



# System-Area Networks (SAN) Group

---

*Distributed Shared-Memory Parallel  
Computing with UPC on SAN-based Clusters*

Dr. Alan D. George, Director  
HCS Research Laboratory  
University of Florida

---

# Outline

- Objectives and Motivations
- Background
- Related Research
- Approach
- Preliminary Results
- Conclusions and Future Plans

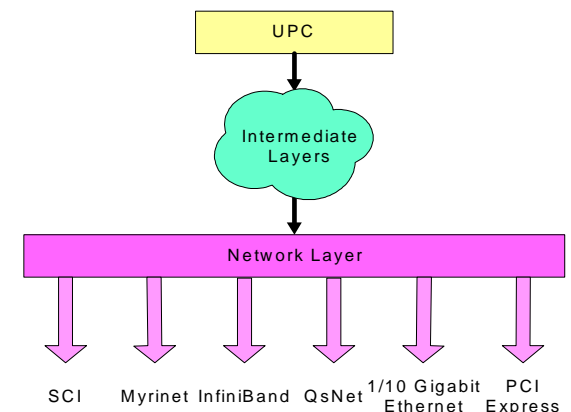
# Objectives and Motivations

## ■ Objectives

- ❑ Support advancements for HPC with Unified Parallel C (UPC) on cluster systems exploiting high-throughput, low-latency system-area networks (SANs)
- ❑ Design and analysis of tools to support UPC on SAN-based systems
- ❑ Benchmarking and case studies with key UPC applications
- ❑ Analysis of tradeoffs in application, network, service and system design

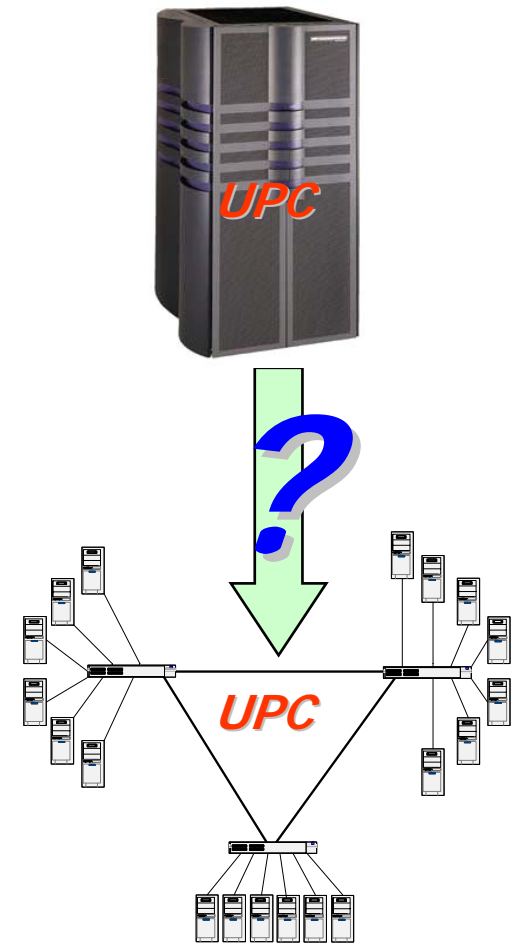
## ■ Motivations

- ❑ Increasing demand in sponsor and scientific computing community for shared-memory parallel computing with UPC
- ❑ New and emerging technologies in system-area networking and cluster computing
  - Scalable Coherent Interface (SCI)
  - Myrinet
  - InfiniBand
  - QsNet (Quadrics)
  - Gigabit Ethernet and 10 Gigabit Ethernet
  - PCI Express (3GIO)
- ❑ Clusters offer excellent cost-performance potential



# Background

- Key sponsor applications and developments toward shared-memory parallel computing with UPC
  - More details from sponsor are requested
- UPC extends the C language to exploit parallelism
  - Currently runs best on shared-memory multiprocessors (notably HP/Compaq's UPC compiler)
  - First-generation UPC runtime systems becoming available for clusters (MuPC, Berkeley UPC)
- Significant potential advantage in cost-performance ratio with COTS-based cluster configurations
  - Leverage economy of scale
  - Clusters exhibit low cost relative to tightly-coupled SMP, CC-NUMA, and MPP systems
  - Scalable performance with commercial off-the-shelf (COTS) technologies



# Related Research

## ■ University of California at Berkeley

- ❑ UPC runtime system
- ❑ UPC to C translator
- ❑ Global-Address Space Networking (GASNet) design and development
- ❑ Application benchmarks



## ■ George Washington University

- ❑ UPC specification
- ❑ UPC documentation
- ❑ UPC testing strategies, testing suites
- ❑ UPC benchmarking
- ❑ UPC collective communications
- ❑ Parallel I/O



## ■ Michigan Tech University **MichiganTech.**

- ❑ Michigan Tech UPC (MuPC) design and development
- ❑ UPC collective communications
- ❑ Memory model research
- ❑ Programmability studies
- ❑ Test suite development

## ■ Ohio State University

- ❑ UPC benchmarking



## ■ HP/Compaq

- ❑ UPC compiler



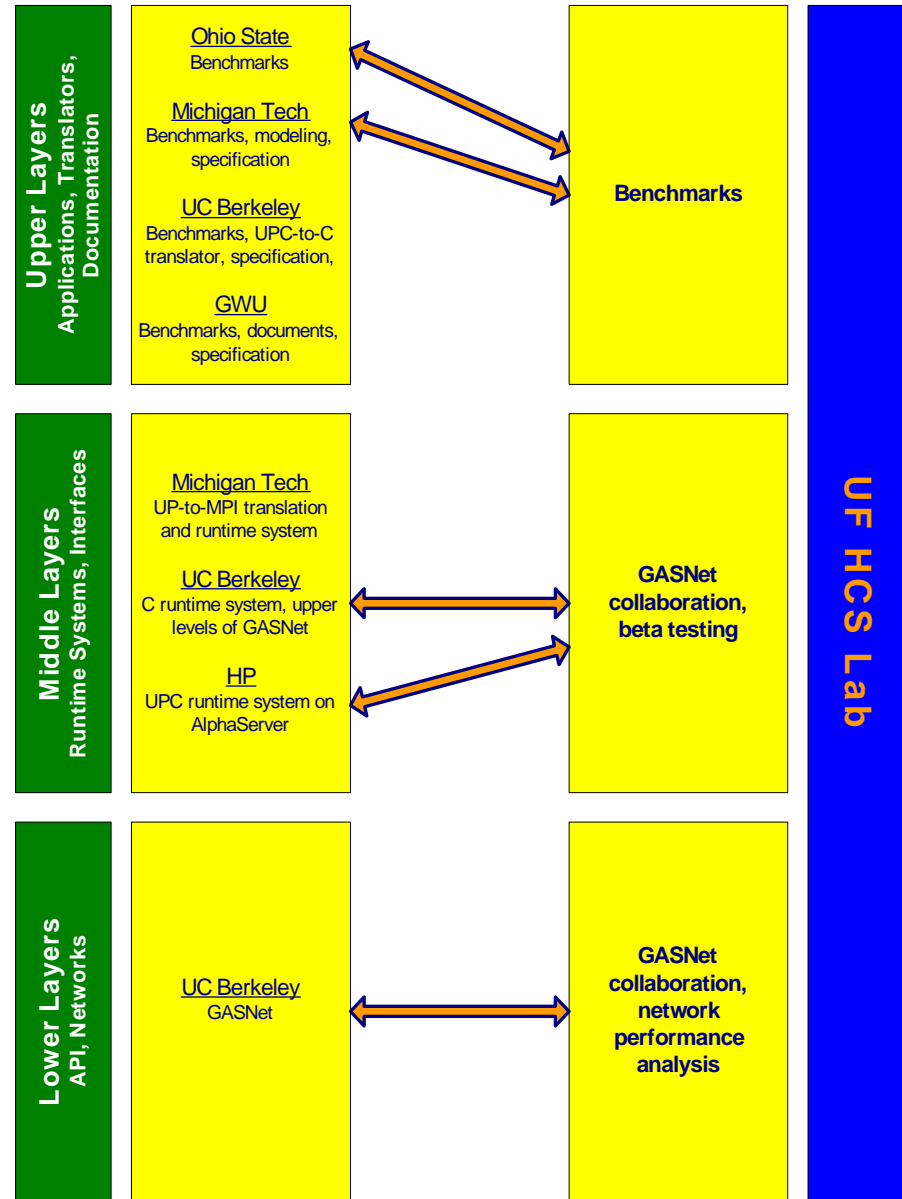
## ■ Intrepid

- ❑ GCC UPC compiler



# Approach

- Collaboration
  - HP/Compaq UPC Compiler V2.1 running in lab on new ES80 AlphaServer (Marvel)
  - Support of testing by OSU, MTU, UCB/LBNL, UF, et al. with leading UPC tools and system for function performance evaluation
  - Field test of newest compiler and system
- Benchmarking
  - Use and design of applications in UPC to grasp key concepts and understand performance issues
- Exploiting SAN Strengths for UPC
  - Design and develop new SCI Conduit for GASNet in collaboration UCB/LBNL
  - Evaluate DSM for SCI as option of executing UPC
- Performance Analysis
  - Network communication experiments
  - UPC computing experiments
- Emphasis on SAN Options and Tradeoffs
  - SCI, Myrinet, InfiniBand, Quadrics, GigE, 10GigE, etc.



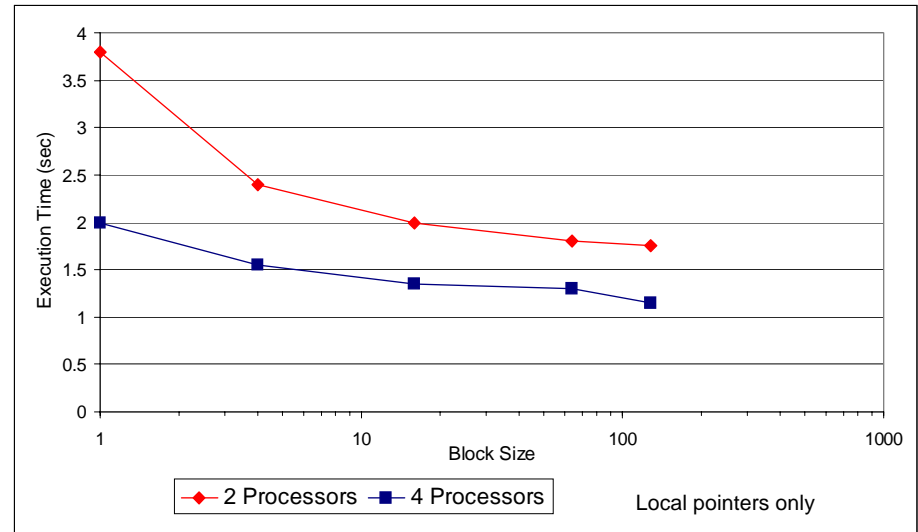
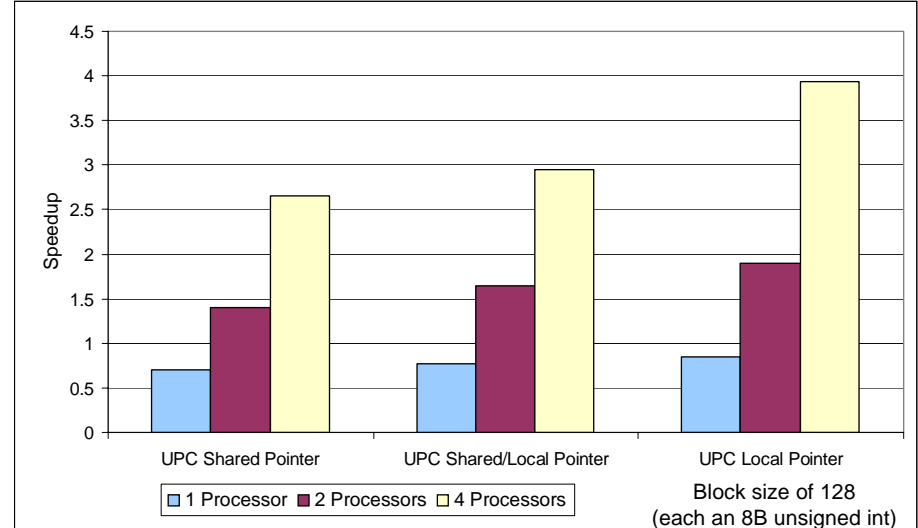
# UPC Benchmarks

- Test bed
  - ES80 “Marvel” AlphaServer, 4 1.0GHz Alpha EV7 CPUs, 8GB RAM, Tru64 V5.1b, UPC compiler V2.1, C compiler V6.5, and MPI V1.96 from HP/Compaq
- Benchmarking
  - Analyzed overhead and potential of UPC
  - STREAM
    - Sustained memory bandwidth benchmark
    - Measures memory access time with and without processing
      - Copy  $a[i] = b[i]$ , Scale  $a[i]=c*b[i]$ , Add  $a[i]=b[i]+c[i]$ , Triad  $a[i]=d*b[i]+c[i]$
      - Implemented in UPC, ANSI C, and MPI
        - Comparative memory access performance analysis
      - Used to understand memory access performance for various memory configurations in UPC
  - Differential attack simulator of S-DES (10-bit key) cipher\*
    - Creates basic components used in differential cryptanalysis
      - S-Boxes, Difference Pair Tables (DPT), and Differential Distribution Tables (DDT)
    - Implemented in UPC to expose parallelism in DPT and DDT creation
      - Various access methods to shared memory
        - Shared pointers, shared block pointers
        - Shared pointers cast as local pointers
        - Various block sizes for distributing data across all nodes

# UPC Benchmarks

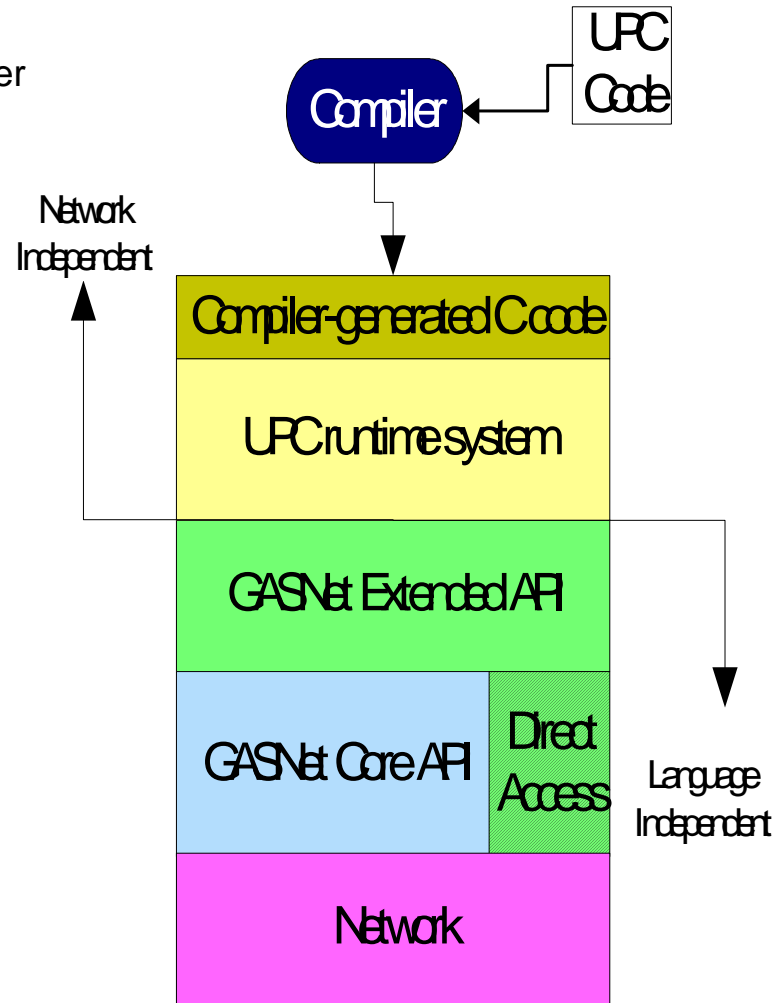
## ■ Analysis

- Sequential UPC is as fast as ANSI C
  - No extra UPC overhead with local variables
- Casting shared pointer to local pointers to access shared data in UPC
  - Only one major address translation per block
  - Decreases data access time
  - Parallel speedup (compared to sequential) becomes almost linear
  - Reduces overhead introduced by UPC shared variables
- Larger block sizes increase performance in UPC
  - Fewer overall address translations when used with local pointers



# GASNet/SCI

- Berkeley UPC runtime system operates over GASNet layer
  - Comm. APIs for implementing global-address-space SPMD languages
    - Network and language independent
    - Two layers
      - Core (Active messages)
      - Extended (Shared-memory interface to networks)
- SCI Conduit for GASNet
  - Implements GASNet on SCI network
  - Core API implementation nearly completed via Dolphin SISCO API
    - Starts with Active Messages on SCI
    - Uses best aspects of SCI for higher performance
      - PIO for small, DMA for large, writes instead of reads
- GASNet Performance Analysis on SANs
  - Evaluate GASNet and Berkeley UPC runtime on a variety of networks
  - Network tradeoffs and performance analysis through
    - Benchmarks
    - Applications
    - Comparison with data obtained from UPC on shared-memory architectures (performance, cost, scalability)



# GASNet/SCI - Core API Design

## ■ 3 types of shared-memory regions

### □ Control (ConReg)

- Stores: Message Ready Flags (MRF,  $N \times 4$  Total) + Message Exist Flag (MEF, 1 Total)
- SCI Operation: Direct shared memory write  
(\*x = value)

### □ Command (ComReg)

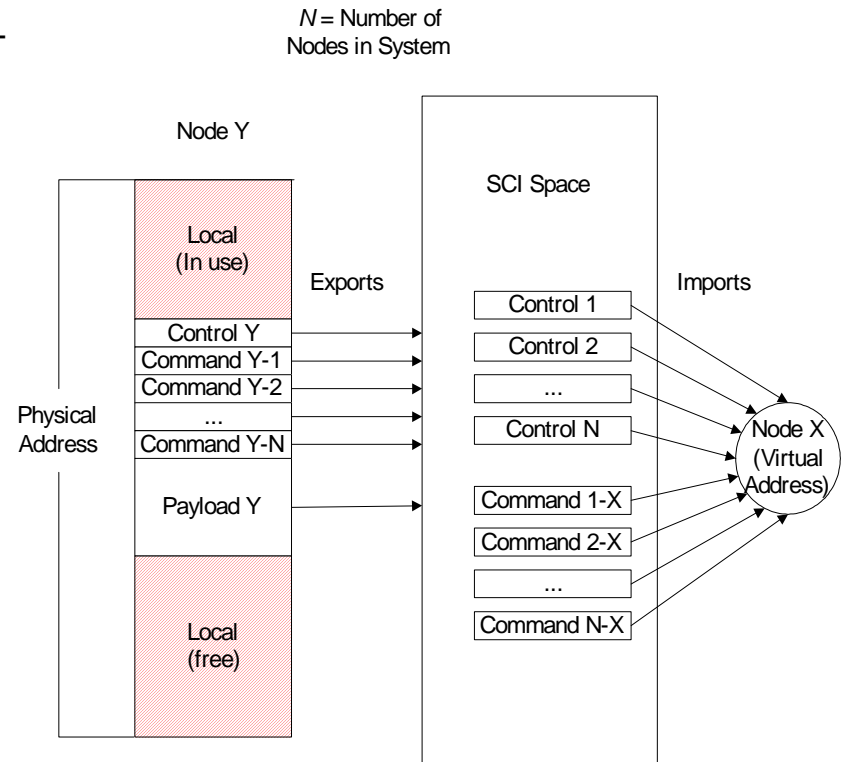
- Stores: 2 pairs of request/reply message
  - AM Header (80B)
  - Medium AM payload (up to 944B)
- SCI Operation: memcpy() write

### □ Payload (PayReg) – Long AM payload

- Stores: Long AM payload
- SCI Operation: DMA write

## ■ Initialization phase

- All nodes export 1 ConReg,  $N$  ComReg and 1 PayReg
- All nodes import  $N$  ConReg and  $N$  ComReg
- Payload regions not imported in prelim. version



# GASNet/SCI - Core API Design

## Execution phase

- Sender sends a message through 3 types of shared-memory regions

### Short AM

- Step 1: Send AM Header
- Step 2: Set appropriate MRF and MEF to 1

### Medium AM

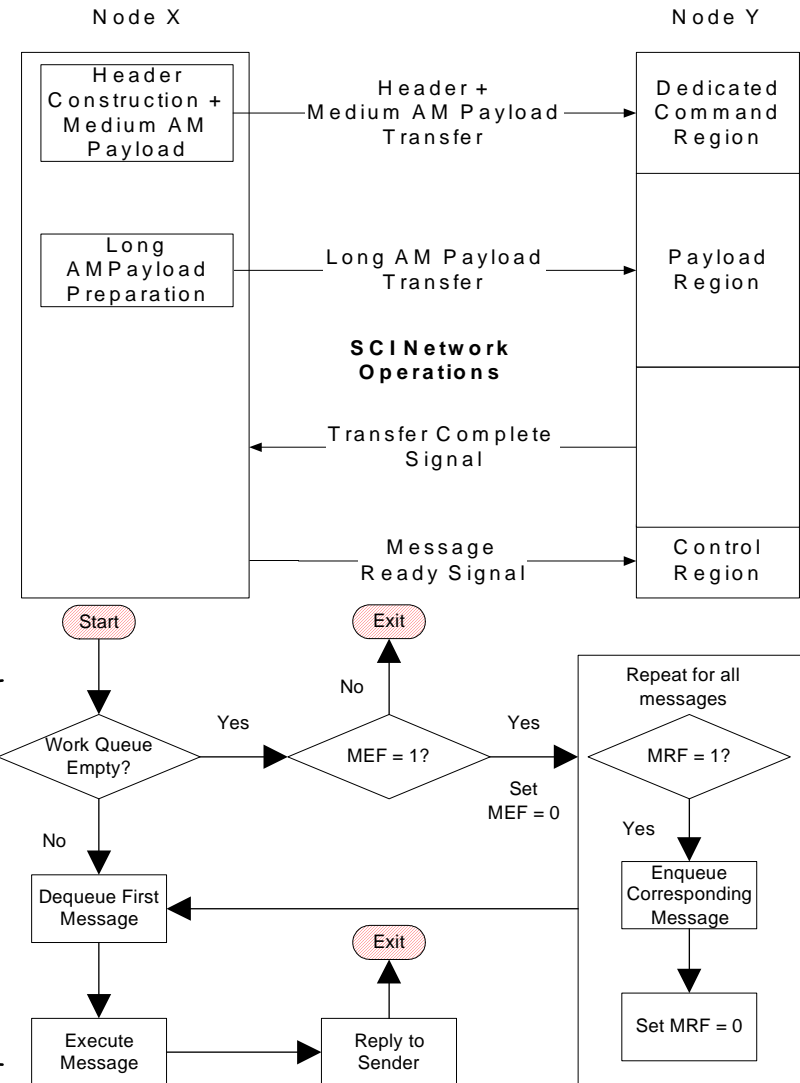
- Step 1: Send AM Header + Medium AM payload
- Step 2: Set appropriate MRF and MEF to 1

### Long AM

- Step 1: Imports destination's payload region, prepare source's payload data
- Step 2: Send Long AM payload + AM Header
- Step 3: Set appropriate MRF and MEF to 1

- Receiver polls new message by

- Step 1: Check work queue. If not empty, skip to step 4
- Step 2: If work queue is empty, check if MEF = 1
- Step 3: if MEF = 1, set MEF = 0, check MRFs and enqueue corresponding messages, set MRF = 0
- Step 4: Dequeue new message from work queue
- Step 5: Read from ComReg and executes new message
- Step 6: Send reply to sender



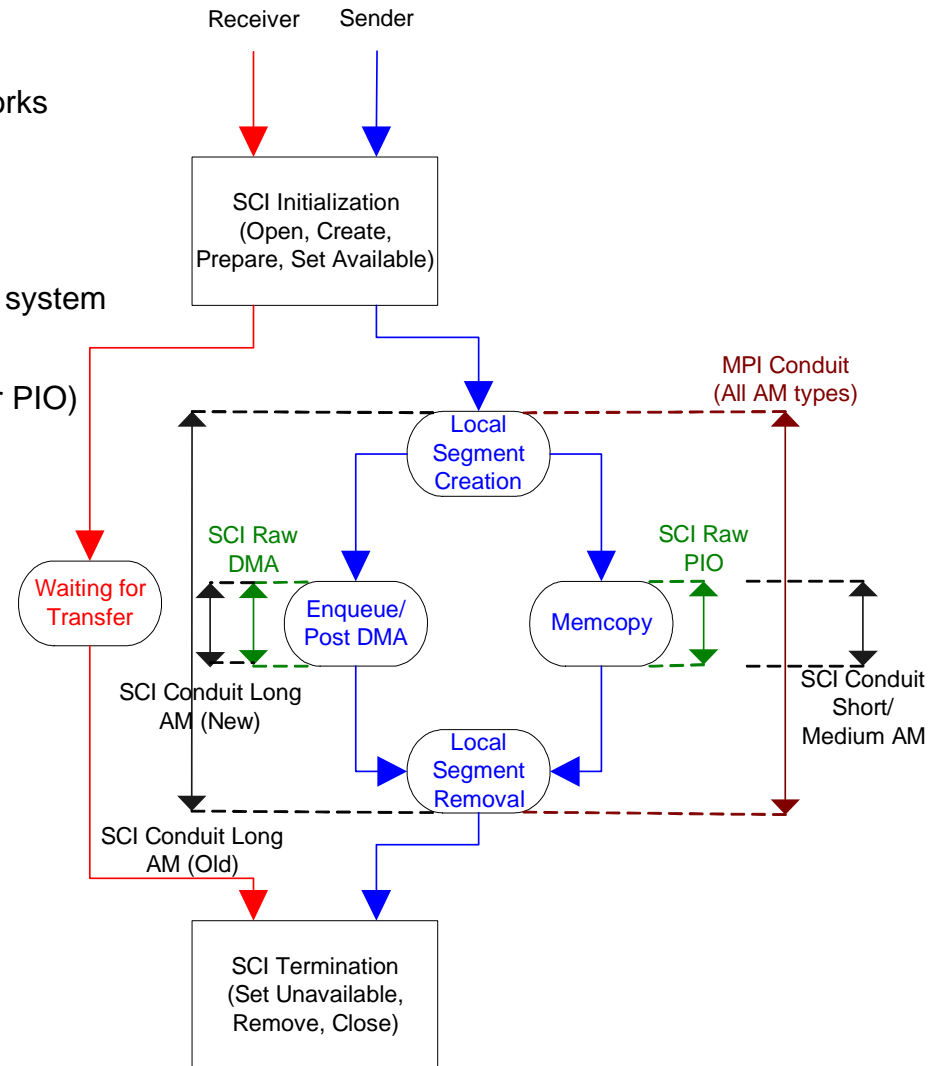
# GASNet/SCI - Experimental Setup (Short/Medium/Long AM)

## Testbed

- Dual 1.0 GHz Pentium-III's, 256MB PC133, ServerWorks CNB20LE Host Bridge (rev. 6)
- 5.3 Gb/s Dolphin SCI D339 NICs, 2x2 torus

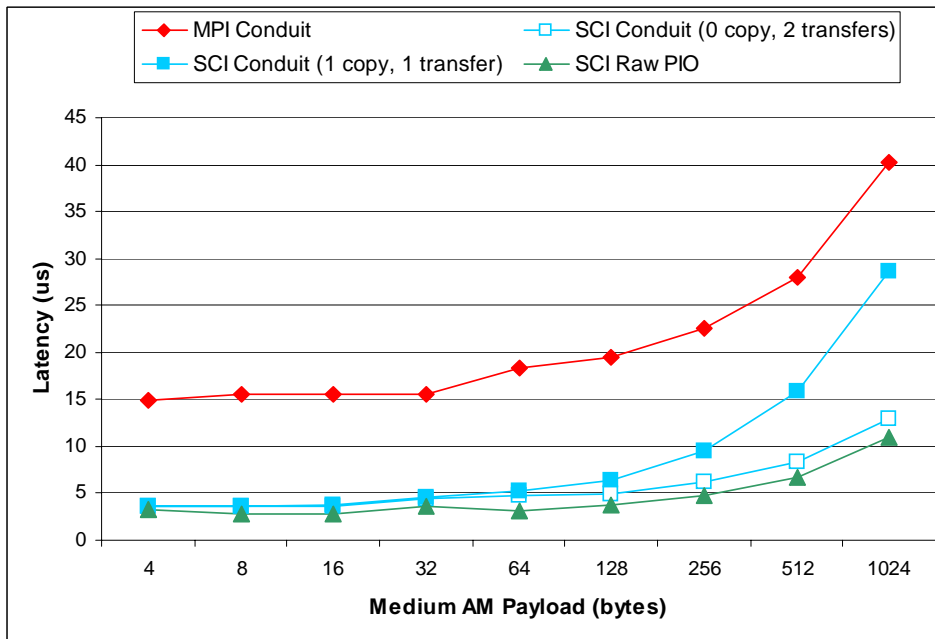
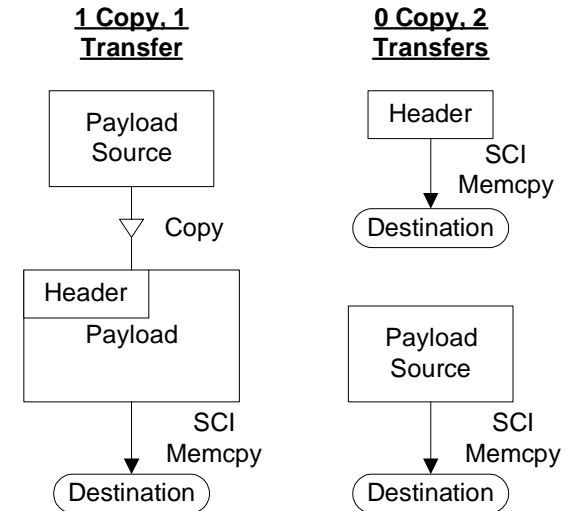
## Test Setup

- Each test was executed using 2 out of 4 nodes in the system
  - 1 Receiver, 1 Sender
- SCI Raw (SISCI's dma\_bench for DMA, own code for PIO)
  - One-way latency
    - Ping-Pong latency / 2 (PIO)
    - DMA completion time (DMA)
    - 1000 repetitions
- GASNet MPI Conduit (ScaMPI)
  - Include message polling overhead
  - One-way latency
    - Ping-Pong latency / 2
    - 1000 repetitions
- GASNet SCI Conduit
  - One-way latency
    - Ping-Pong latency / 2
    - 5000 repetitions (low variance)



# GASNet/SCI - Preliminary Results (Short/Medium AM)

- Short AM
  - Latency
    - SCI Conduit = 3.28  $\mu$ s
    - MPI Conduit (ScaMPI) = 14.63  $\mu$ s
- Medium AM
  - SCI-Conduit\*
    - Two ways to transfer Header + payload
      - 1 Copy, 1 Transfer
      - 0 Copy, 2 Transfers
      - 0 Copy, 1 Transfer not possible due to non-contiguous header/payload regions

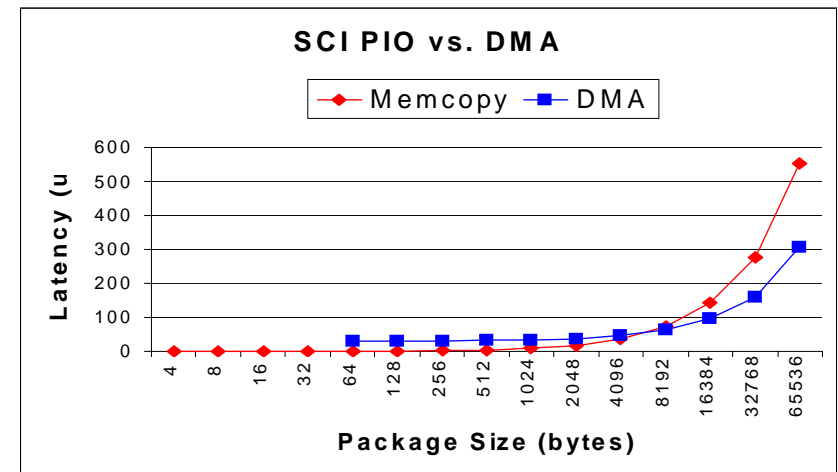
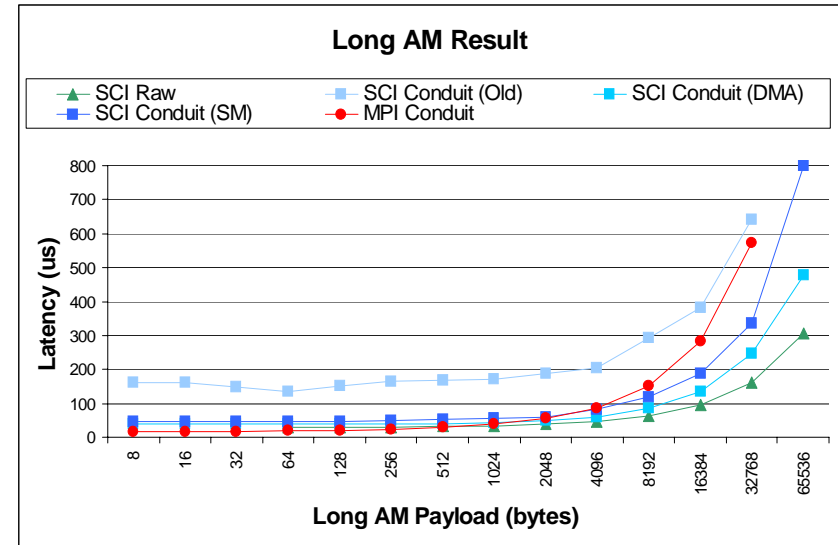


- Analysis
  - More efficient to import all segments at initialization time
    - SCI Conduit performs better than MPI Conduit
    - MPI Conduit utilizes dynamic allocation
      - Incurs large overhead to establish connection
  - SCI Conduit - setup time unavoidable
    - Need to package message to appropriate format
    - Need to send 80B Header
  - Copy Time > 2nd Transfer Time
    - 0 Copy, 2 Transfer case performs better than 1 Copy, 1 Transfer case

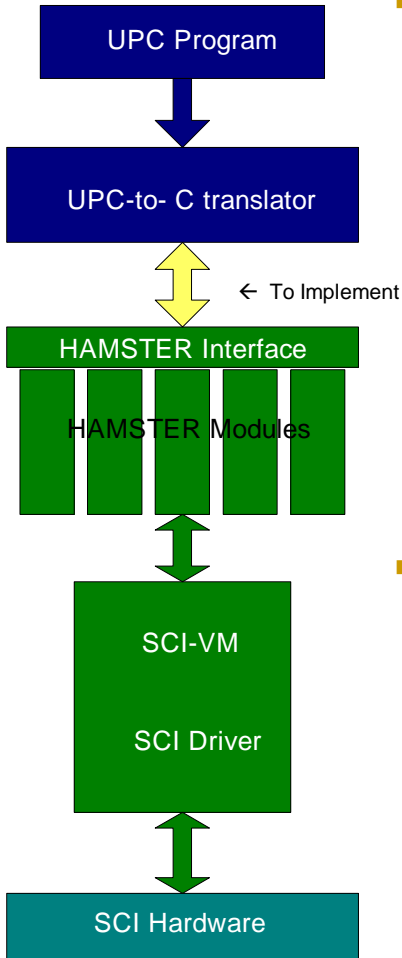
# GASNet/SCI - Preliminary Results (Long AM)


## ■ Analysis

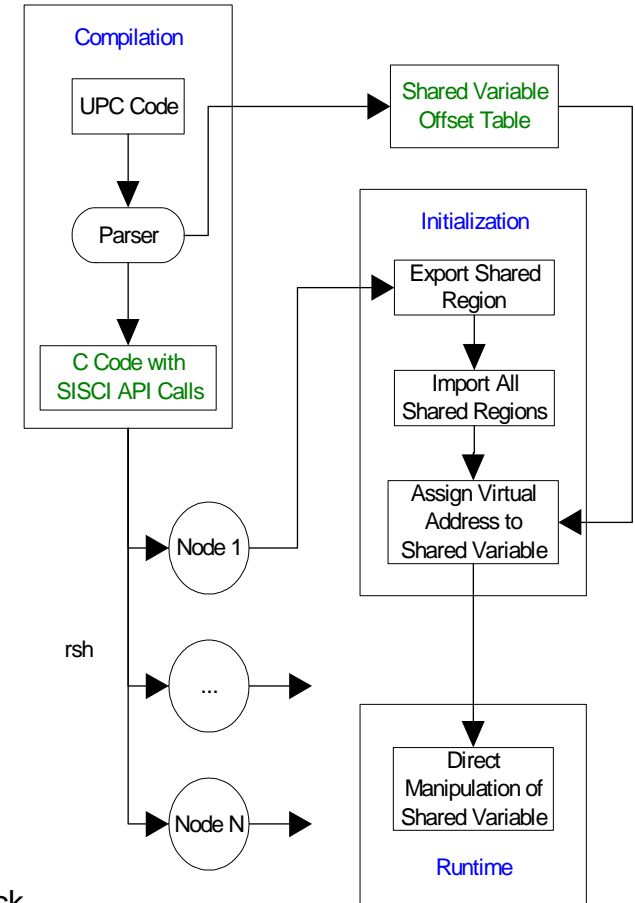
- Purely Import As Needed approach yields unsatisfactory result (SCI Conduit – Old)
  - 100+  $\mu$ s Local DMA queue setup time
- Performance can be improved by importing all payload segments at initialization time
  - Unfavorable with Berkeley Group
    - Desire scalability (32-bits Virtual Address Limitation) over performance
  - Expect to converge with SCI Raw DMA
- Hybrid approach reduces latency (SCI Conduit – DMA)
  - Single DMA queue setup at initialization time
  - Remote segment connected at transfer time
    - Incur fixed overhead
  - Transfer achieved by
    - Data copying from source to DMA queue
    - Transfer across network using DMA queue and remote segment
  - Result includes 5  $\mu$ s AM Header Transfer
    - SCI RAW DMA result only includes DMA transfer
  - High performance gain with slightly reduced scalability
  - Diverge from SCI Raw DMA at large payload size due to data copying overhead
    - Unavoidable due to SISI API limitation
    - SCIRegisterMemorySegment() not operational
- SCI PIO vs. DMA result suggests possible optimization
  - PIO for message < 8KB, DMA for rest
  - Approach yield unsatisfactory result (SCI Conduit – SM)
    - Mapping of remote segment incurs greater overhead than DMA approach



# Developing Concepts



- UPC/DSM/SCI
  - SCI-VM (DSM system for SCI) 
  - HAMSTER interface allows multiple modules to support MPI and shared memory models
    - Created using Dolphin SISCO API, ANSI C
  - Executes different threads in C on different machines in SCI-based cluster
  - Minimal development time by utilizing available Berkeley UPC-to-C translator and SCI-VM C module
    - Build different software systems
    - Integration of independent components
  
- SCI-UPC
  - Implements UPC directly on SCI
  - Parser converts UPC code to C code with embedded SISCO API calls
  - Converts shared pointer to local pointer
    - Performance improvement base on benchmarking result
  - Environment setup at initialization time
    - Reduce runtime overhead
  - Direct call to SISCO API
    - Minimize overhead by compressing stack



# Network Performance Tests

- Detailed understanding of high-performance cluster interconnects
  - Identifies suitable networks for UPC over clusters
  - Aids in smooth integration of interconnects with upper-layer UPC components
  - Enables optimization of network communication; unicast and collective
- Various levels of network performance analysis

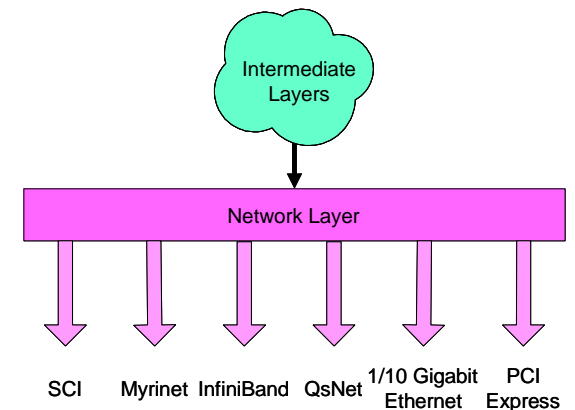
- Low-level tests

- InfiniBand based on Virtual Interface Provider Library (VIPL)
- SCI based on Dolphin SISCi and SCALI SCI
- Myrinet based on Myricom GM
- QsNet based on Quadrics Elan Communication Library
- Host architecture issues (e.g. CPU, I/O, etc.)



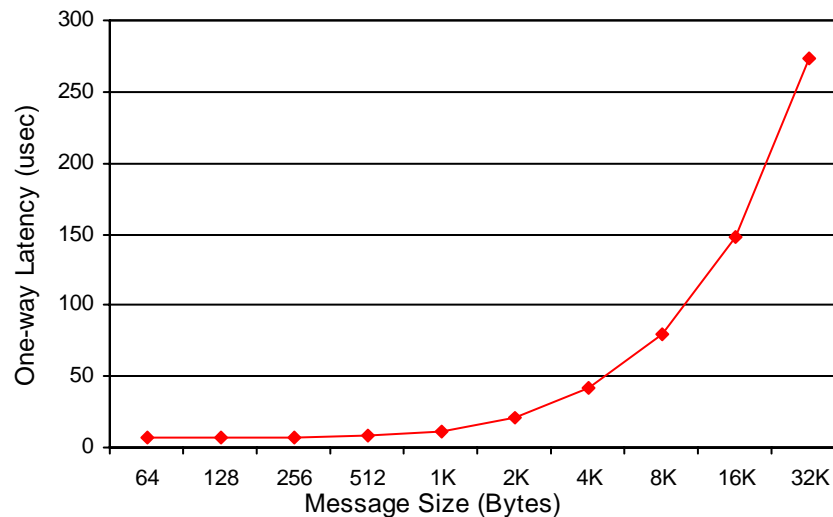
- Mid-level tests

- Sockets
  - Dolphin SCI Sockets on SCI
  - BSD Sockets on Gigabit and 10Gigabit Ethernet
  - GM Sockets on Myrinet
  - SOVIA on Infinband
- MPI
  - InfiniBand and Myrinet based on MPI/PRO
  - SCI based on ScaMPI and SCI-MPICH



# Network Performance Tests - VIPL

- Virtual Interface Provider Library (VIPL) performance tests on InfiniBand
  - Testbed
    - Dual 1.0 GHz Pentium-III's, 256MB PC133, ServerWorks CNB20LE Host Bridge (rev. 6)
    - 4 nodes, 1x Fabric Networks' InfiniBand switch, 1x Intel IBA HCA
  - One-way latency test
    - 6.28 usec minimum latency at 64B message size
- Currently preparing VIPL API latency and throughput tests for 4x (10 Gb/s) Fabric Networks InfiniBand 8-node testbed



# Network Performance Tests - Sockets

## ■ Testbed

- SCI Sockets results provided by Dolphin
- BSD Sockets
  - Dual 1.0 GHz Pentium-IIIs, 256MB PC133, ServerWorks CNB20LE Host Bridge (rev. 6)
  - 2 nodes, 1 Gb/s Intel Pro/1000 NIC

## ■ Sockets latency and throughput tests

- Dolphin SCI Sockets on SCI
- BSD Sockets on Gigabit Ethernet

## ■ One-way latency test

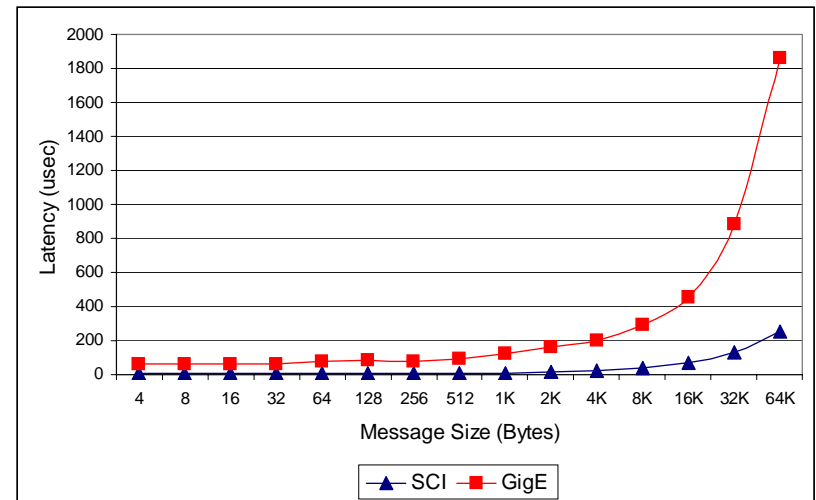
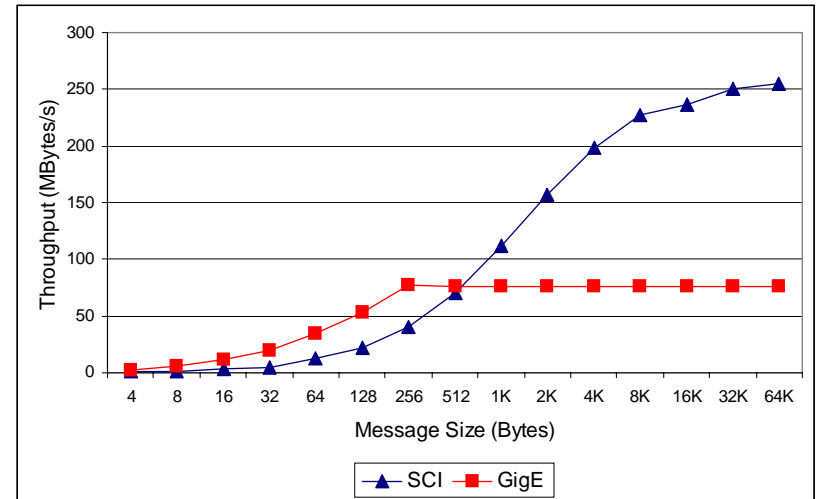
- SCI - 5 is min. latency
- BSD - 54 is min. latency

## ■ Throughput test

- SCI - 255 MB/s (2.04 Gb/s) max. throughput
- BSD - 76.5 MB/s (612 Mb/s) max. throughput
  - Better throughput (> 900 Mb/s) with optimizations

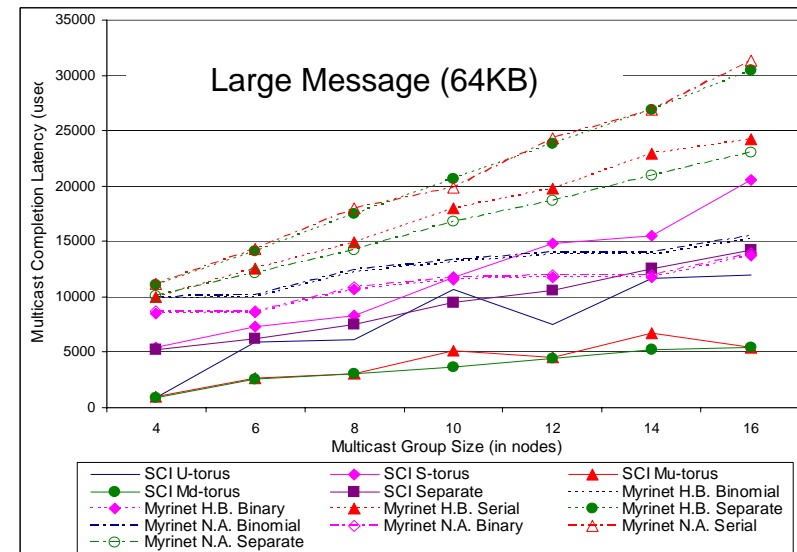
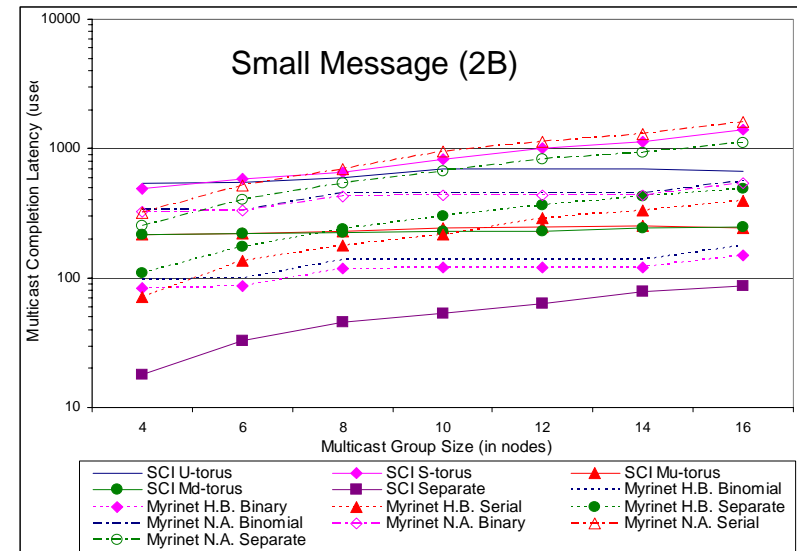
## ■ SCI Sockets allow high-performance execution of standard BSD socket programs

- Same program can be run without recompilation on either standard sockets or SCI Sockets over SCI
- SCI Sockets are still in design and development phase



# Network Performance Tests – Collective Comm.

- User-level multicast comparison of SCI and Myrinet
  - SCI's flexible topology support
    - Separate addressing, S-torus,  $M_d$ -torus,  $M_u$ -torus, U-torus
  - Myrinet NIC's co-processor for work offloading from host CPU
    - Host-based (H.B.), NIC-assisted (N.A.)
      - Separate addressing, serial forwarding, binary tree, binomial tree
- Small and large message sizes
  - Multicast completion latency
  - Host CPU utilization
- Testbed
  - Dual 1.0 GHz Pentium-IIIs, 256MB PC133, ServerWorks CNB20LE Host Bridge (rev. 6)
  - 5.3 Gb/s Dolphin SCI D339 NICs, 4x4 torus
  - 1.28 Gb/s Myricom M2L-PCI64A-2 Myrinet NICs, 16 nodes
- Analysis
  - Multicast completion latency
    - Simple algorithms perform better for small messages
      - SCI separate addressing
    - More sophisticated algorithms perform better for large messages
      - SCI  $M_d$ -torus and SCI  $M_u$ -torus
    - SCI performs better than Myrinet for large messages
      - Higher link data rate compared to Myrinet
      - SCI best for small messages with separate addressing
  - Host CPU utilization
    - Myrinet NIC-Assisted scheme provides low host CPU utilization for all message and group sizes
      - Both complex and simple algorithms



# Conclusions and Future Plans

- Accomplishments
  - Baselineing of UPC on shared-memory multiprocessors
  - Evaluation of promising tools for UPC on clusters
  - Leverage and extend communication and UPC layers
  - Conceptual design of new tools
  - Preliminary network and system performance analyses
  
- Key insights
  - Existing tools are limited but developing and emerging
  - SCI is a promising interconnect for UPC
    - Inherent shared-memory features and low latency for memory transfers
    - Breakpoint in latency @ 8KB; copy vs. transfer latencies; scalability from 1D to 3D tori
  - Method of accessing shared memory in UPC is important for optimal performance
  
- Future Plans
  - Further investigation of GASNet and related tools, and extended design of SCI conduit
  - Evaluation of design options and tradeoffs
  - Comprehensive performance analysis of new and emerging SANs and SAN-based clusters
  - Evaluation of UPC methods and tools on various architectures and systems
  - Cost/Performance analysis for all options