**For:raj**
**Printed on:Mon, Feb 15, 1999 18:45:39**

**Document:SE 372.2.99**
**Last saved on:Mon, Feb 15, 1999 18:44:40**

# Network

# Performance



HEWLETT PACKARD

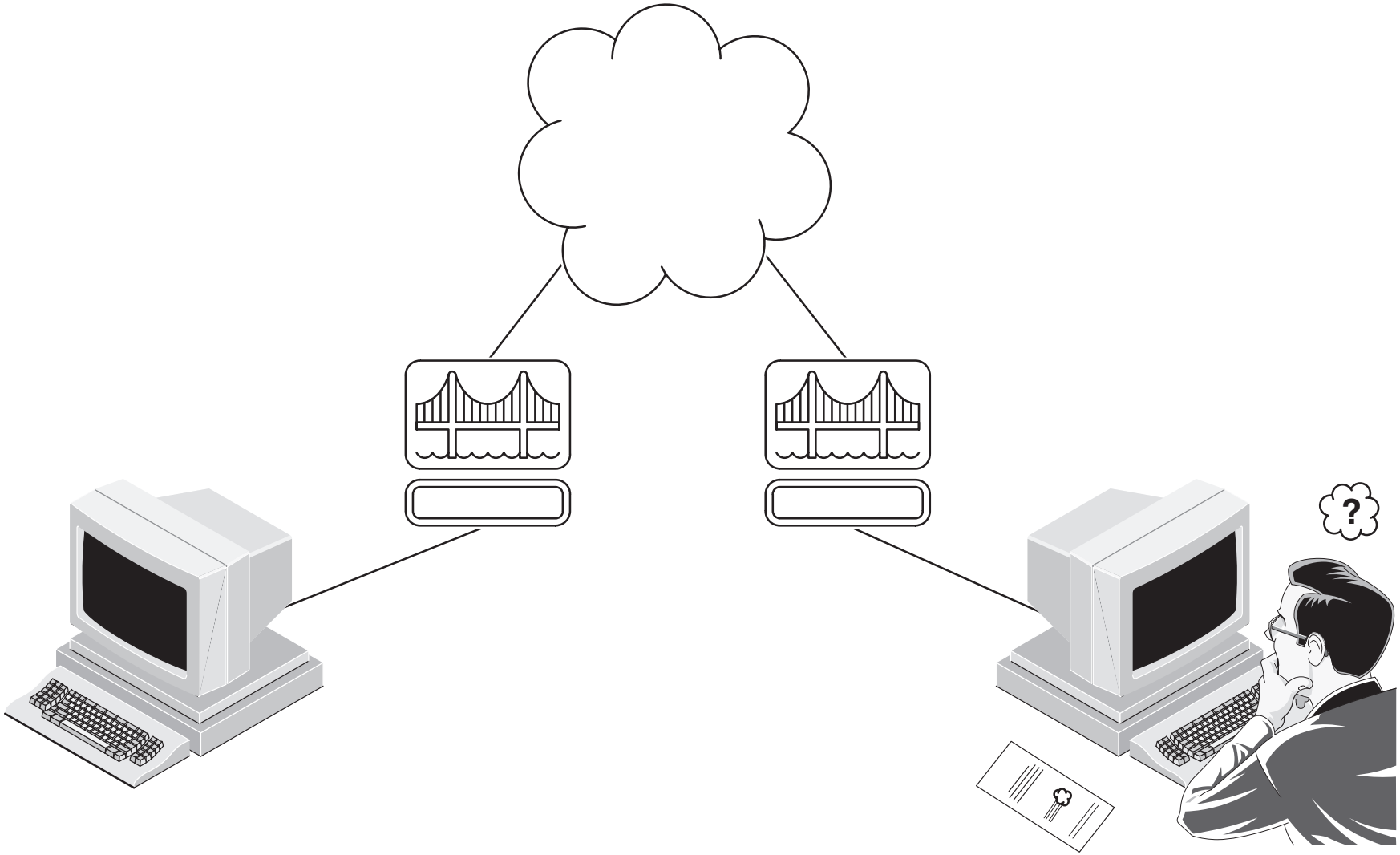
# Network Performance: Concepts

- The ‘‘System” does not end at the sheet metal
- Bandwidth does NOT guarantee Latency
- Performance begins with setsockopt()
- Performance ends with retransmissions
- Netstat and netperf are your netfriends
- Don’t raise the bridge, lower the river



HEWLETT PACKARD

# Network Performance: The “System”






HEWLETT PACKARD

# Network Performance: Bandwidth & Latency

- Bandwidth is Throughput – Work per Unit Time
- Latency is Response Time – Time per Unit Work
- High Bandwidth does not imply Low Latency
- Bandwidth X Latency gives you ‘‘pipe size”



HEWLETT PACKARD

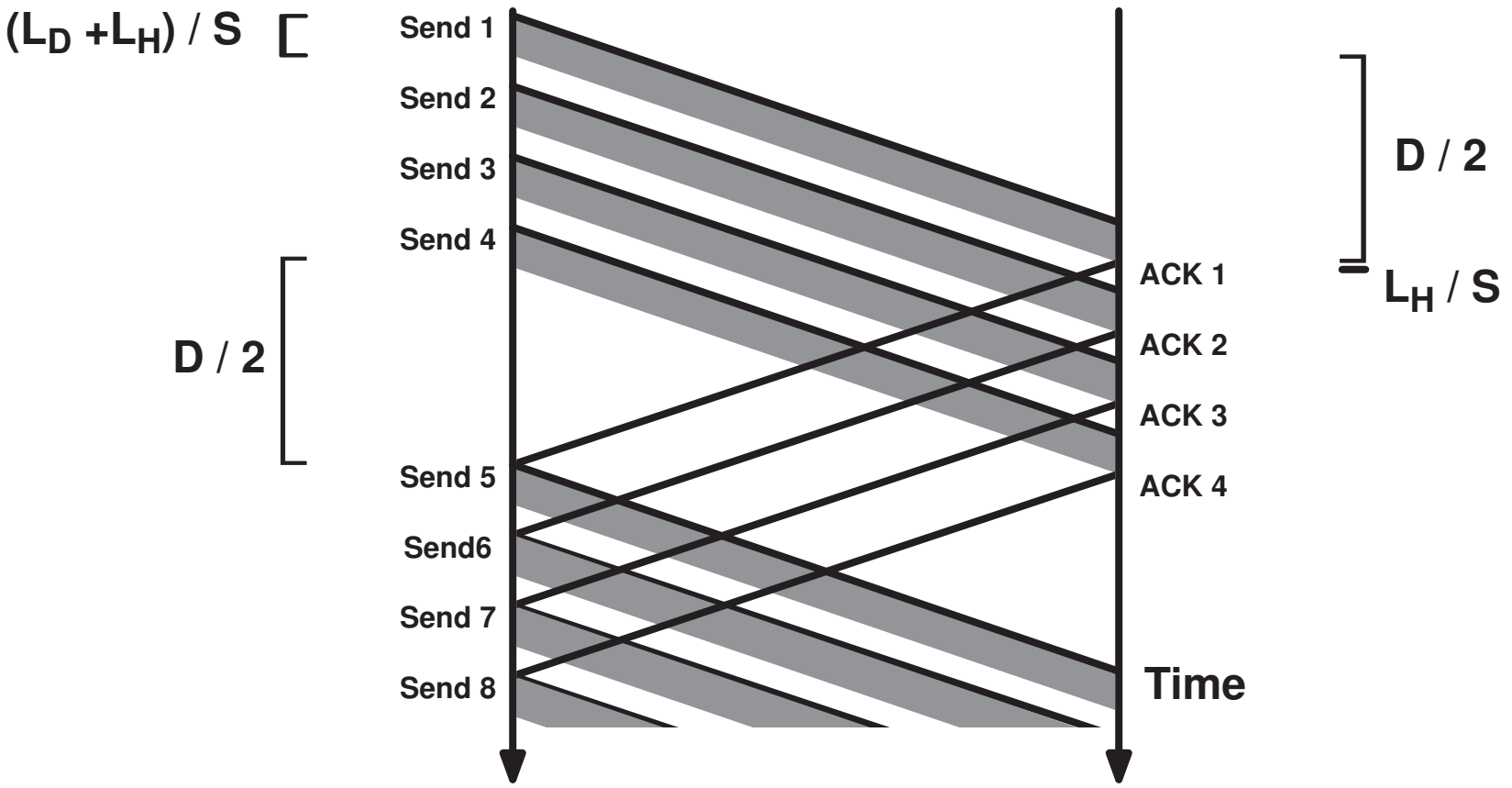
# TCP Performance: Theoretical (Network Perspective)

- D $\equiv$ Round Trip Time ; W $\equiv$ Window
- S $\equiv$ Bandwidth ; $L_D \equiv Length_{data}$; $L_H \equiv Length_{header}$
- $Perf_{win} \equiv (W/(D+(L_D+2L_H))/S)$
- $Perf_{eff} \equiv (S * (L_D/(L_D + L_H)))$
- TCP Performance $\approx$ min( $Perf_{win}$ , $Perf_{eff}$ )



HEWLETT PACKARD

# TCP Performance: $Perf_{win}$ Example W=4



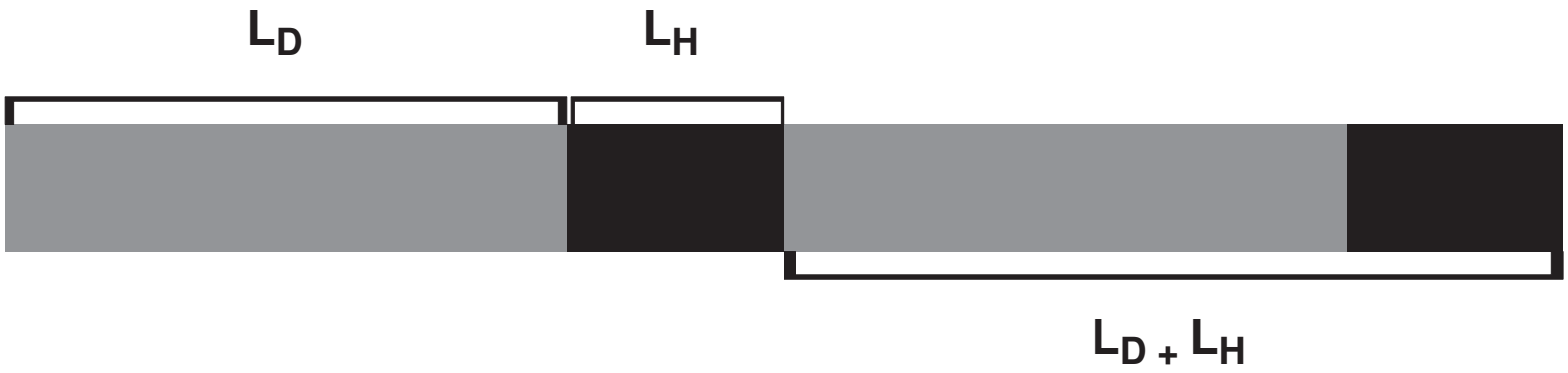

**Time from beginning of Send 1 to end of ACK 1 = $D + ((L_D + 2L_H) / S)$**

**In that time we sent "W" units of data**




HEWLETT PACKARD

# TCP Performance: $Perf_{eff}$ Example




**To deliver $L_D$ bytes to the receiver, we must transmit $L_D + L_H$ bytes on the wire. If the wire transmits at S, then our effective throughput will be $S * (L_D / (L_D + L_H))$**

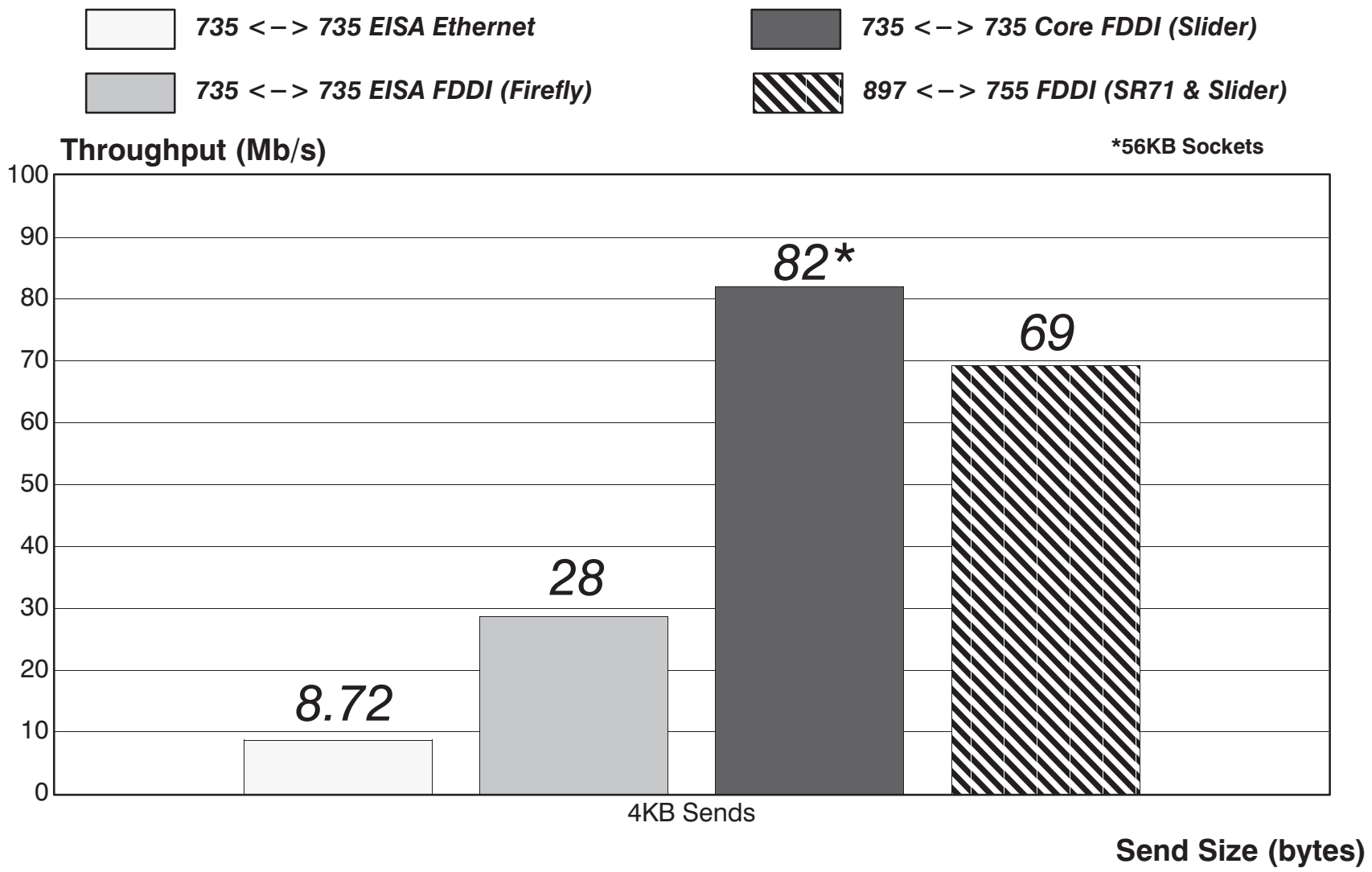
**Question: What about TCP ACKs?**




HEWLETT PACKARD

# Network Performance: Bandwidth & Latency

## TCP Throughput Performance

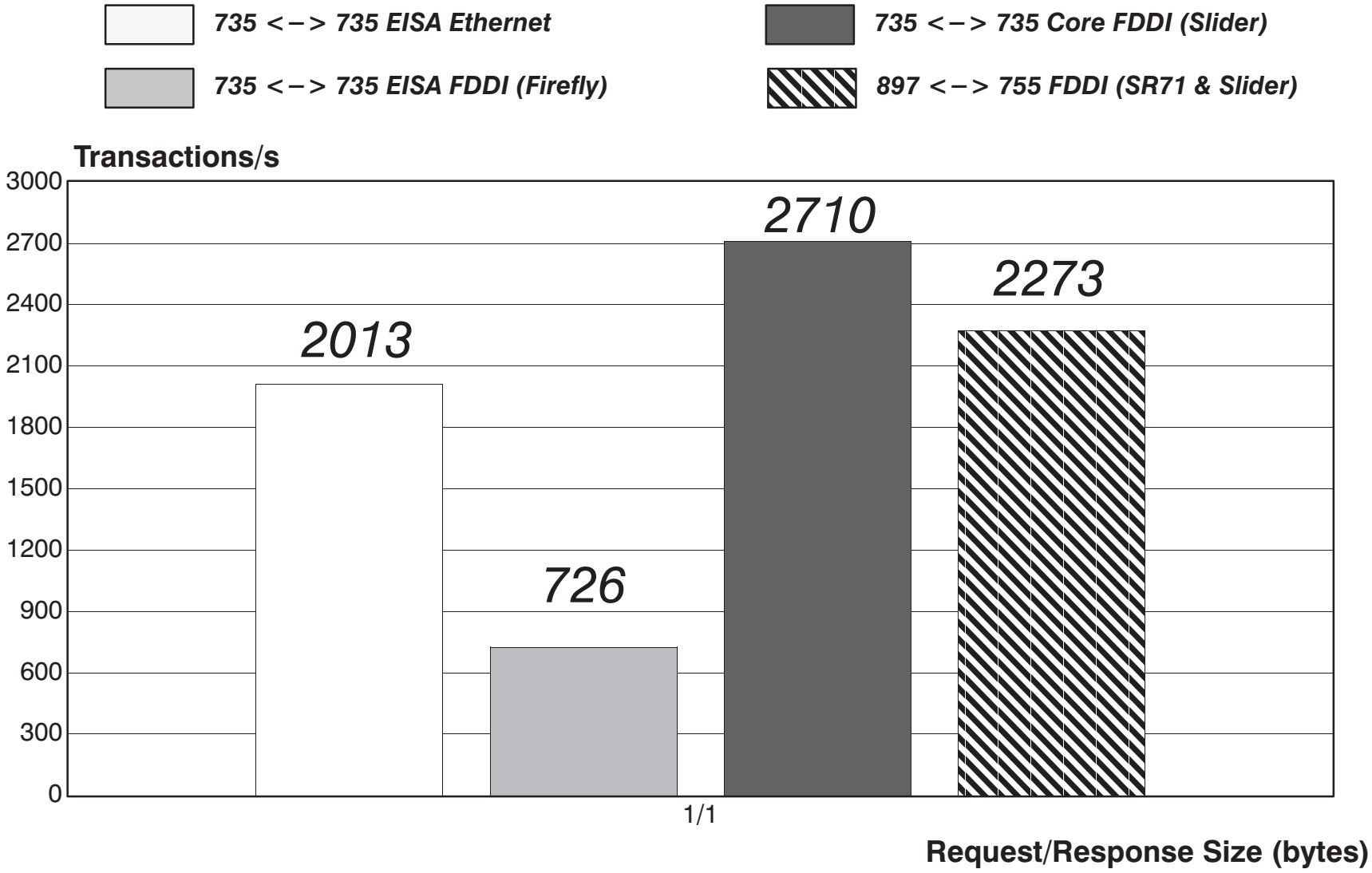





HEWLETT PACKARD

# Network Performance: Bandwidth & Latency

## TCP Request/Response Performance






HEWLETT PACKARD

# Network Performance: Tuning

- Retransmissions are evil. Find the cause and nuke it.
- Bigger buffers are better buffers for Throughput – use setsockopt() and big send() buffers.
- Biod's are NFS' ‘‘send buffers.”
- The only good data copy is one which never happens.
- Always write all “logically associated data” to the transport at one time.
- Setting TCP_NODELAY is often a kludge for a broken application.
- Save ndd tuning for last.

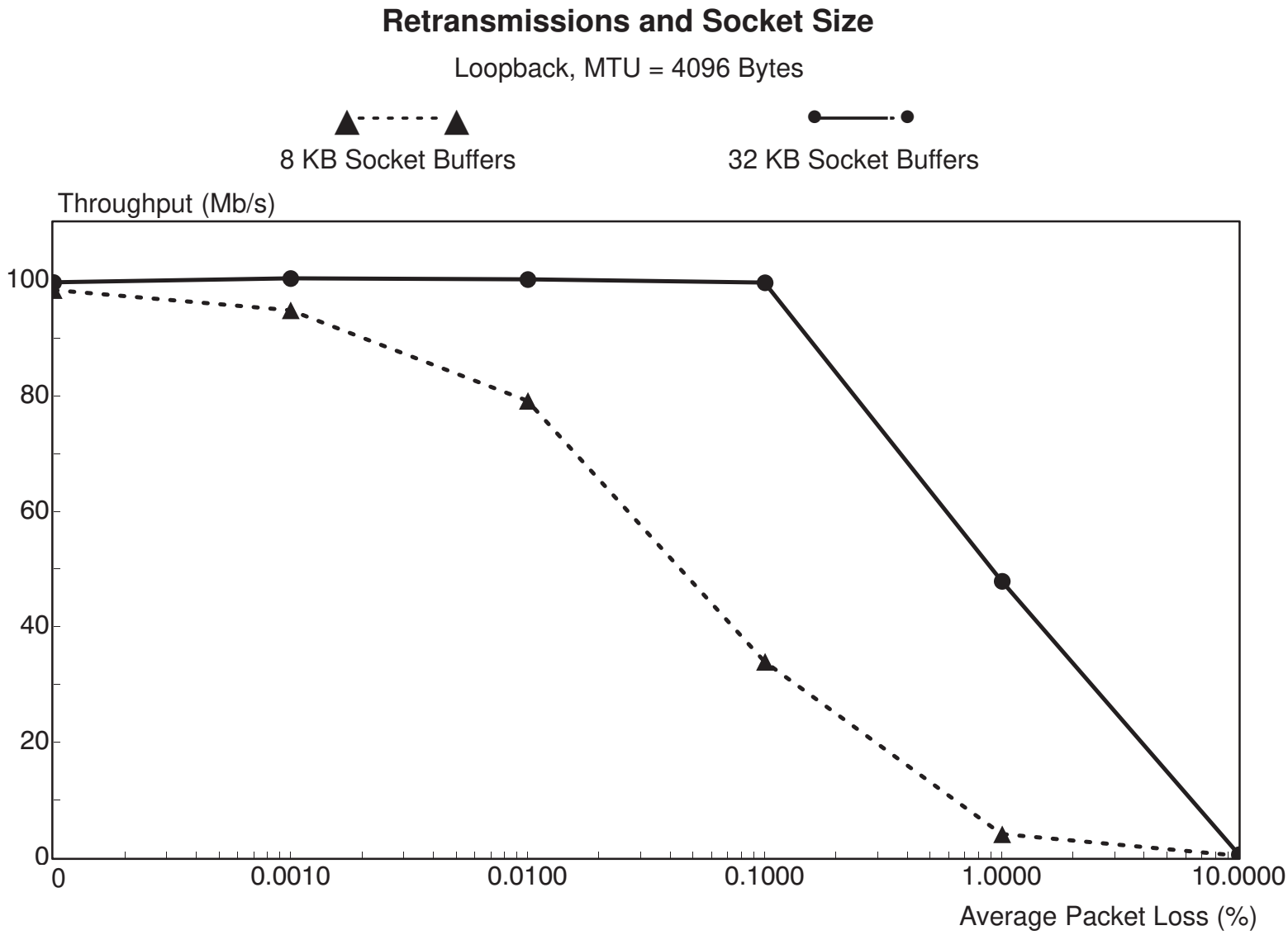
# Network Tuning: Retransmissions

- Two types of TCP retransmissions – "fast" and "slow"
  - TCP Fast retransmit happens without a timeout
  - TCP Slow retransmit waits for a timeout – ~0.5 seconds
- One type of NFS retransmission – slow
  - NFS retransmission is a timeout ~0.7 seconds



HEWLETT PACKARD

# Network Tuning: Retransmissions



# Network Tuning: Copy Avoidance

- If data sits in the filesystem, use the sendfile() system call
  - first provided in HP−UX 11.00
  - data flows “directly” from the FS buffer cache out the stack
  - optional header and/or trailer buffer may be supplied
- Used by both HP and third−party applications
  - FTP
  - rcp
  - Zeus Web Server
  - others to follow



HEWLETT PACKARD

# Network Performance: Benchmarks and Tools

- netperf — TCP, UDP, DLPI Throughput and Latency
- SPECweb96 — Web server static content performance
- SPEC SFS — NFS Server Throughput and Latency
- netstat — TCP/IP Statistics for the local system
- nfsstat — NFS Statistics for the local system
- OpenView — Statistics for all devices on the network
- Glance — Statistics for the local system
- ndd — new command in 11.0 to ‘‘tune” TCP et al



HEWLETT PACKARD

# Network Tuning: Guidelines

- Most ndd tuneables probably never need tuning
- Be certain you completely understand your applications
- Be certain you completely understand what the tune affects.
- Many settings can break your network connectivity.
  - tcp_ip_ttl
  - tcp_rexmit_interval_max
- Ndd settings can be store in /etc/rc.config.d/nddconf
  - settings will take effect on every reboot
- Don’t use a tuneable without an ndd −h description



HEWLETT PACKARD

# Network Tuning: potential ndd tunes

- tcp_[xmit|recv]_hiwater_def <*bytes*>
  - controls the default TCP window size for applications not calling setsockopt()
  - higher values might force a smaller TCP MSS
- tcp_conn_request_max <*backlog*>
  - the system's limit on the listen queue specified by the listen() socket call
- tcp_conn_strategy <*0|1*>
  - useful for systems with a high connection rate
  - tells the system to trade memory for CPU cycles

# Network Tuning: ndd tunes to avoid

- tcp_rexmit_interval_max
  - hinders tcp's ability to respond to congestion
  - may trigger spurious retransmissions and performance loss
- tcp_deferred_ack_interval
  - can lead to spurious retransmissions if too large
  - can lead to excessive control traffic if too small
- tcp_dupack_fast_retransmit
  - hinders fast retransmission of data if too large
  - leads to spurious fast retransmissions if too small



HEWLETT PACKARD

# Network Performance: Thinking

- Network performance is a larger task than System performance
- Something gone wrong elsewhere can affect you here
- Don’t be a net.sheep
- Use a scientific approach – create, examine, and validate a hypothesis
- Share what you learn and make the network ‘‘self diagnosing” – POST TO netnews



HEWLETT PACKARD