

---

# Sockets-GM

*- Mapping Distributed Applications to Myrinet -*



**MUG-2002**

Markus Fischer  
Myricom, Inc.  
fischer@myri.com

Myrinet Users Group Conference  
Vienna, Austria  
13 May 2002



**MUG-2002**

# Outline

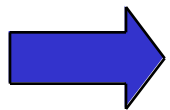
---

- Motivation
  - Overall Goal for Sockets-GM
- Available Platforms
  - BSD: Linux / Solaris
  - Winsock 1.1/ Winsock 2: Windows
- Implementation Details
  - Shared/Static Libraries
  - LSP's and Winsock Direct (SDP)
- Performance Measurements, Benchmarks and Applications
  - Tuning TCP/IP
  - Netperf, Netpipe, Iperf, NTttcp, ...
  - SQL Server
- Conclusion

# Motivation: Problems with Existing Transport Protocol Stacks

---

- Excessive CPU utilization
  - Memory-to-memory copying of data
  - User context switches
  - User/kernel transitions
  - Interrupts
  - Protocol overhead
    - Increased Latency
    - Decreased Bandwidth
- Excessive memory bandwidth utilization
  - Copying of data increases memory bandwidth requirements
- TCP/IP Performance for W2K
  - GigEth = 85 MB/s, 80usec Latency
  - GM = ~ ( 7usec, 247 MB/s)



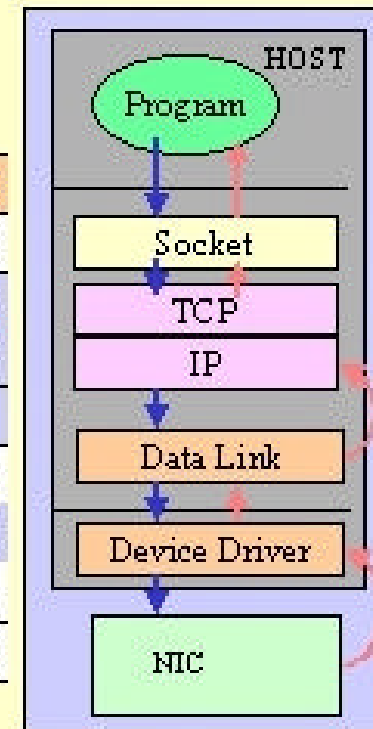
**How can existing, distributed applications using TCP/IP be improved ?**

# TCP/IP Overhead

## TCP/IP Protocol Overhead Analysis on Linux 2.2.12

| Processing                  | Overhead  | Rate % |
|-----------------------------|-----------|--------|
| System call and socket      | 1.6 usec  | 3.6    |
| TCP                         | 15.5 usec | 34.6   |
| IP                          | 6.2 usec  | 13.8   |
| Protocol Handler Invocation | 3.2 usec  | 7.1    |
| Device Driver               | 4.7 usec  | 10.5   |
| Hardware Interrupt          | 5.9 usec  | 13.2   |
| NIC+Media                   | 7.7 usec  | 17.2   |
| Total (1/2 Round Trip)      | 44.8 usec | 100    |

- TCP/IP: 48.4%, Hardware Interrupt: 13.2 %  
Protocol Handler Invocation: 7.1%



From RWCP

# Sockets-GM ...

---

- ... Provides a thin layer which mimics TCP/IP semantics
- ... Includes additional control to original socket functions
  - Use Companion Sockets for high speed data transfer
  - Challenge: map existing TCP/IP calls to Sockets-GM layer
- ... Mimics System, Kernel Functionality !
  - Detect processes which got a SegV, Kill, Control C ... !
  - This is User Level Communication
- ... is BSD Sockets compliant for Linux / Solaris
  - Supports fork ()
- ... is Winsock1.1 and Winsock 2 compliant
  - Winsock 2 adds overlapping functions!
  - A bunch of additional socket functions !
  - Other functions can perform operations on socket handles
  - You are allowed to combine Winsock 1.1 & Winsock 2 functions

# Simplifications Possible using Sockets-GM

---

- Protocol offload
  - Reliable, in-order delivery in hardware
    - Msg Flow at HW level
  - NIC de-multiplexes data stream instead of Operating System
    - GM Port is given to process
  - Avoid copying of data when appropriate
  - Use of RDMA Reads and Writes enables direct data placement into application buffer
- Kernel bypass
- Interrupt avoidance

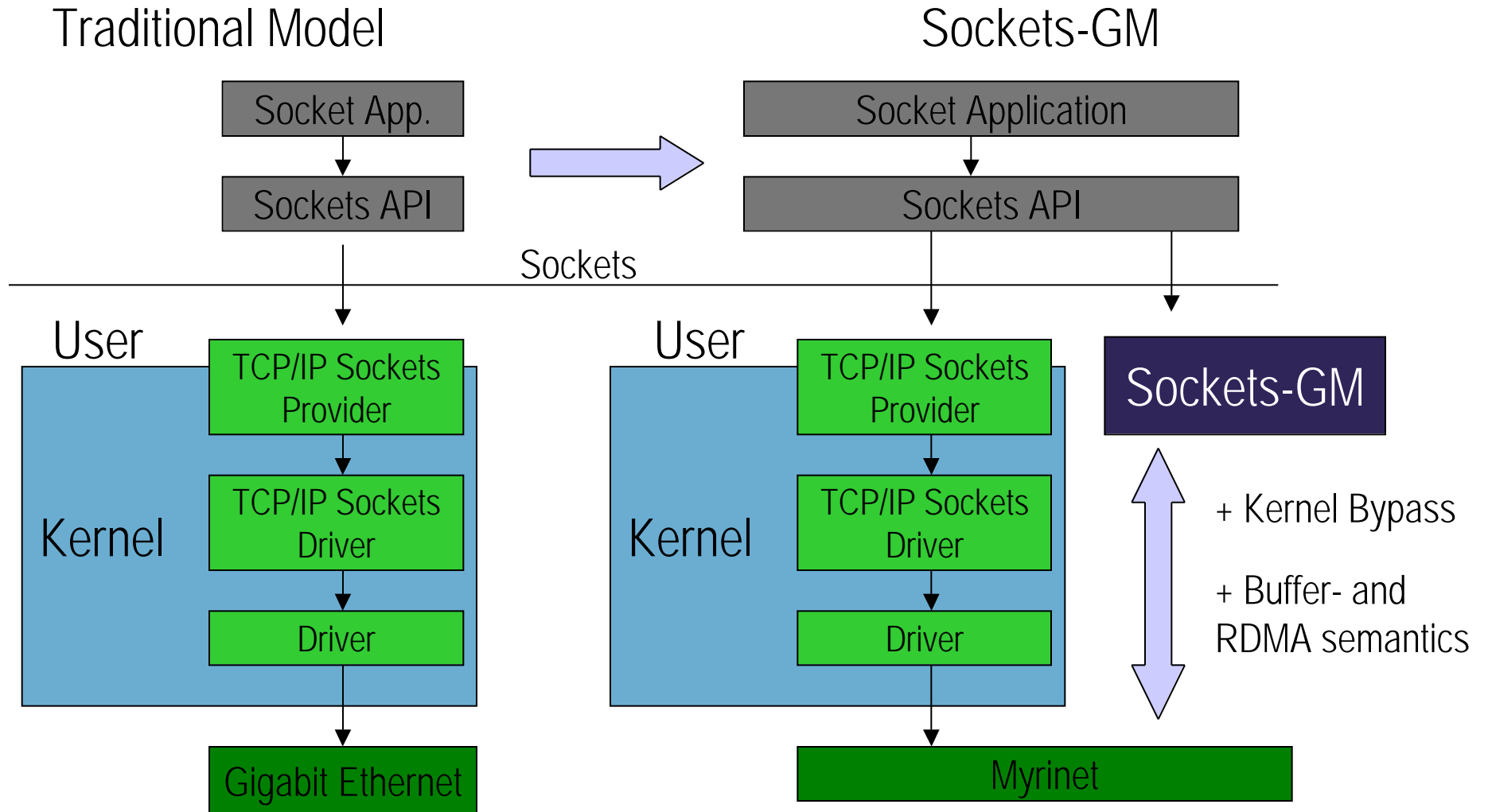
# Sockets-GM Protocols

---

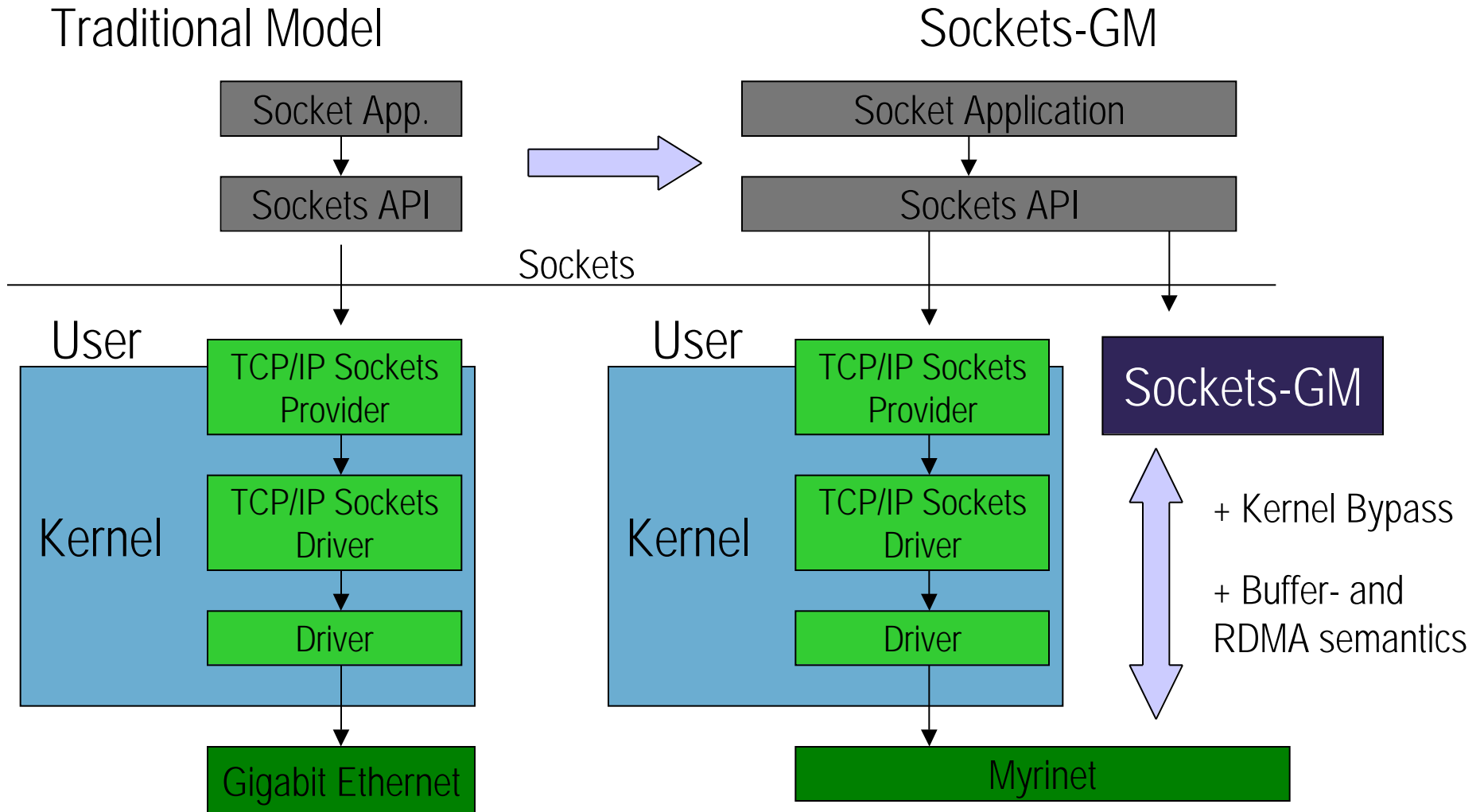
- Socket Stream semantics (TCP)
  - Dominant API for networking programming
  - SOCK\_STREAM w/ TCP error semantics
  - Abortive and Graceful close, including half closed sockets
  - Socket options
  
- Socket Datagram semantics (UDP)
  - SOCK\_DGRAM w/ UDP error semantics
  - Provide sendto, recvfrom functionality
    - determine GM properties for connectionless sendto calls
    - fill sockaddr struct in call to recvfrom
  - Socket options

Compatibility available at Binary Level !

# Sockets-GM Concept



# Sockets-GM Concept



How to intercept current linked applications to use Sockets-GM ?

# Sockets-GM Layout

---

64/32 Bit Application (Distributed, Using TCP / UDP Sockets, DCOM)

Windows NT, 2000 Prof., Server, Advanced Server, Datacenter Server

LINUX / Solaris / ...

TCP/IP - UDP/IP

GM – Myricom's Low Level API for Myrinet

Myrinet NIC

---

# Sockets-GM Layout

64/32 Bit Application (Distributed, Using TCP / UDP Sockets, DCOM)

Windows NT, 2000 Prof., Server, Advanced Server, Datacenter Server

LINUX / Solaris / ...

TCP/IP - UDP/IP



GM – Myricom's Low Level API for Myrinet

Myrinet NIC

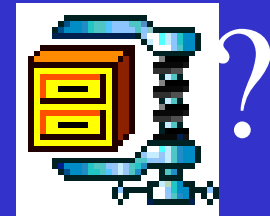
# Sockets-GM Layout

64/32 Bit Application (Distributed, Using TCP / UDP Sockets, DCOM)

Windows NT, 2000 Prof., Server, Advanced Server, Datacenter Server

LINUX / Solaris / ...

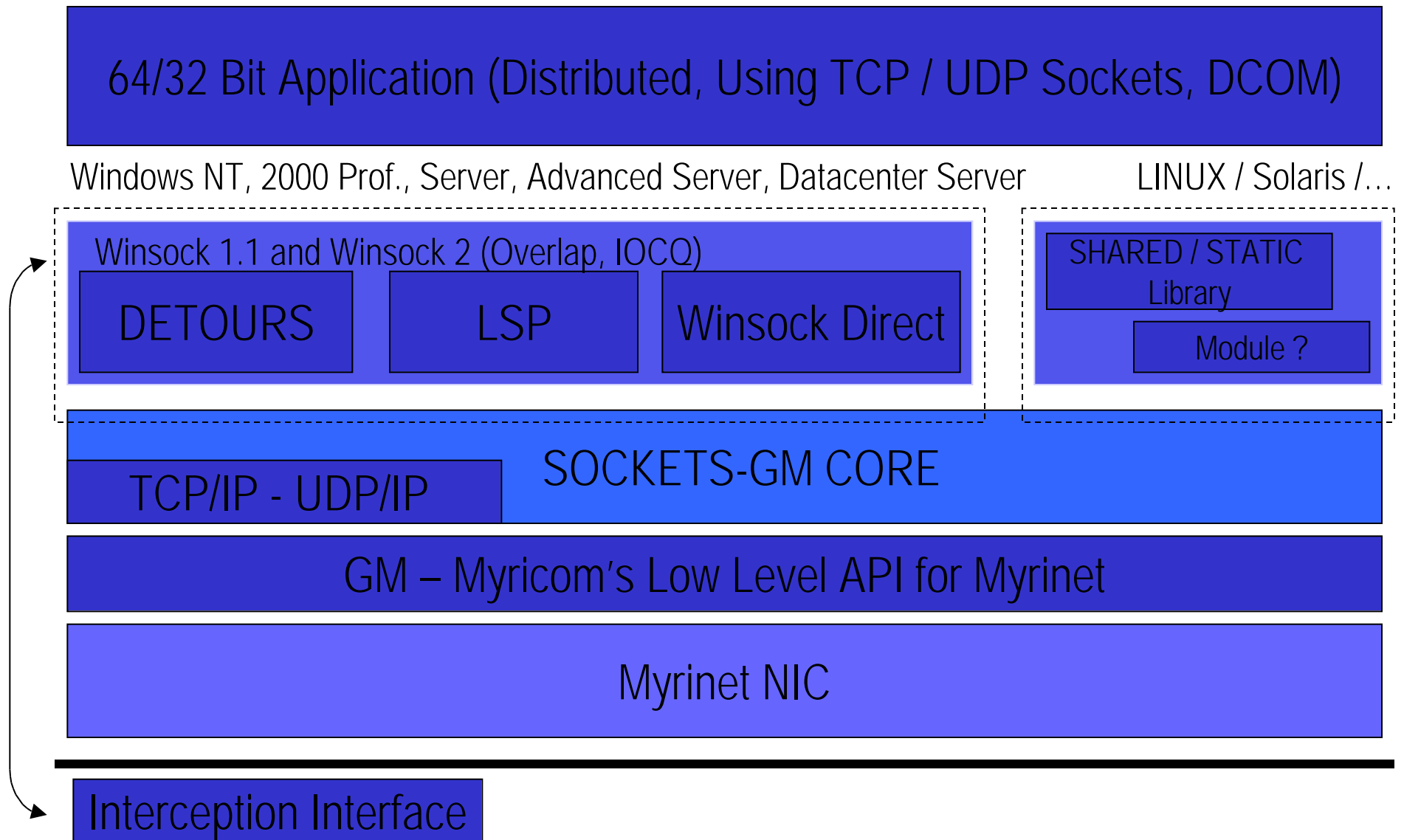
TCP/IP - UDP/IP



GM – Myricom's Low Level API for Myrinet


Myrinet NIC

# Sockets-GM Layout



# GM Functionality

---

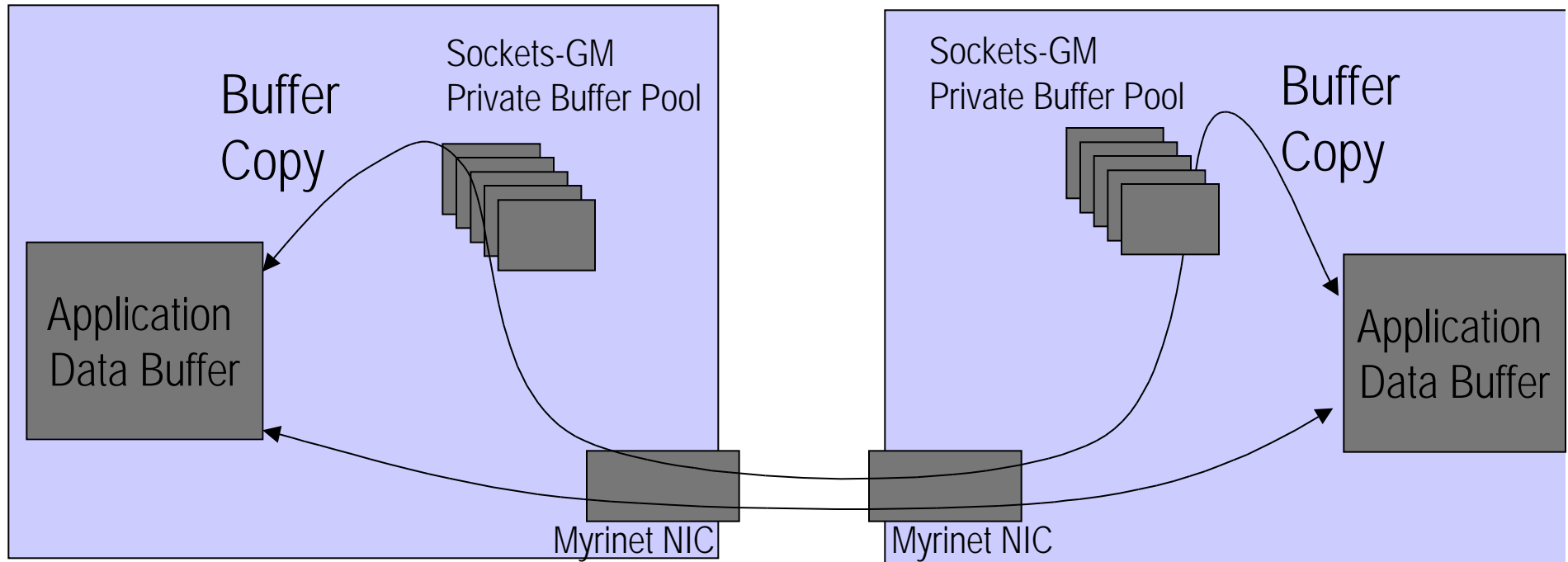
- Availability for W2K, LINUX, Solaris, ...
  - Connectionless, reliable, Point to Point Communication
  - Message Passing Stylish API
    - Non-blocking `gm_receive()`, `gm_send_with_callback ()`
  - Provides Zero Copy Methods
    - `gm_register_memory()`, `gm_directed_send()`
  - Provides get / put functionality
  - DMA Engines ideal for overlapping / CPU offloading
-  - Fits well the socket programming style !  
- Winsock 2 allows for highest efficiency

# Sockets-GM Core Implementation Details

---

- On a pt2pt connection establish companion socket with GM
- Set up information to distinguish between socket ports on the same host (sender node id), add header to payload / message
- header = identify src, identify special tags (control for EOF, shutdown, close...)
- Start Thread to synch incoming messages
- carefully look at BSD Socket / Winsock specification
  - Replacing standard TCP/IP requires fully featured protocol
  - Socket modes (blocking, non-blocking)
  - send -> returning out of send means that buffer can be re-used
  - recv -> performed on single descriptor, gm\_receive picks next message in queue
- Use Zero Copy for large messages / for offloading CPU

# Sockets-GM Data Exchange Model

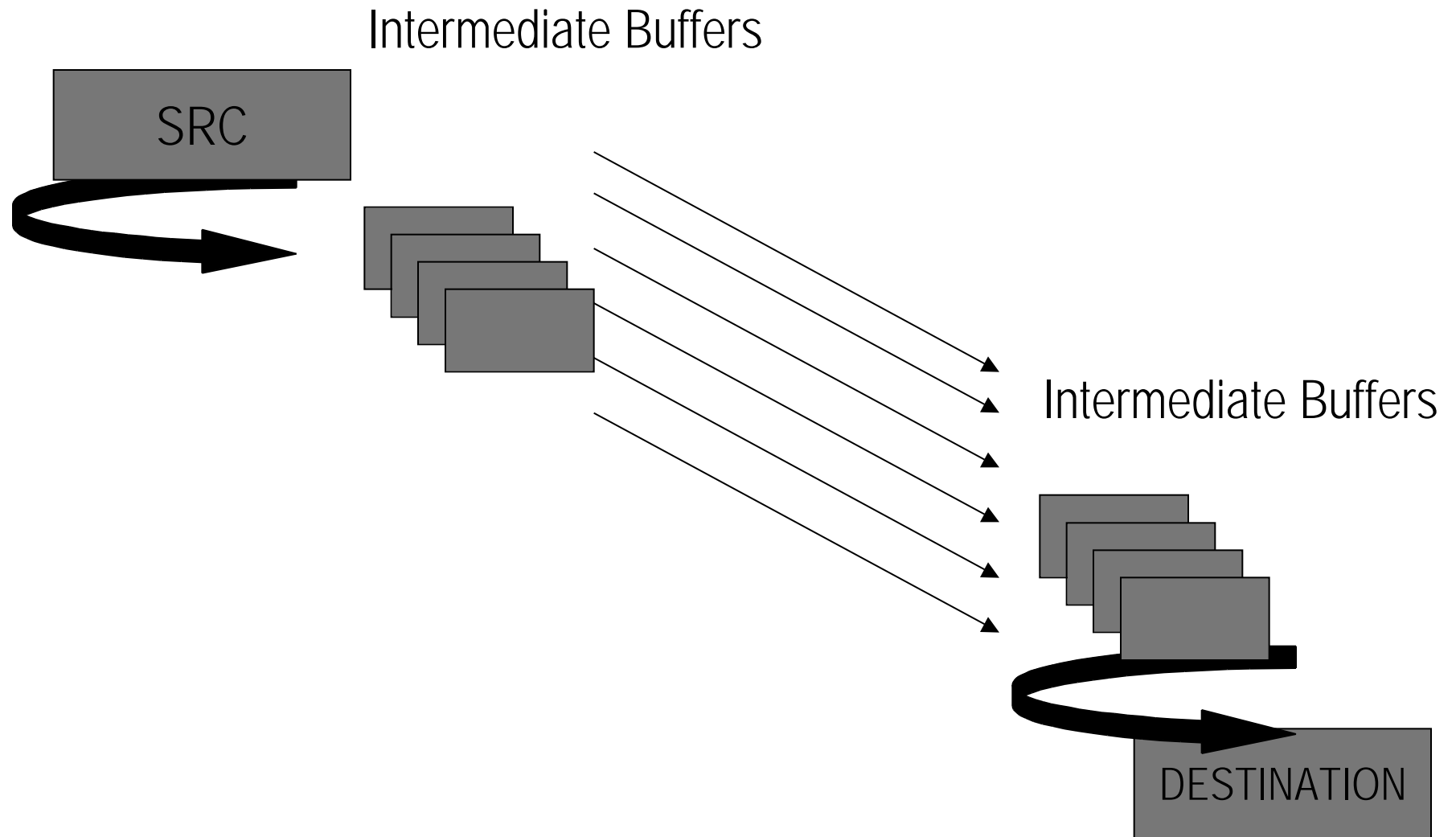


- Enables buffer-copy when
  - Transfer is short
  - Application needs buffering
- Enables zero-copy when
  - Transfer is long
  - Application uses Overlapping Functions (e.g: Winsock 2)

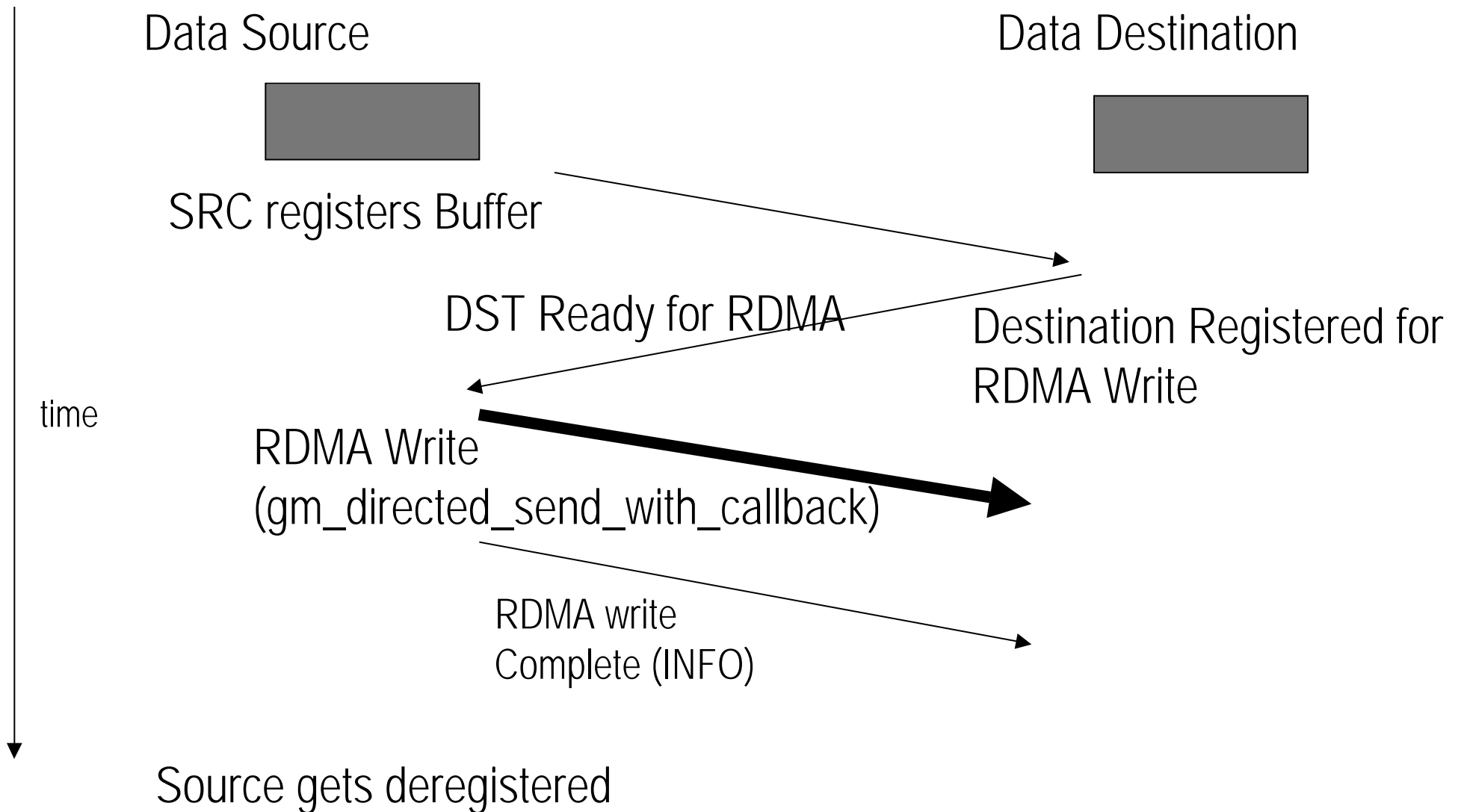
Zero Copy Protocol / RDMA

# Data Transfer Mechanisms Buffered Copy

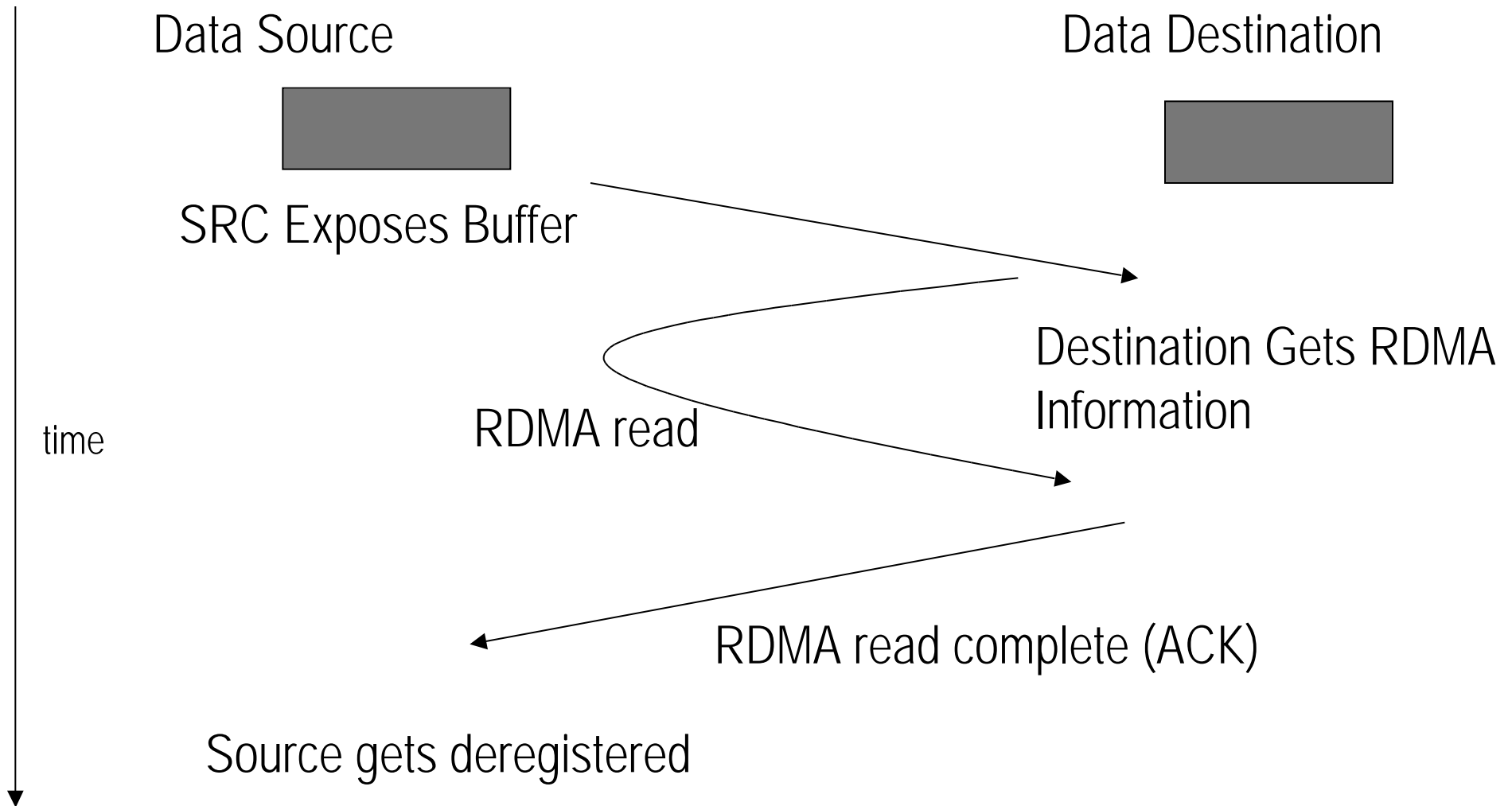
---



# Sockets-GM Data Transfer Mechanisms Write Zero Copy



# Data Transfer Mechanisms using Read Zero Copy (GM 2.0)



# Sockets-GM for BSD

---

- Operating System: Linux / Solaris
- Shared / Static Library
  - gcc and Operating System specific compiler (for example cc compiler from SUN)
  - Pre – loading shared library
    - Be aware that 32Bit applications can not load 64 library (e.g.: ls )
  - Static Library
    - Libsocketsgm.a will re-arrange symbols, linking required
- Optimization for BSD Sockets ?
  - Int send (int s, void \*buf, int len, int flags);
  - Int recv (int s, void \*buf, int len, int flags);
- Fork () issues
  - Fork() is an expensive operation !
  - Sockets are shared, who handles de-multiplexing of data ?

# Fork Issues and Sockets-GM

---

- Fork () – revert to conventional method
  - Child process gets own GM port
  - Inform connection partners (clients) about fork()
  - Child connects to clients
  - If necessary send back data to clients to be submitted again through regular mechanism
  - Flow control using Sockets-GM Tokens to guarantee correct data delivery (you can intercept fork() )
- Fork () – pass data to child using GM
  - As above, however client keeps on sending to parent until it closes its socket
  - Client maintains queue of connection partners, dequeue if connection gets closed
- Fork () – matching with MCP
  - Let MCP de-multiplex data, SRAM is limited resource

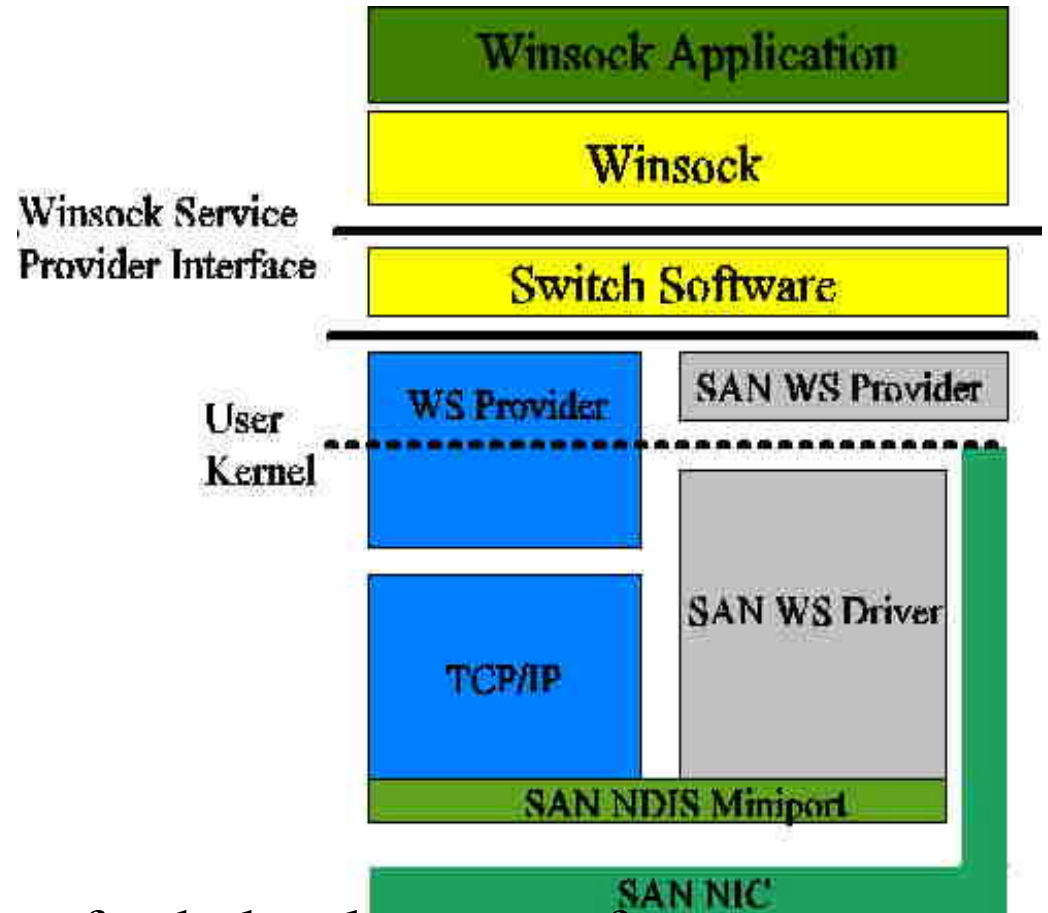
# Sockets-GM as LSP

---

- Vendor can provide a library that can be hooked into the WinSock2 DLL dynamically at run time
  - Allows for accessing services from one or more transport protocols simultaneously.
- Offers registration of new transport protocols in a protocol catalog
  - any protocol stack registered is said to be a WinSock “service provider”
- Layered Protocols are only used through the Service Provider Interface (SPI)
- Install a Transport Service Provider
  - Winsock2 architecture divides the Winsock subsystem into two general layers, the DLL providing the Winsock API and a series of service providers which plug in underneath via the SPI
- “layered protocol” relies on a base protocol for services
- SPI provides an abstraction layer
  - own communications transports/media can be provided
- Windows does not have fork ()
  - Windows has WSADuplicateSocket to be passed to a newly created Process

# Winsock Direct (SDP)

- Winsock Direct is an LSP which abstracts from different SANs.
- The counterpart, which is called by Winsock Direct has to be implemented (registration of data, message exchanges)



=> inserts an additional overhead, a fresh development of an LSP should be more efficient, since it can be optimized for a particular SAN

# Implementation Details / Winsock 1.1 and Winsock 2

---

- Winsock 1.1 like BSD Sockets
  - Int send (int s, ...)
  - Int recv (int s, ...)
- Winsock 2 Extends Winsock 1.1 Functionality By
  - Several Advanced Functions
    - WSAAcceptEx, WSAConnectEx, WSATransmitFile, ...
  - Overlapped Communication (Post Function, Query Status later)
    - WSARecv / WSASend / WSAConnect / WSAAccept/ ...
    - Modify Worker Thread to dispatch messages into posted overlapped structures
    - Let Worker Thread mimic OS features (such as Event setting)
  - IO Completion Queues
    - Very Resource Efficient, Windows specific notification mechanism

# Winsock 2 Functions For Overlapping Data Exchanges

---

int WINAPI **WSARecv** (SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD lpNumberOfBytesRecv, LPDWORD lpFlags, LPWSAOVERLAPPED lpOverlapped, LPWSAOVERLAPPED\_COMPLETION\_ROUTINE lpCompletionRoutine );

int WINAPI **WSASend** (SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD lpNumberOfBytesSent, DWORD dwFlags, LPWSAOVERLAPPED lpOverlapped, LPWSAOVERLAPPED\_COMPLETION\_ROUTINE lpCompletionRoutine);

bool WINAPI **WSAGetOverlappedResult** (SOCKET s, LPWSAOVERLAPPED lpOverlapped, LPDWORD lpcbTransfer, BOOL fWait, LPDWORD lpdwFlags);

bool WINAPI **TransmitFile** (SOCKET hSocket, HANDLE hFile, DWORD nNumberOfBytesToWrite, DWORD nNumberOfBytesPerSend, LPOVERLAPPED lpOverlapped, LPTRANSMIT\_FILE\_BUFFERS lpTransmitBuffers, DWORD dwFlags );

# Winsock 2 Overlapped Example

---

```
if ((sock = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0,
WSA_FLAG_OVERLAPPED)) == INVALID_SOCKET) {
    // HANDLE ERROR
    return 0;
}
// ACCEPT / CONNECT
olapEvent = CreateEvent (NULL, FALSE, FALSE, NULL);

recvBuffers = (LPWSABUF) malloc(sizeof(WSABUF) * numBufs);
for(i=0; i < numBufs; i++) {
    recvBuffers[i].len = pktSize;
    recvBuffers[i].buf = (char *) malloc(pktSize);
}
```

## Winsock 2 Overlapped Example (cont'd)

---

```
recvBytes = 0; flags = 0;
if (WSARecv(new_sock, recvBuffers, numBufs, &recvBytes, &flags, &olapStruct, NULL) ==
SOCKET_ERROR) {
    if ((retErr = WSAGetLastError()) == WSA_IO_PENDING) {
        olapFlags = 0;
        if (WSAGetOverlappedResult(new_sock, &olapStruct, &xferBytes,
TRUE, &olapFlags)) {
            /* data received thru overlapped i/o */
            ResetEvent(olapEvent);
            if (xferBytes != (DWORD) pktSize*numBufs)
                }
        } else { // other error than WSA_IO_PENDING
            /* free receive buffers, ERROR Handling */
            exit(0);
        }
    }
}
```

---

# **PCI 64 Bit / 66 Mhz Performance Measurements**

**Myrinet 2000, Windows 2000**

**Supermicro 370DLE, PIII 1 Ghz**  
(455 MB/s bus read, 512 MB/s bus write)

# Sockets-GM Performance

---

- The following slides depict the performance comparison of applications using
  - a Winsock compliant SOCKETS-GM for Windows 2000 (TCP/IP & UDP/IP)
  - a BSD compliant SOCKETS-GM for LINUX/Solaris/UNICES (TCP/IP & UPD/IP)
  - a TCP/IP and UDP/IP using GigEth **SysKonnnect 9821**
  - a TCP/IP over Myrinet
    - The TCP/IP stack has been tuned with various parameters as transport
- Applications are
  - netpipe (ping ping, ping pong mode)
  - netperf (ping ping)
  - Iperf
  - NTttcp
  - and some additional perf test (ping pong) for UDP/IP and TCP/IP
- Comparison with
  - GM Raw Performance (~maximal performance)

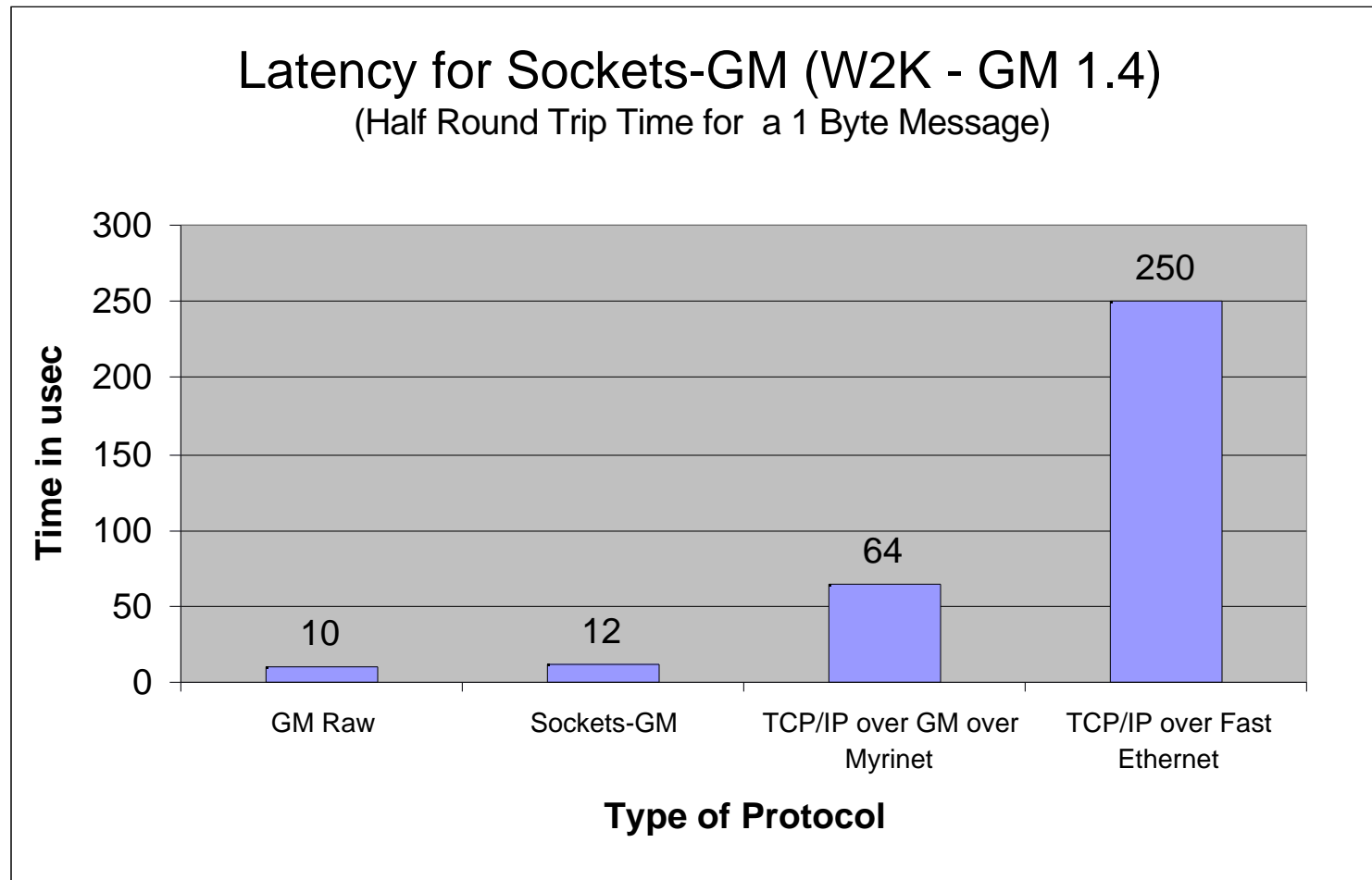
# TCP/IP Tuning for W2K

---

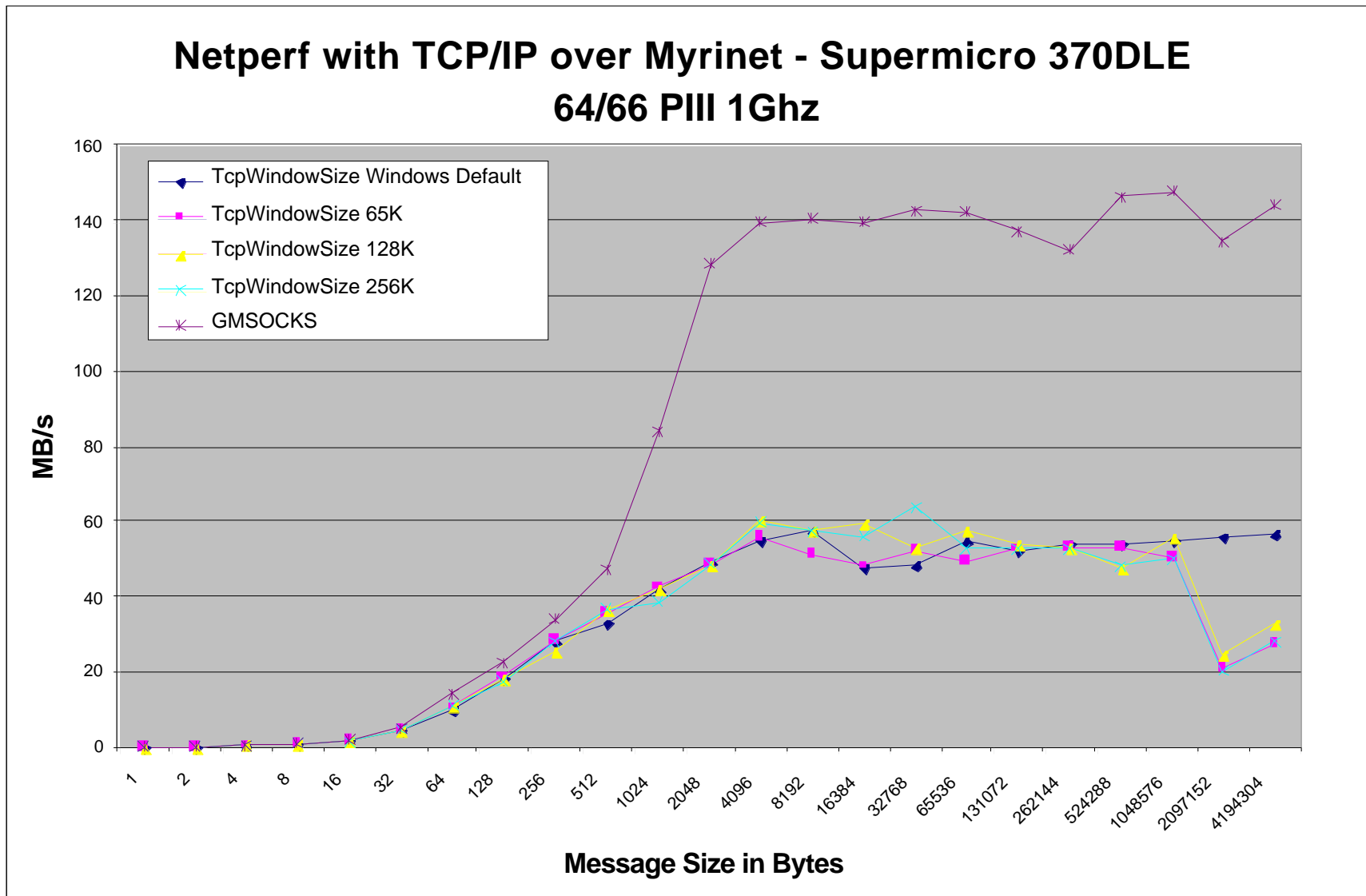
- Shareware ? : EasyMTU, TuneMTU, ...
  - poor performance, even: there goes your TCP/IP functionality
- W2K has tuning build in
  - Bottom Line: You can't do better
- Registry Keys under Windows
  - TcpWindowSize
  - MTU
  - ...
- Benchmarks do have options to tune TCP/IP (setsockopt), however existing applications don't

oh, yes, the change of a registry value requires a REBOOT ...

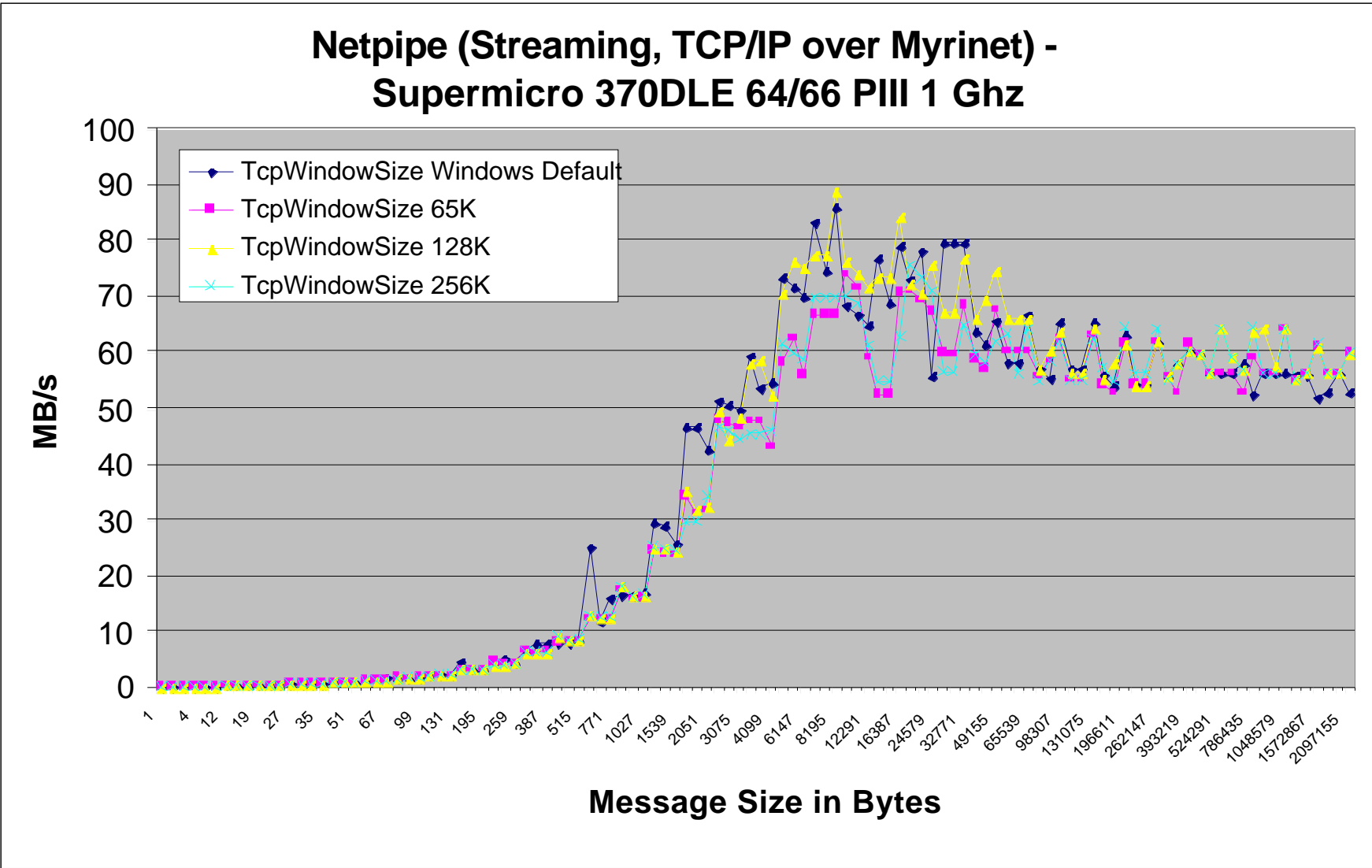
# Latency Comparison



# Netperf with TCP/IP over Myrinet vs Sockets-GM

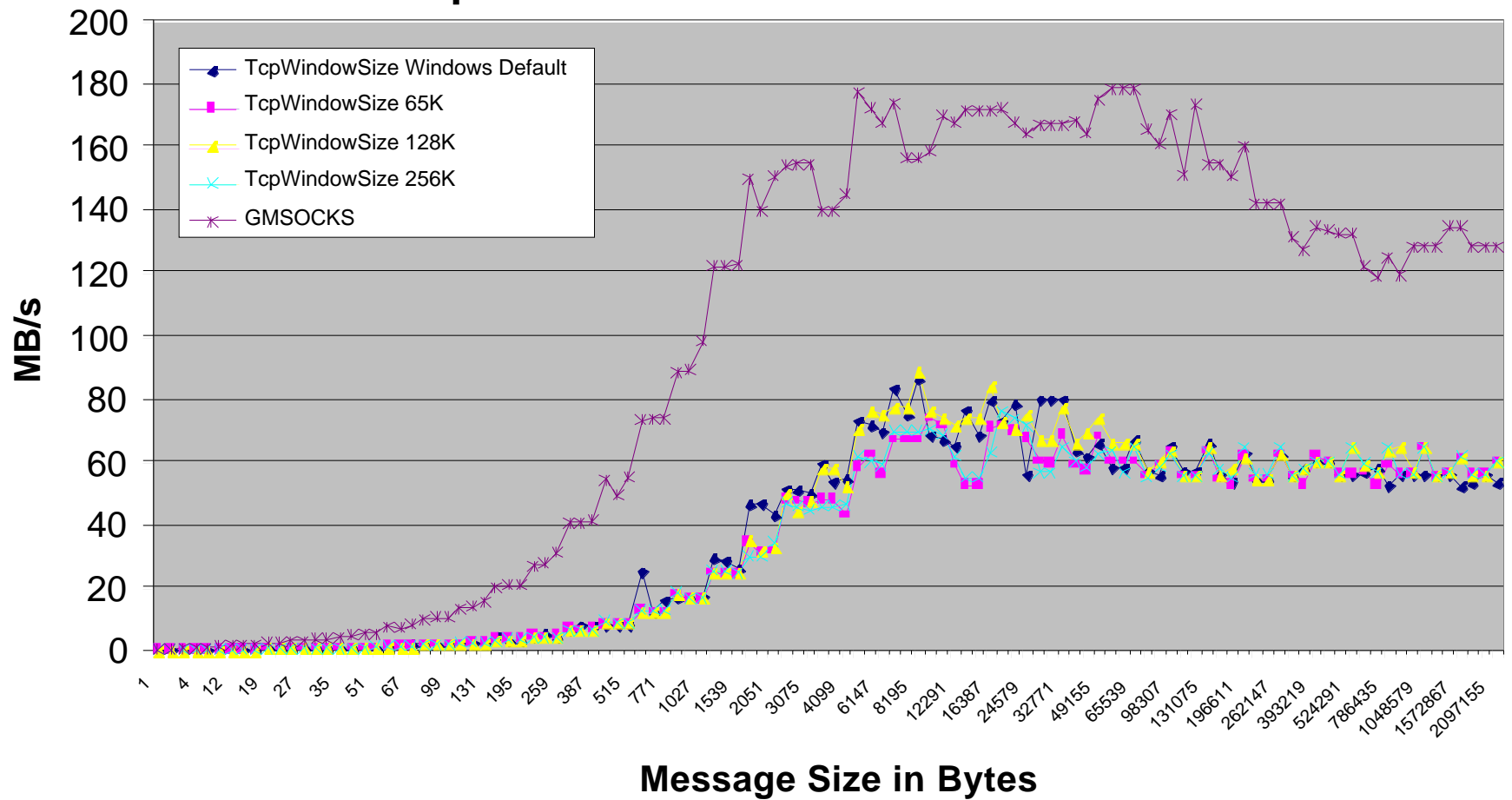


# Netpipe (Streaming)



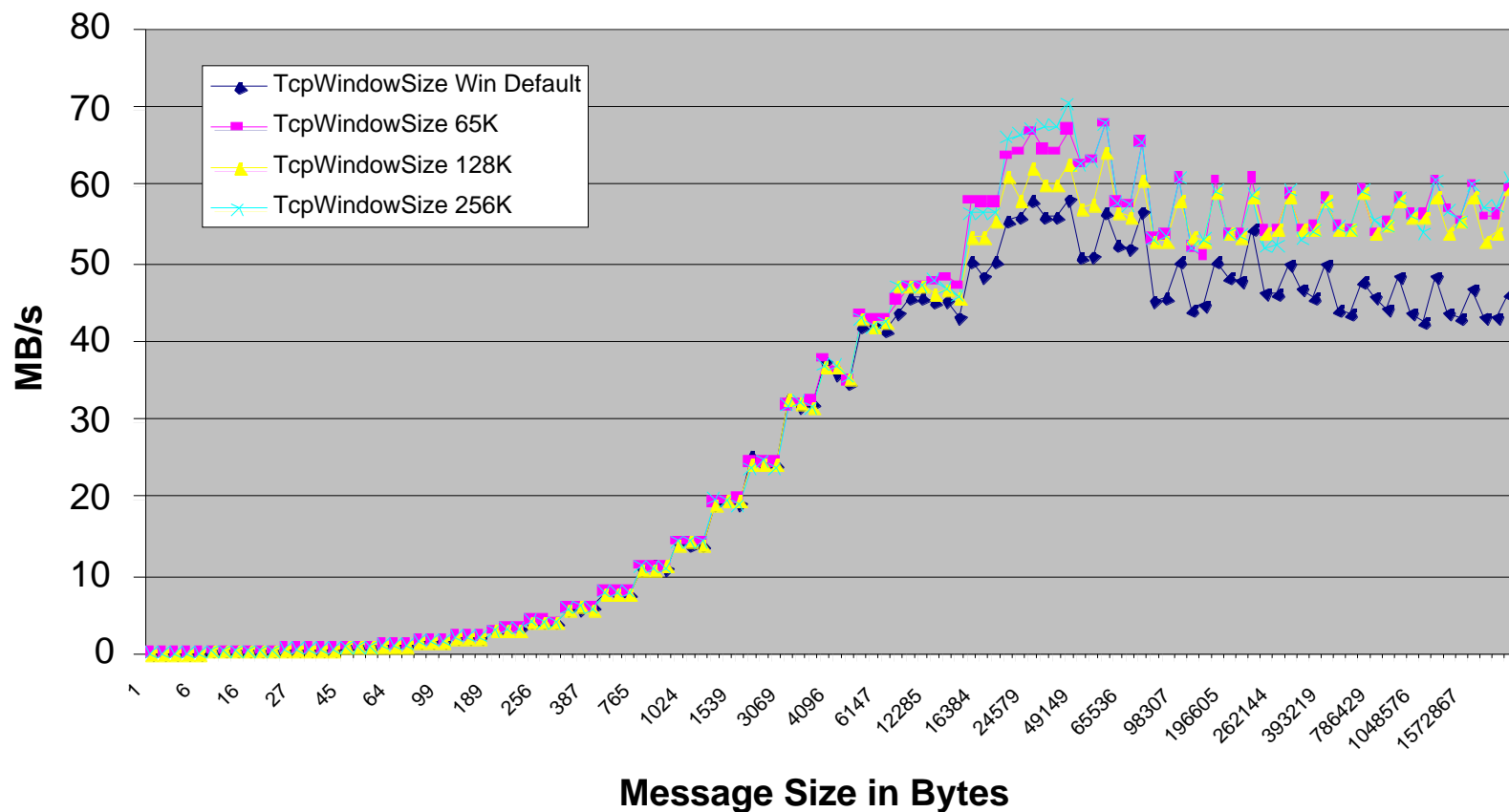
# Netpipe (Streaming)

**Netpipe (Streaming, TCP/IP over Myrinet vs GMSOCKS) -  
Supermicro 370DLE 64/66 PIII 1 Ghz**



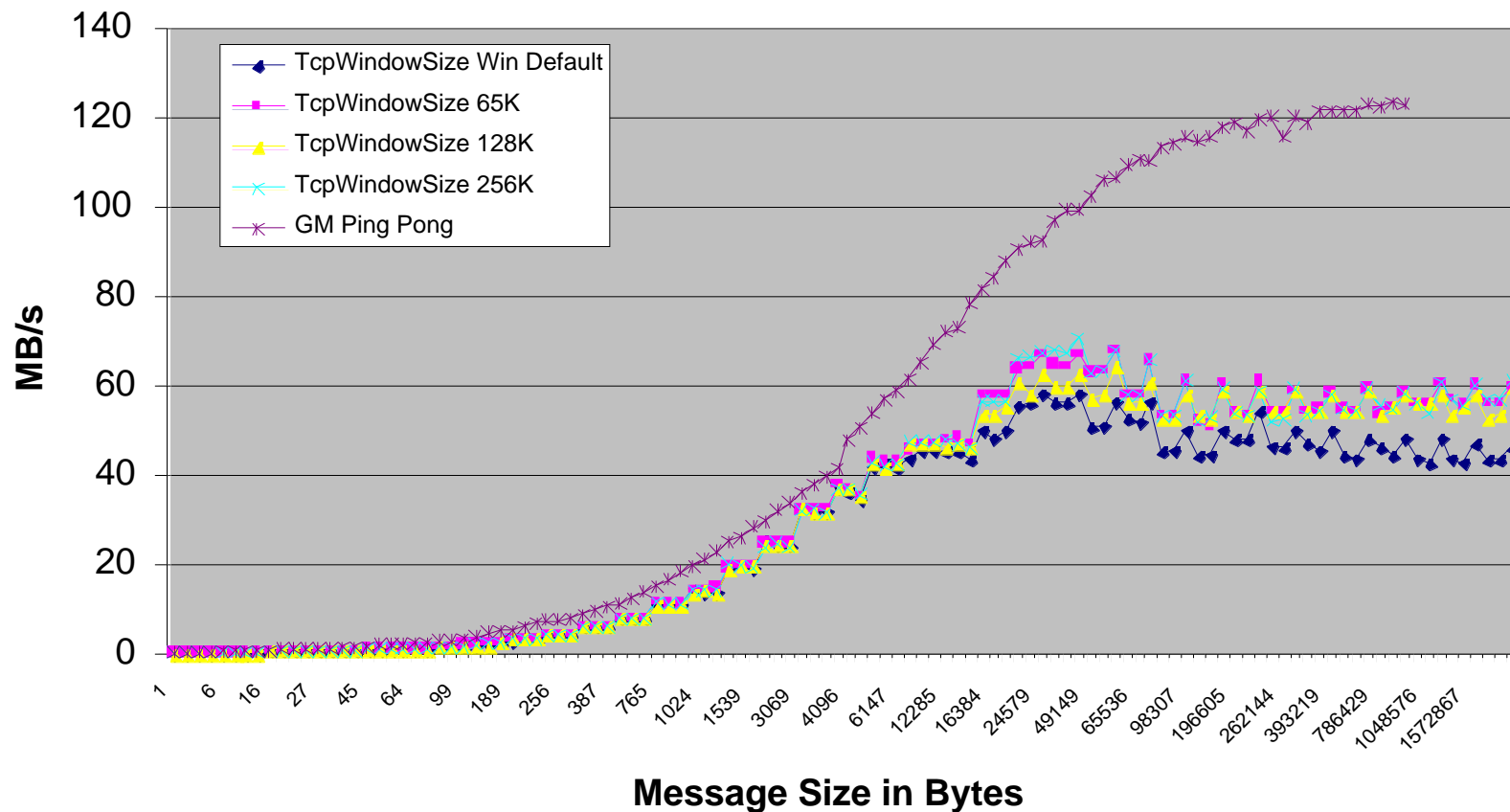
# Netpipe (Ping Pong, TCP/IP over Myrinet)

Netpipe (Ping Pong) with TCP/IP over Myrinet -  
Supermicro 370DLE, PCI 64/66, PIII 1Ghz

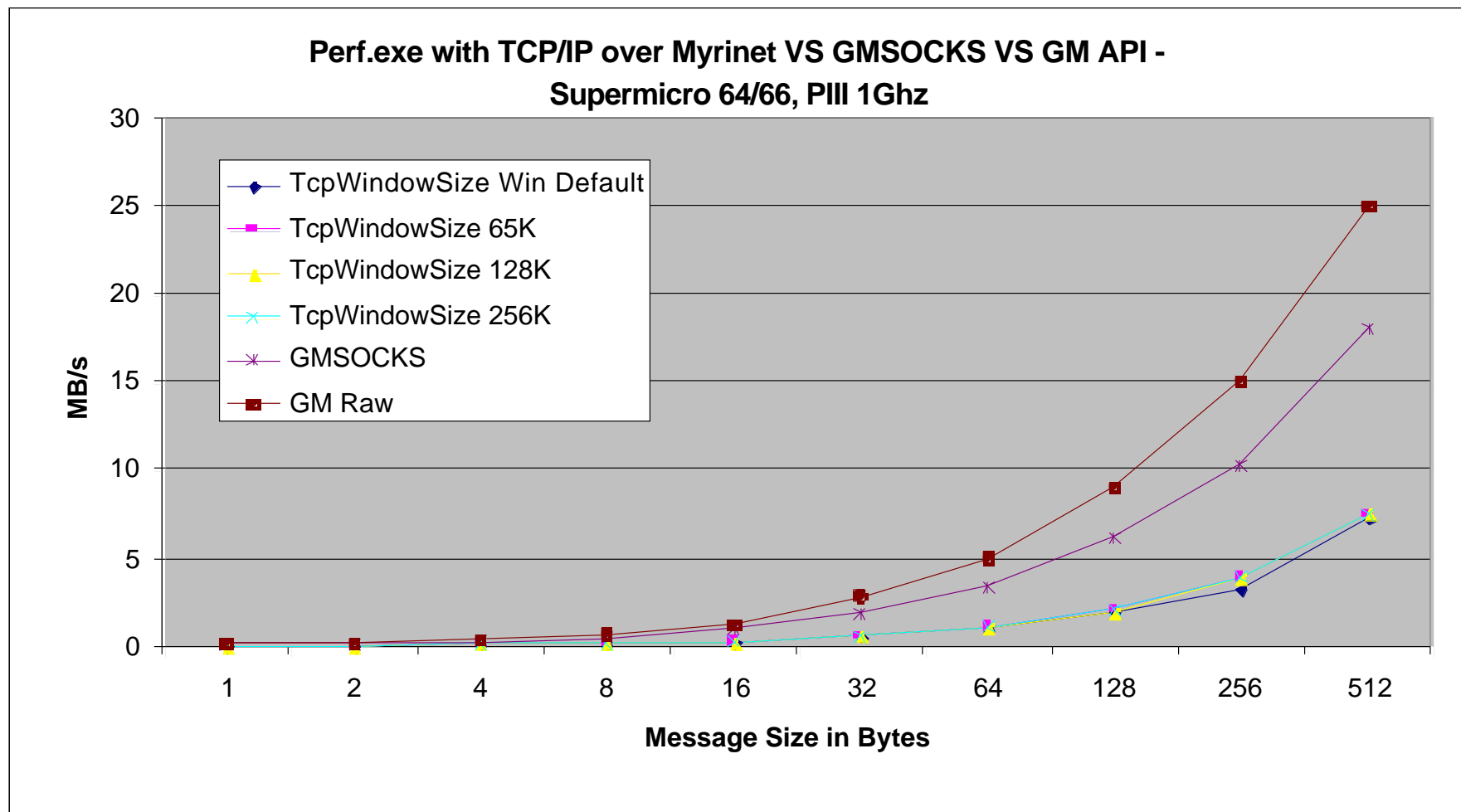


# Netpipe (Ping Pong, TCP/IP over Myrinet VS GM)

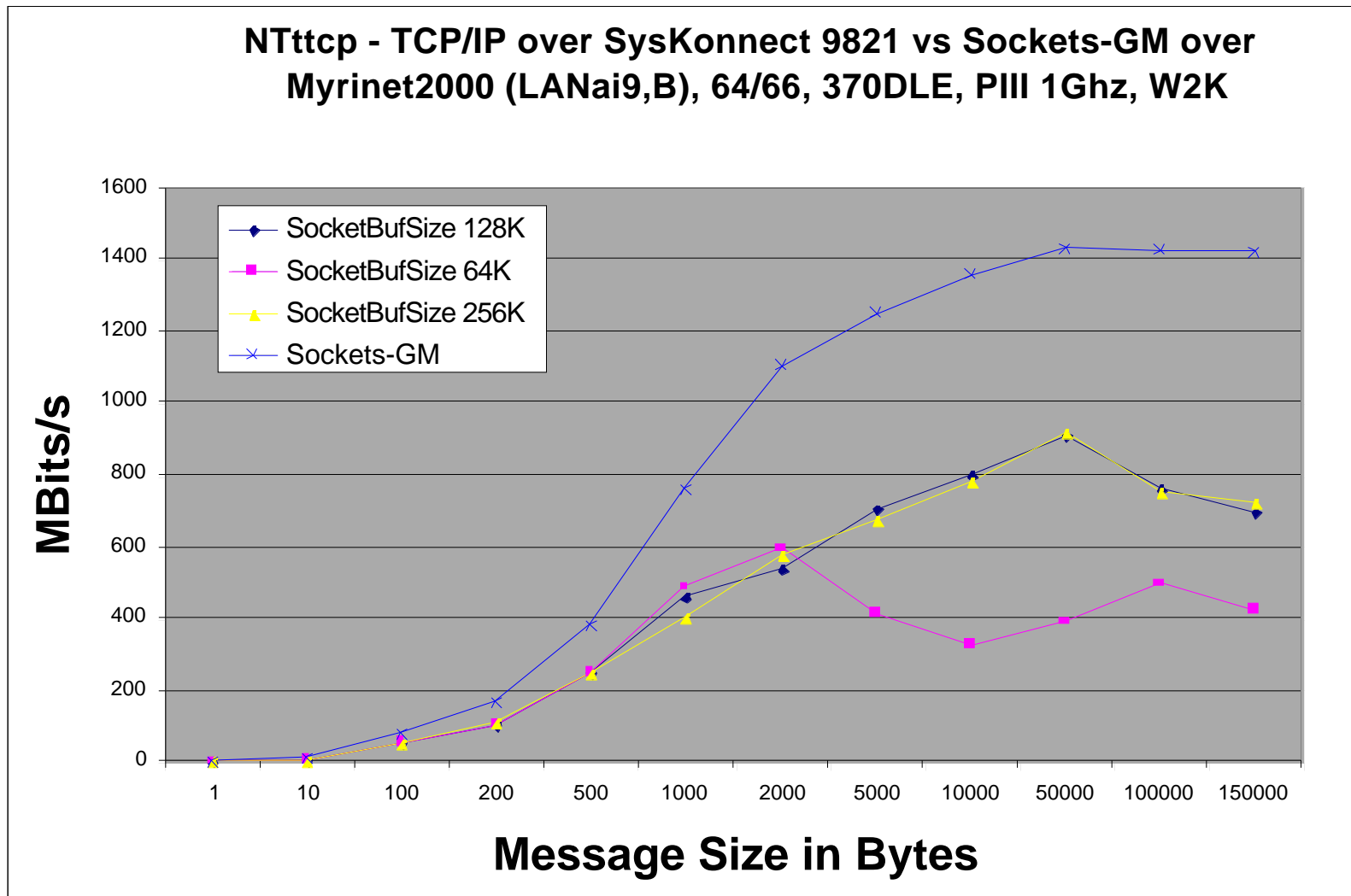
## Netpipe (Ping Pong) with TCP/IP over Myrinet VS GM (Ping Pong) - Supermicro 370DLE, PCI 64/66, PIII 1Ghz



# Perf.exe with TCP/IP over Myrinet vs Sockets-GM vs GM API

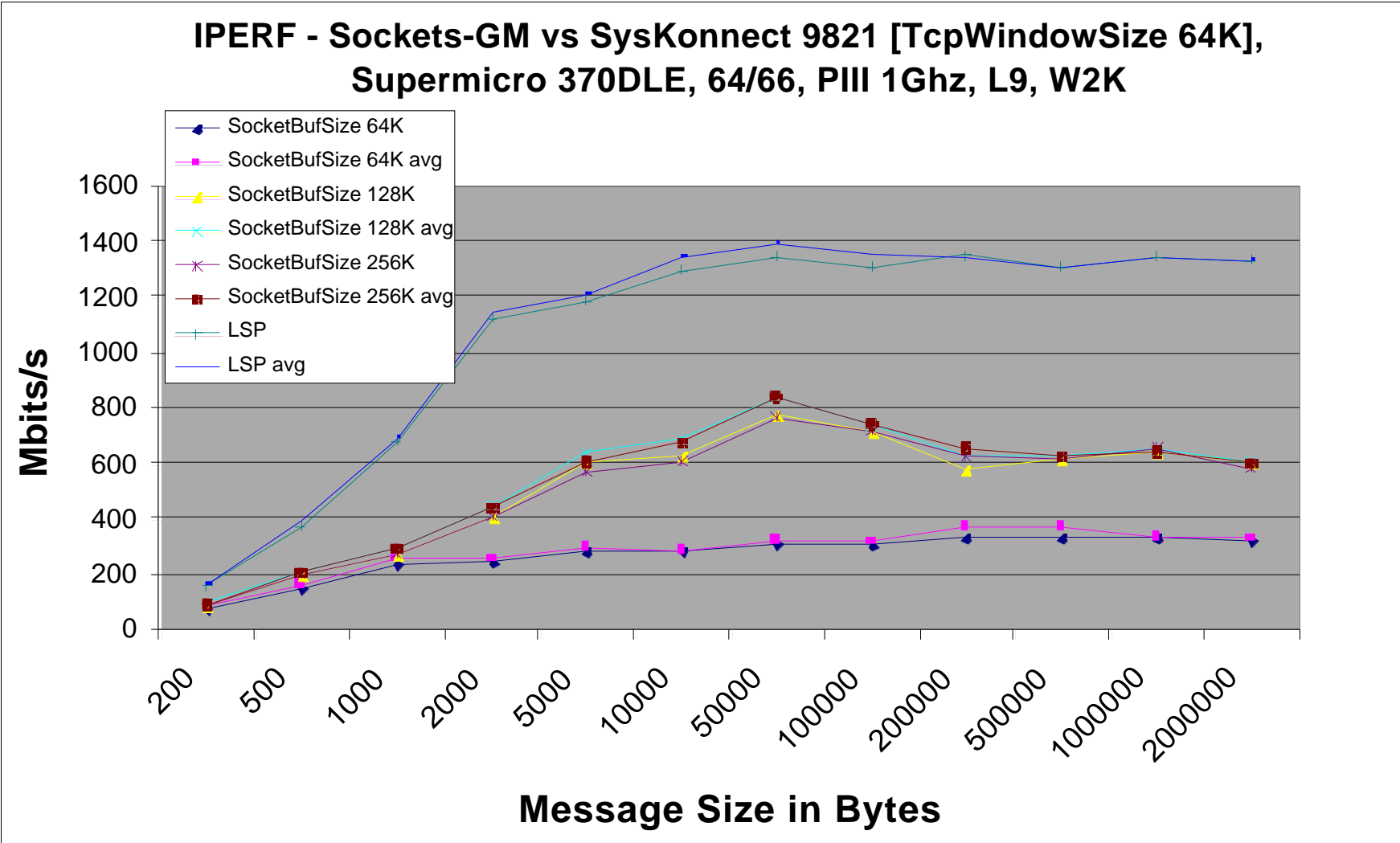


## NTttcp with TCP/IP over SysKonnct [TcpWindowSize 65K] vs SOCKETS-GM

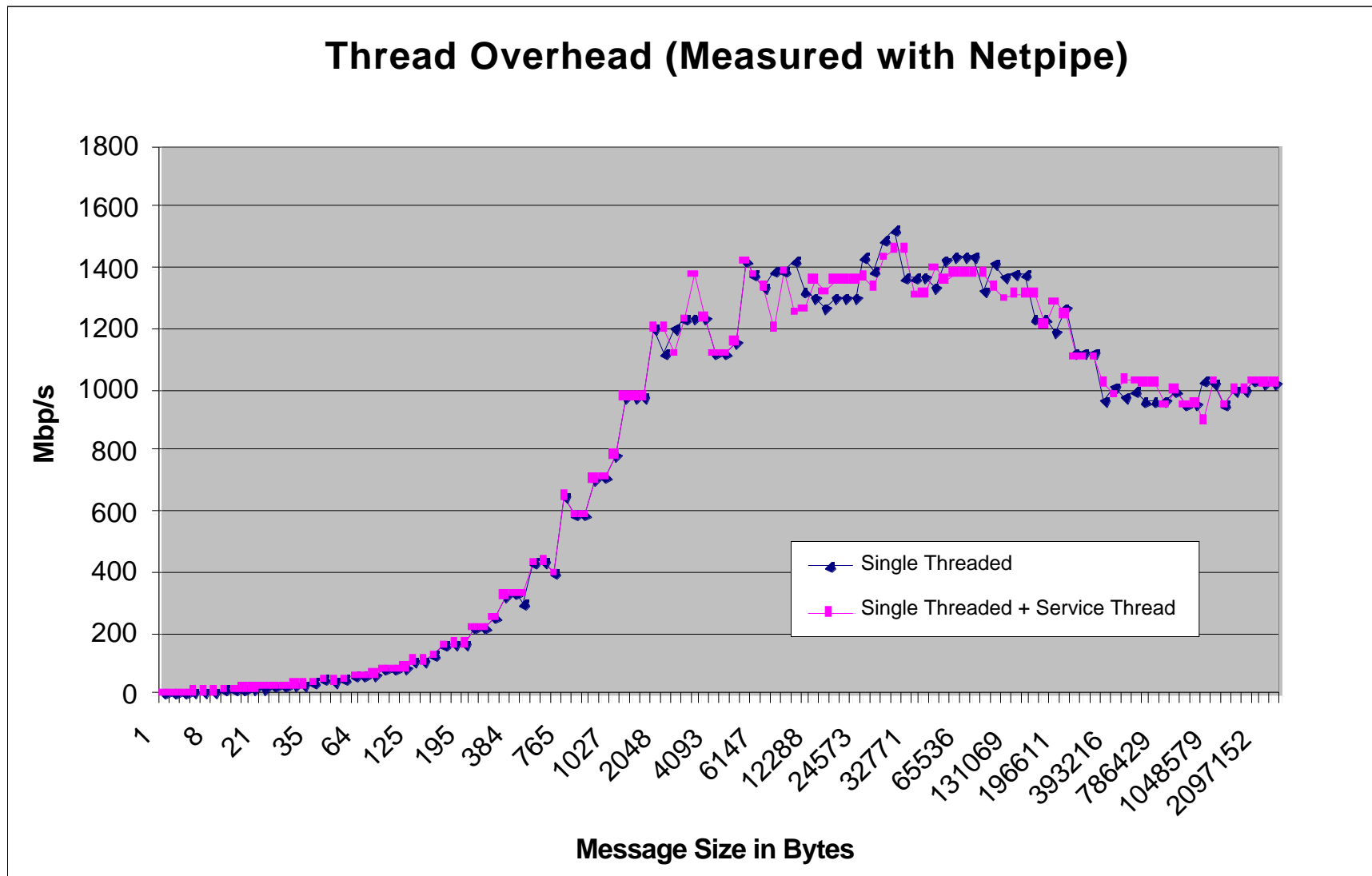


Will you be able to auto-tune your socket buffers for TCP/IP ?

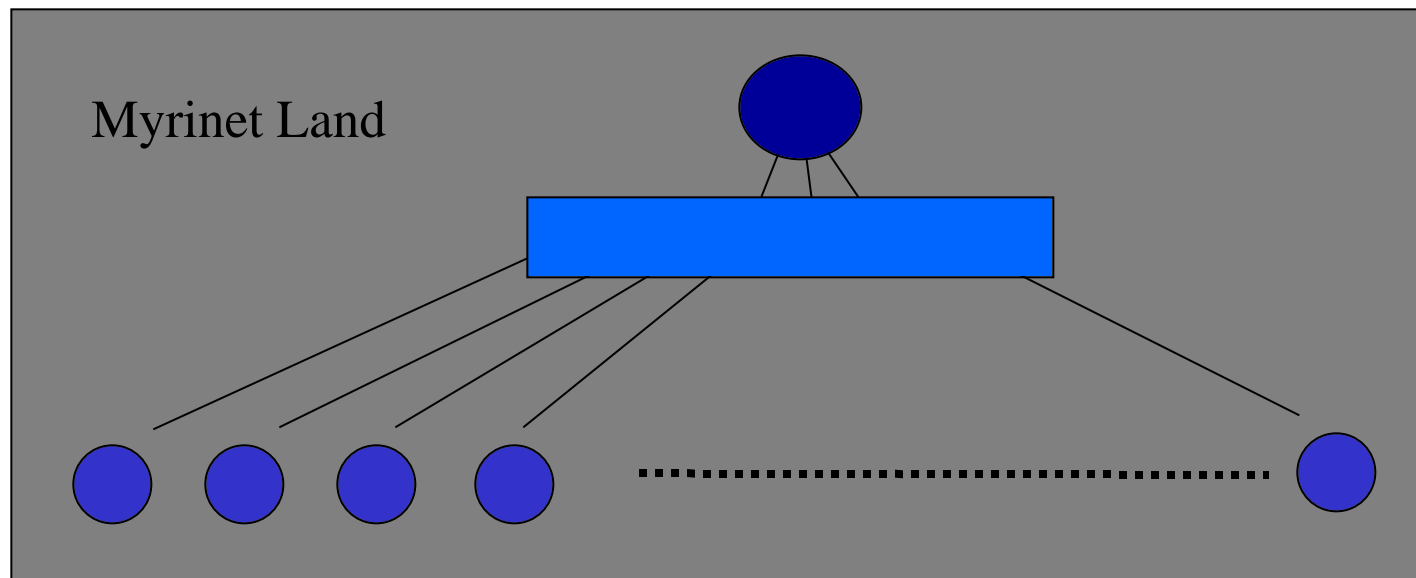
# Iperf Performance



# Thread Overhead (Single CPU System)

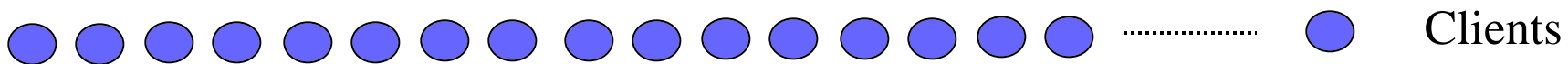


# A real world application



Database Server  
# nodes  $\geq 1$

Application Server  
# nodes  $\geq 30$  (50)



- Speed up Number of Transactions

# Conclusion & Outlook

---

- Sockets-GM running under
  - Linux/Solaris as Shared / Static Library
  - Windows NT / 2000 / XP and XP IA64 as LSP (full support of handling overlapping operations as well as IO completion queues)
- Sockets-GM offers Binary Compatibility !
  - Comes also with static library if you want to link. Linking automatically re-maps socket functions
- Winsock Direct [SDP] (work in progress)
- Continue efforts to offload the CPU ( << 5 % )
  - Work in Progress to provide switch to use polling for lowest latency and highest bandwidth, or use zero copy strategies to free CPU – with higher latency and less bandwidth
  - Winsock 2 (and Sockets-GM LSP) provide the required interfaces to have maximum performance with very low CPU utilization