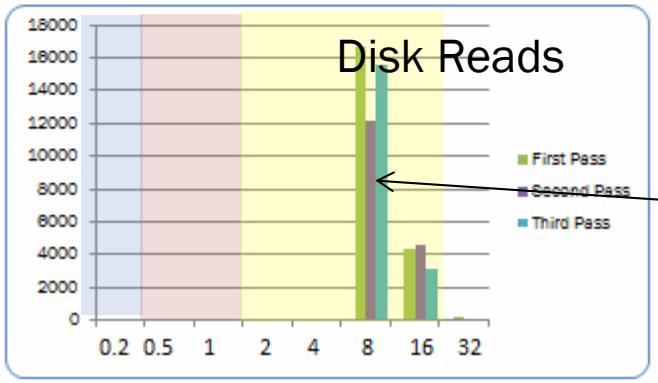
= Disk Reads

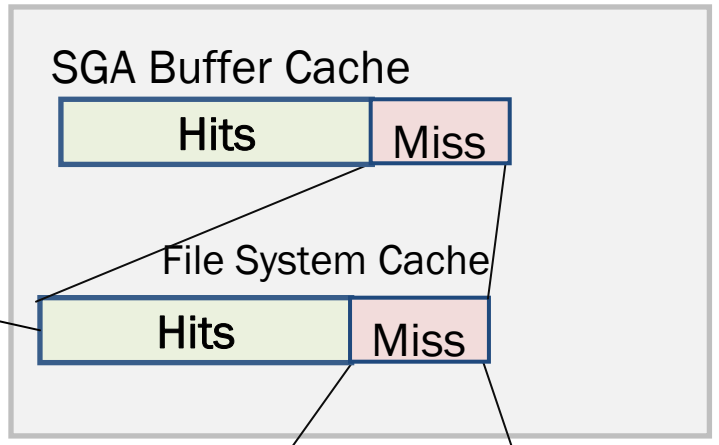
< 0.5 ms

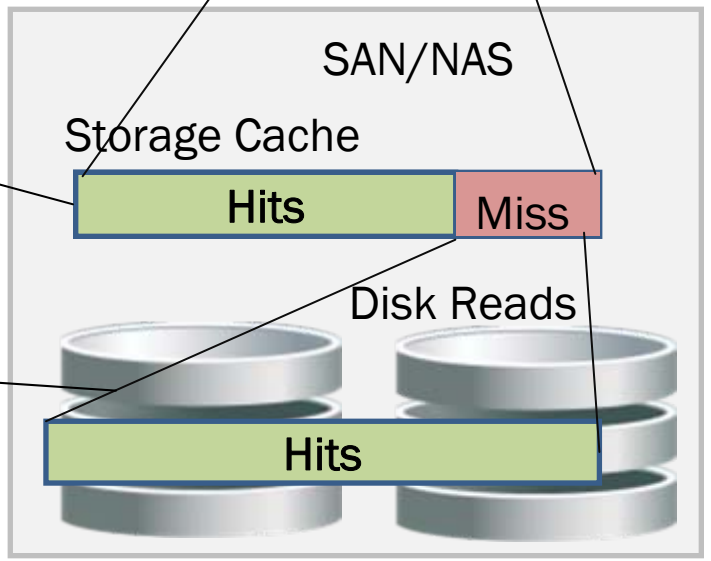
Disk Read
~ 6ms





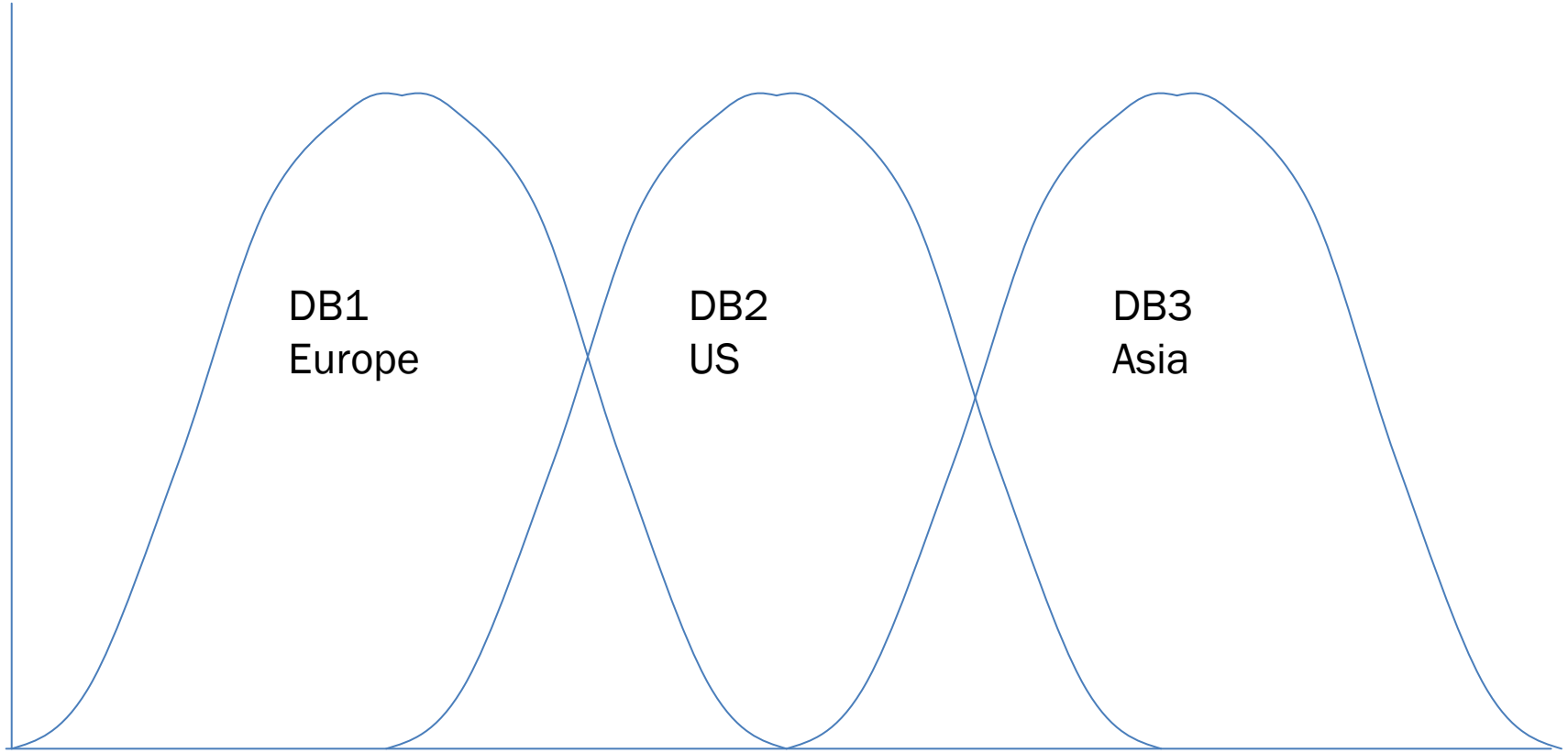






Direct I/O Challenges

Database Cache usage over 24 hours



ACTIMEO=0 , NOAC

- Disable client side file attribute cache
- Increases NFS calls
- increases latency
- Avoid on single instance Oracle
- Metalink says it’s required on LINUX
- Another metalink it should be taken off

=> It should be take off

rsize/wsize

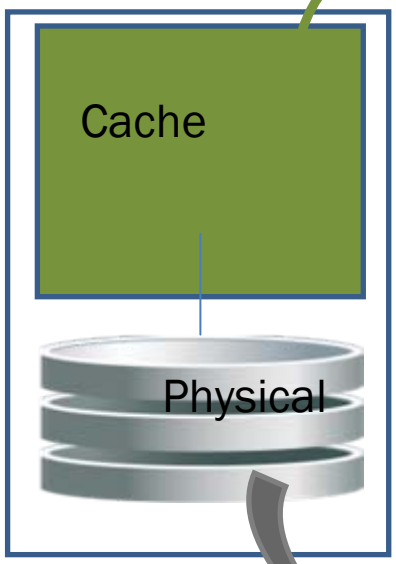
- NFS transfer buffer size
- Oracle says use 32K
- Platforms support higher values and can significantly impact throughput

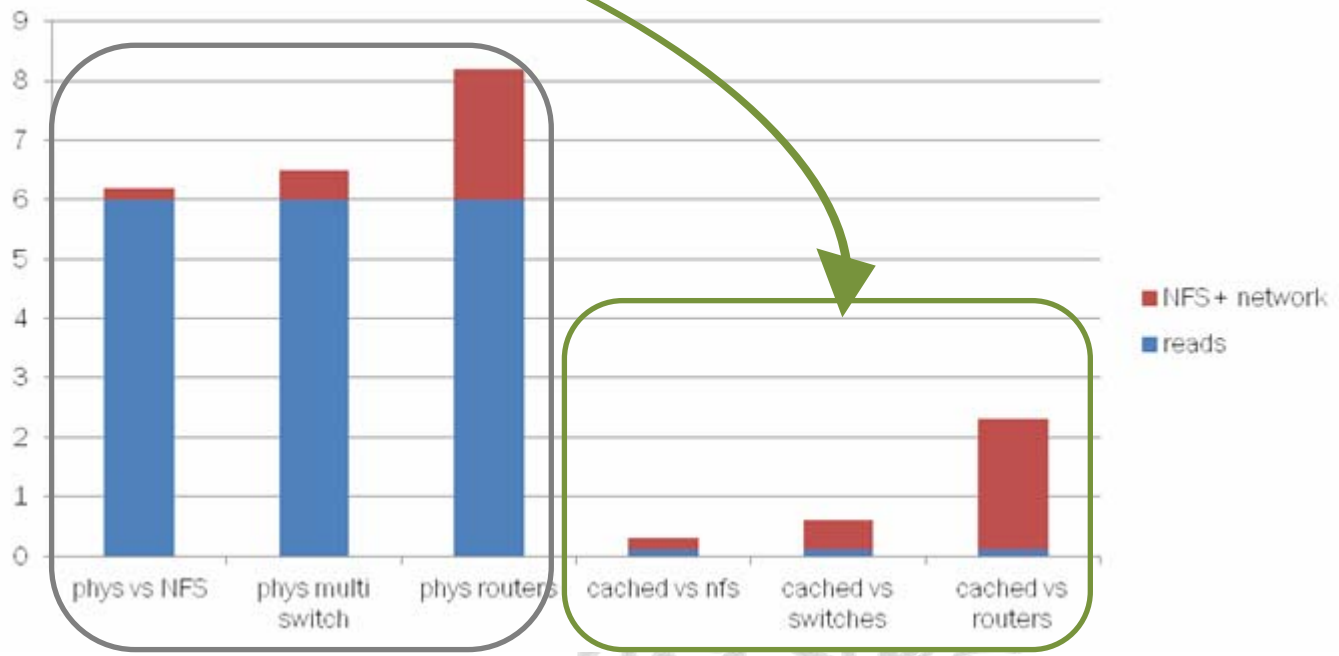
Sun Solaris	**rsize=32768,wsize=32768 , max is 1M**
AIX	**rsize=32768,wsize=32768 , max is 64K**
HPUX	r**size=32768,wsize=32768 , max is 1M**
Linux	r**size=32768,wsize=32768 , max is 1M**

On full table scans using 1M has halved the response time over 32K
Db_file_multiblock_read_count has to large enough take advantage of the size

NFS Overhead Physical vs Cached IO

100us extra over 6ms spindle read is small
100us extra over 100us cache read is 2x as slow
SAN cache is expensive – use it for write cache
Target cache is cheaper – put more on if need be





Takeaway: Move cache to client boxes

where it's cheaper and quicker, save SAN cache for writeback

Conclusions

- NFS close to FC , if
  - Network topology clean
  - Mount
    - Rsize/wsize at maximum,
    - Avoid actimeo=0 and noac
    - Use noactime
  - Jumbo Frames
- Drawbacks
  - Requires clean toplogy
  - NFS failover can take 10s of seconds
  - With Oracle 11g dNFS the client mount issues handled transparently

Conclusion: Give NFS some more love

gigabit switch can be anywhere from
10 to 50 times cheaper than an FC switch

Annan <3

dtrace

List the names of traceable probes:

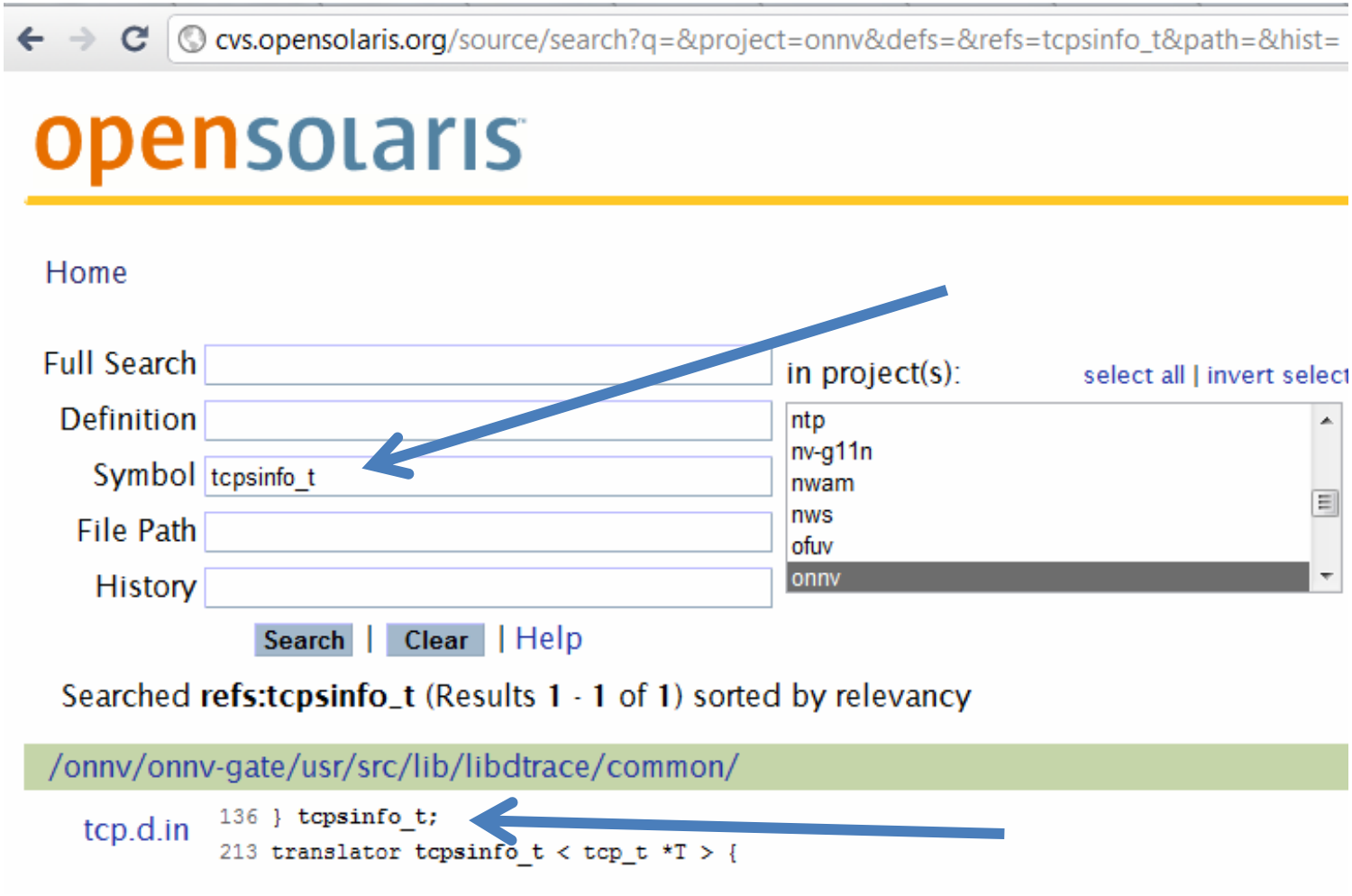
dtrace –ln provider:module:function:name

- -l = list instead of enable probes
- -n = Specify probe name to trace or list
- -v = Set verbose mode

Example
dtrace –ln tcp:::send

```
$ dtrace -lvn tcp:::receive
 5473  tcp  ip  tcp_output send

        Argument Types
                args[0]: pktinfo_t *
                args[1]: csinfo_t *
                args[2]: ipinfo_t *
                args[3]: tcpsinfo_t *
                args[4]: tcpinfo_t *
```

http://cvs.opensolaris.org/source/




cvs.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/lib/libdtrace/common/tcp.d.in#136
opensolaris
xref: /onnv/onnv-gate/usr/src/lib/libdtrace/common/tcp.d.in
Home | History | Annotate | Line # | Download |
Search
only in common
111 typedef struct tcpsinfo {
112 uintptr_t tcps_addr;
113 int tcps_local; /* is delivered locally, boolean */
114 int tcps_active; /* active open (from here), boolean */
115 uint16_t tcps_lport; /* local port */
116 uint16_t tcps_rport; /* remote port */
117 string tcps_laddr; /* local address, as a string */
118 string tcps_raddr; /* remote address, as a string */
119 int32_t tcps_state; /* TCP state */
120 uint32_t tcps_iss; /* Initial sequence # sent */
121 uint32_t tcps_suna; /* sequence # sent but unacked */
122 uint32_t tcps_snxt; /* next sequence # to send */
123 uint32_t tcps_rack; /* sequence # we have acked */
124 uint32_t tcps_rnxt; /* next sequence # expected */
125 uint32_t tcps_swnd; /* send window size */
126 int32_t tcps_snd_ws; /* send window scaling */
127 uint32_t tcps_rwnd; /* receive window size */
128 int32_t tcps_rcv_ws; /* receive window scaling */
129 uint32_t tcps_cwnd; /* congestion window */
130 uint32_t tcps_cwnd_ssthresh; /* threshold for congestion avoidance */
131 uint32_t tcps_sack_fack; /* SACK sequence # we have acked */

Dtrace

```
tcp:::send, tcp:::receive
{   delta= timestamp-walltime;
    walltime=timestamp;
    printf("%6d %6d %6d %8d \ %8s  %8d %8d %8d %8d  %d  \n",
        args[3]->tcps_snxt - args[3]->tcps_suna ,
        args[3]->tcps_rnxt - args[3]->tcps_rack,
        delta/1000,
        args[2]->ip_plength - args[4]->tcp_offset,
        "",
        args[3]->tcps_swnd,
        args[3]->tcps_rwnd,
        args[3]->tcps_cwnd,
        args[3]->tcps_retransmit
      );
}
tcp:::receive
{     delta=timestamp-walltime;
      walltime=timestamp;
      printf("%6d %6d %6d %8s / %-8d  %8d %8d %8d %8d  %d  \n",
        args[3]->tcps_snxt - args[3]->tcps_suna ,
        args[3]->tcps_rnxt - args[3]->tcps_rack,
        delta/1000,
        "",
        args[2]->ip_plength - args[4]->tcp_offset,
        args[3]->tcps_swnd,
        args[3]->tcps_rwnd,
        args[3]->tcps_cwnd,
        args[3]->tcps_retransmit
      );
}
```