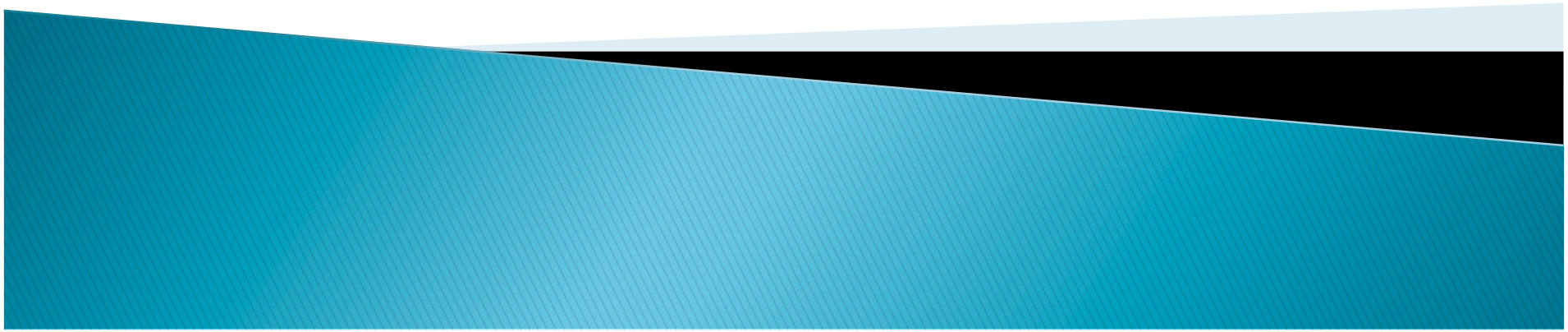
Network

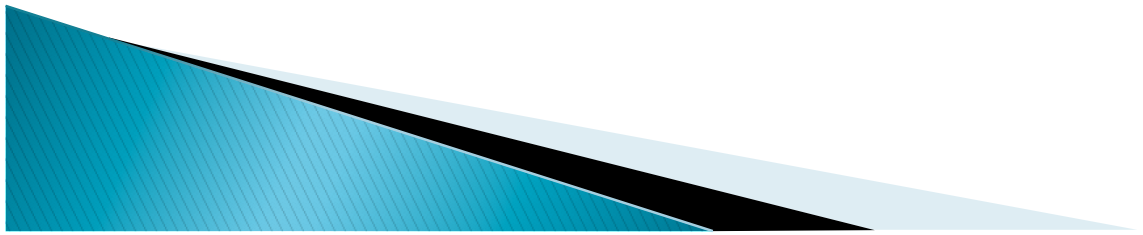
Haoyuan Li
CS 6410 Fall 2009
10/15/2009



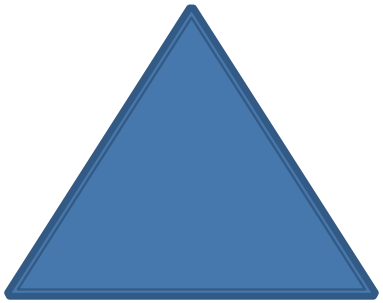
Data Center Arms Race

System Area Network

SAN



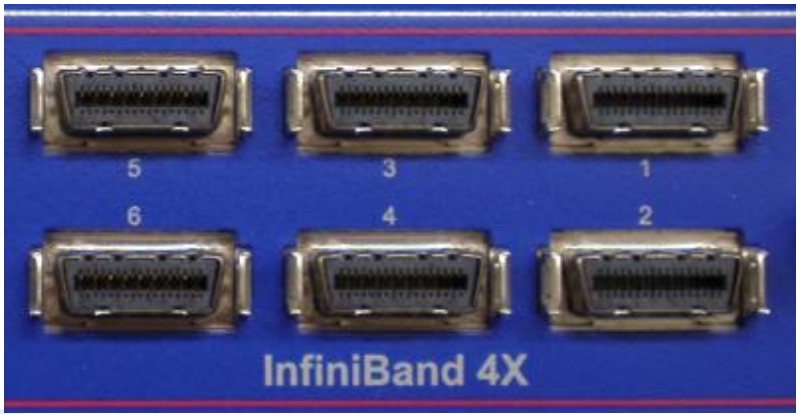
SAN



Virtual Interface Architure

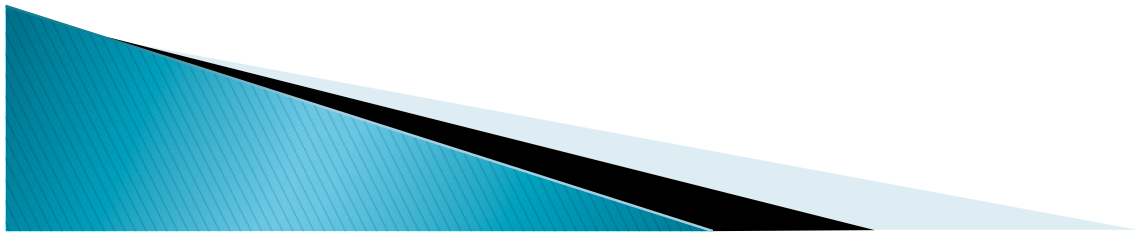
VIA

Microsoft intel COMPAQ



Agenda

- U-Net: A User-Level Network Interface for Parallel and Distributed Computing
  - Thorsten von Eicken, Anindya Basu, Vineet Buch, and Werner Vogels
- Active Messages: a Mechanism for Integrated Communication and Computation
  - Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser



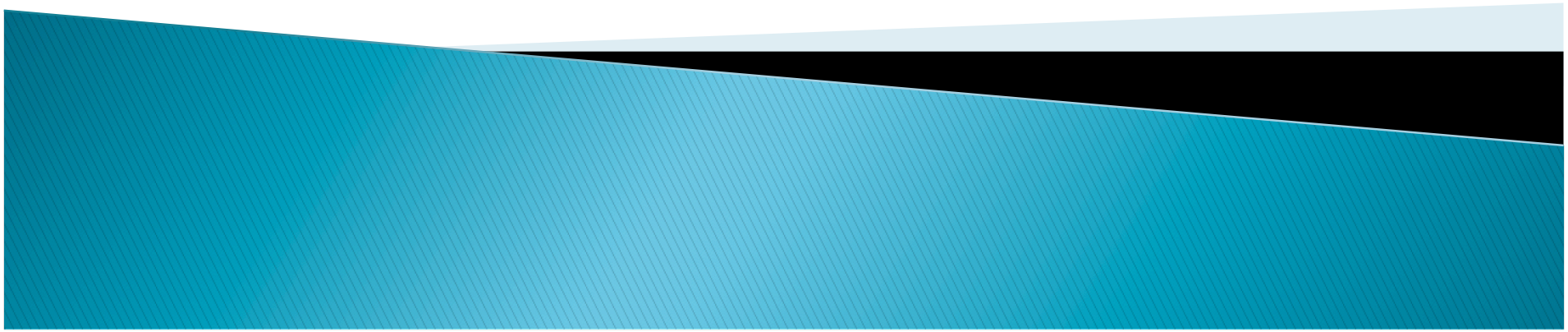
Authors' fact

- Thorsten von Eicken
- Werner Vogels
- David E. Culler
- Seth Copen Goldstein
- Klaus Erik Schauser



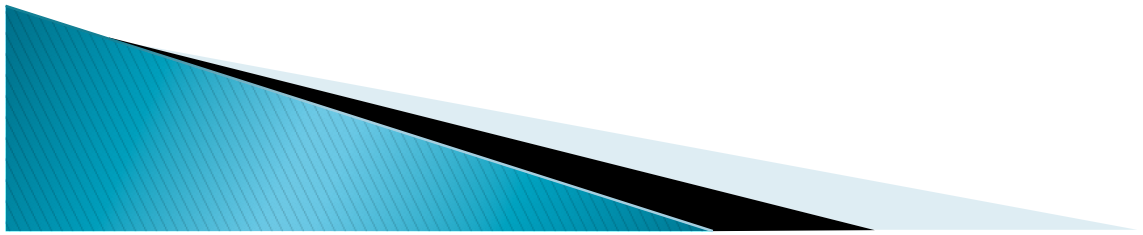


U-Net: A User-Level Network Interface for Parallel and Distributed Computing



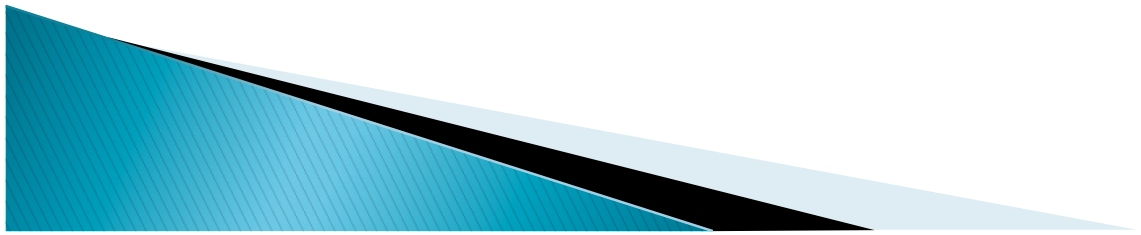
Outline

- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite
- Conclusion



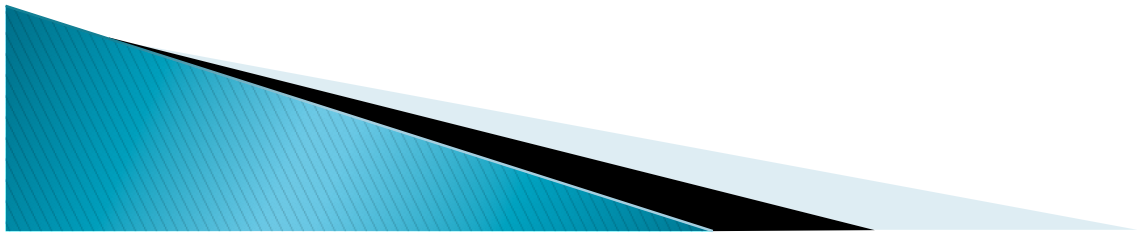
Outline

- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite
- Conclusion



Motivation

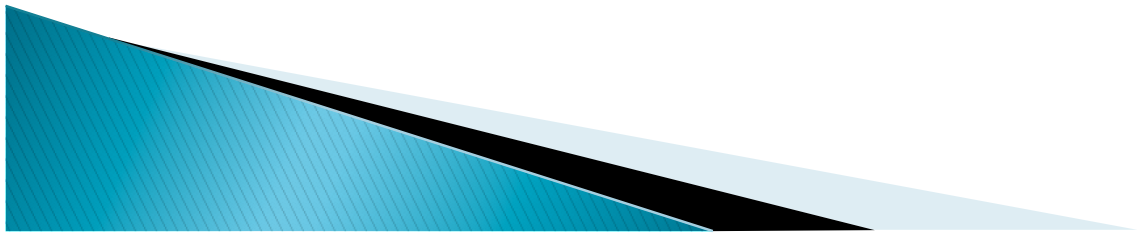
- Processing Overhead
  - Fabrics bandwidth vs. Software overhead
- Flexibility
  - Design new protocol
- Small Message
  - Remote object executions
  - Cache maintaining messages
  - RPC style client/server architecture
- Economic driven
  - Expensive multiprocessors super computers with custom network design
  - vs.
  - Cluster of standard workstations connected by off-the-shelf communication hardware



U-Net Design goal

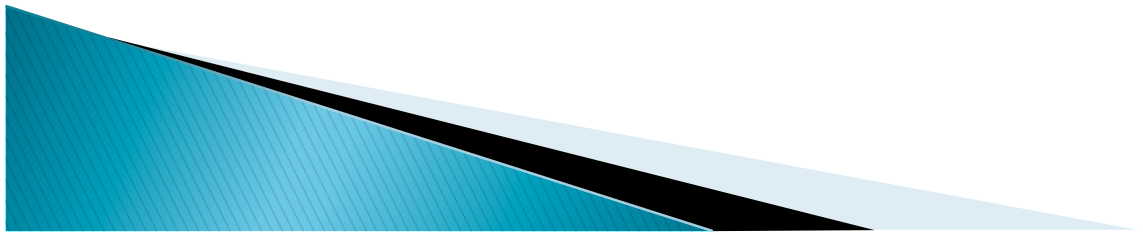
- Provide low-latency communication in local area setting
- Exploit the full network bandwidth even with small message
- Facilitate the use of novel communication protocols

- All built on CHEAP hardware!

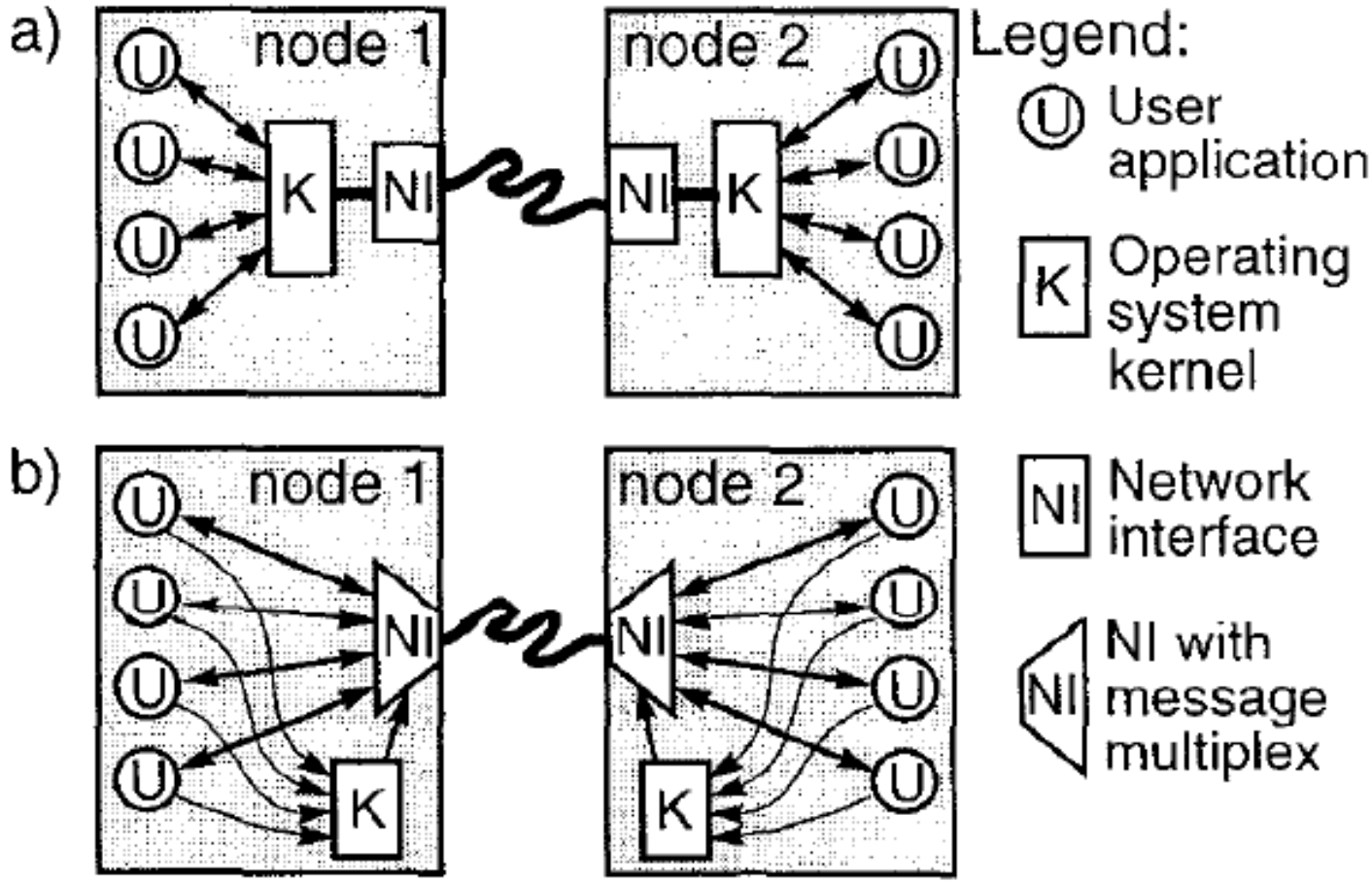


Outline

- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite
- Conclusion

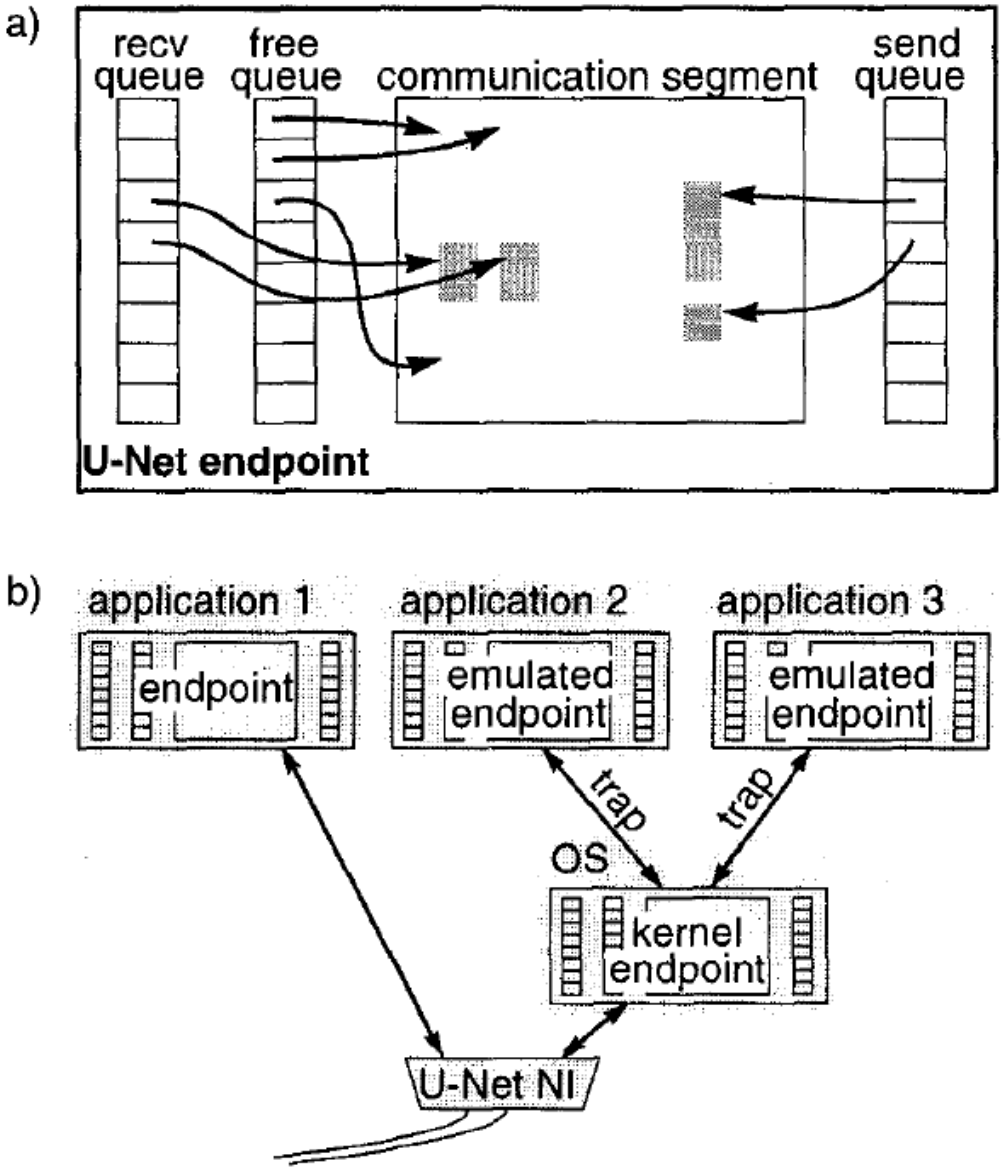


Traditional networking architecture VS. U-Net architecture



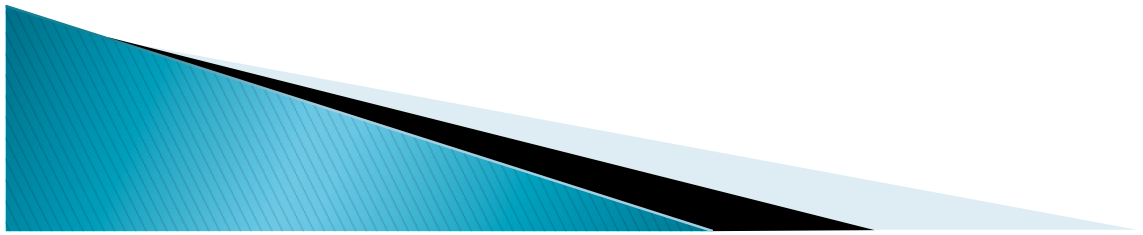
U-Net architecture

- Communication Segments
- Send queue
- Receive Queue
- Free Queue



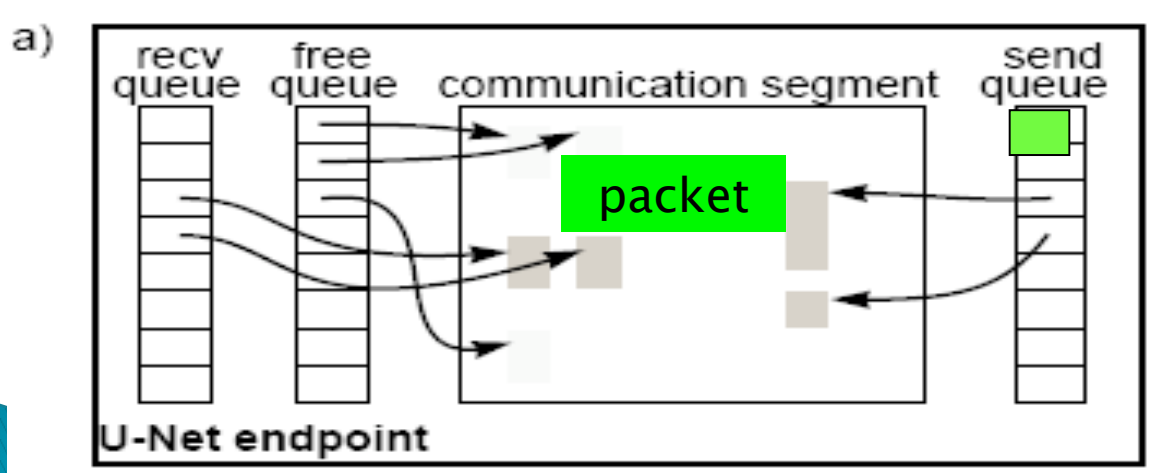
U-Net design details

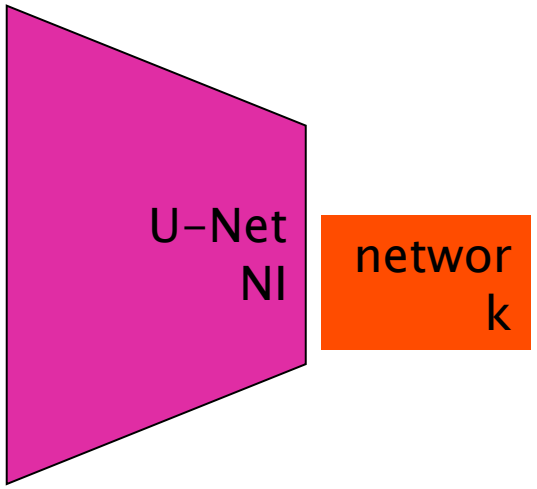
- Send and Receive packet
- Multiplexing and demultiplexing messages
- Zero-copy vs. true Zero-copy
- Base-Level U-Net
- Kernel emulation of U-Net
- Direct-Access U-Net



Sending Packet

- Prepare packet and place it in the Communication segment
- Place descriptor on the Send queue
- U-Net takes descriptor from queue
- transfers packet from memory to network

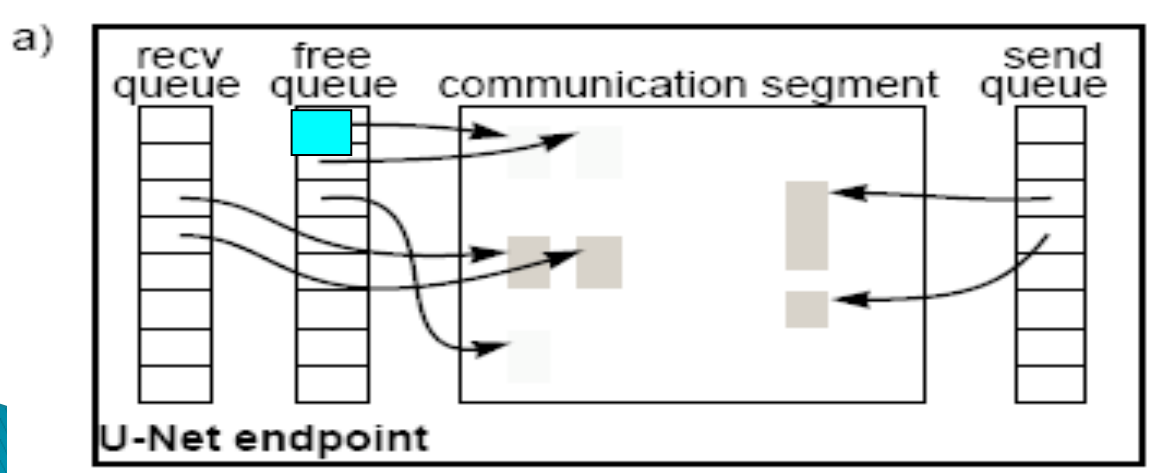


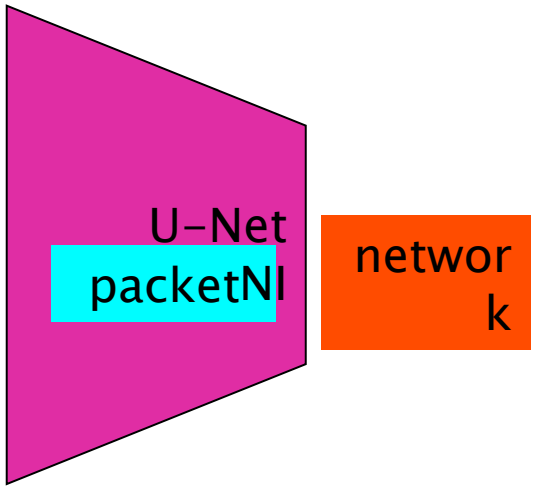


From Itamar Sagi

Receive Packet

- U-Net receives message and decide which Endpoint to place it
- Takes free space from Free Queue
- Place message in Communication Segment
- Place descriptor in receive queue
- Process takes descriptor from receive queue (polling or signal) and reads message

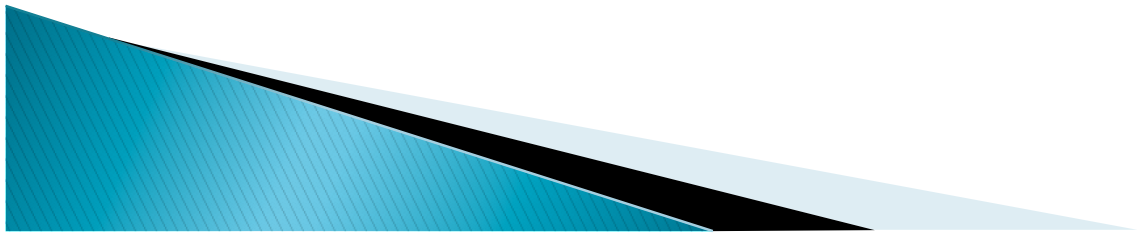




From Itamar Sagi

Multiplexing and demultiplexing messages

- Channel setup and memory allocation
- Communication Channel ID
- Isolation Protection

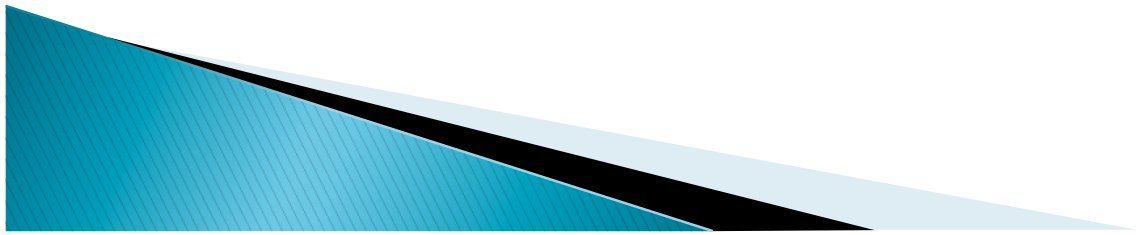


Zero-copy vs. true Zero-copy

- True Zero-copy: No intermediate buffering
  - Direct-Access U-Net
    - Communication segment spans the entire process address space
    - Specify offset where data has to be deposited
- Zero-copy: One intermediate copy into a networking buffer
  - Base-Level U-Net
    - Communication segment are allocated and pinned to physical memory
    - Optimization for small messages
  - Kernel emulation of U-Net
    - Scarce resources for communication segment and message queues

Outline

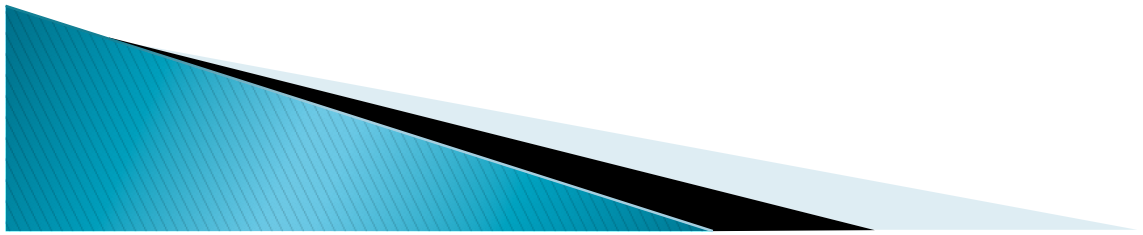
- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite



U-Net Implementation

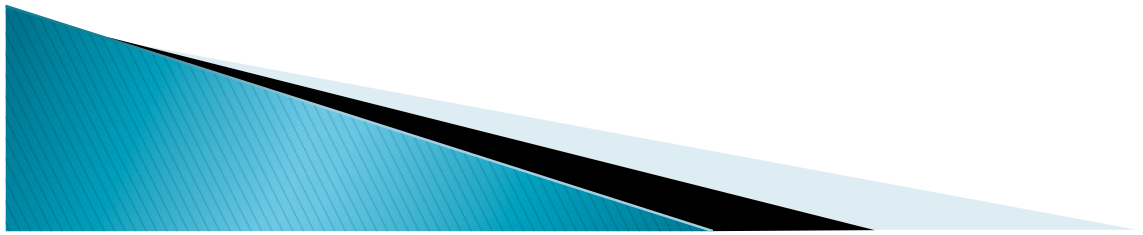
- SPARCstations
- SunOS 4.1.3
- Fore SBA-100 and Fore SBA-200 ATM interfaces by FORE Systems, now part of Ericsson

- AAL5



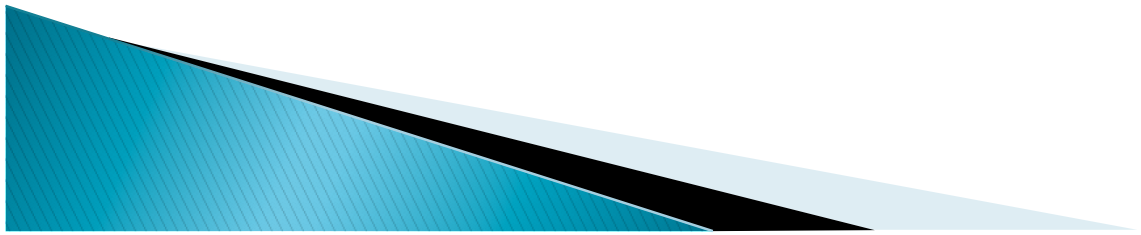
U-Net Implementation on Fore SBA-200

- Onboard processor
- DMA capable
- AAL5 CRC generator
- Firmware changed to implement U-Net NI on the onboard processor



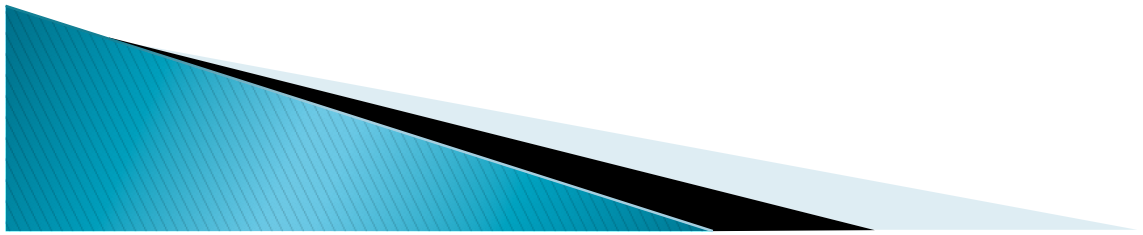
Outline

- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite
- Conclusion



Active Messages Evaluation

- Active Messages
  - A mechanism that allows efficient overlapping of communication with computation in multiprocessors
- Implementation of GAM specification over U-Net



Performance Raw-U-Net and UAM

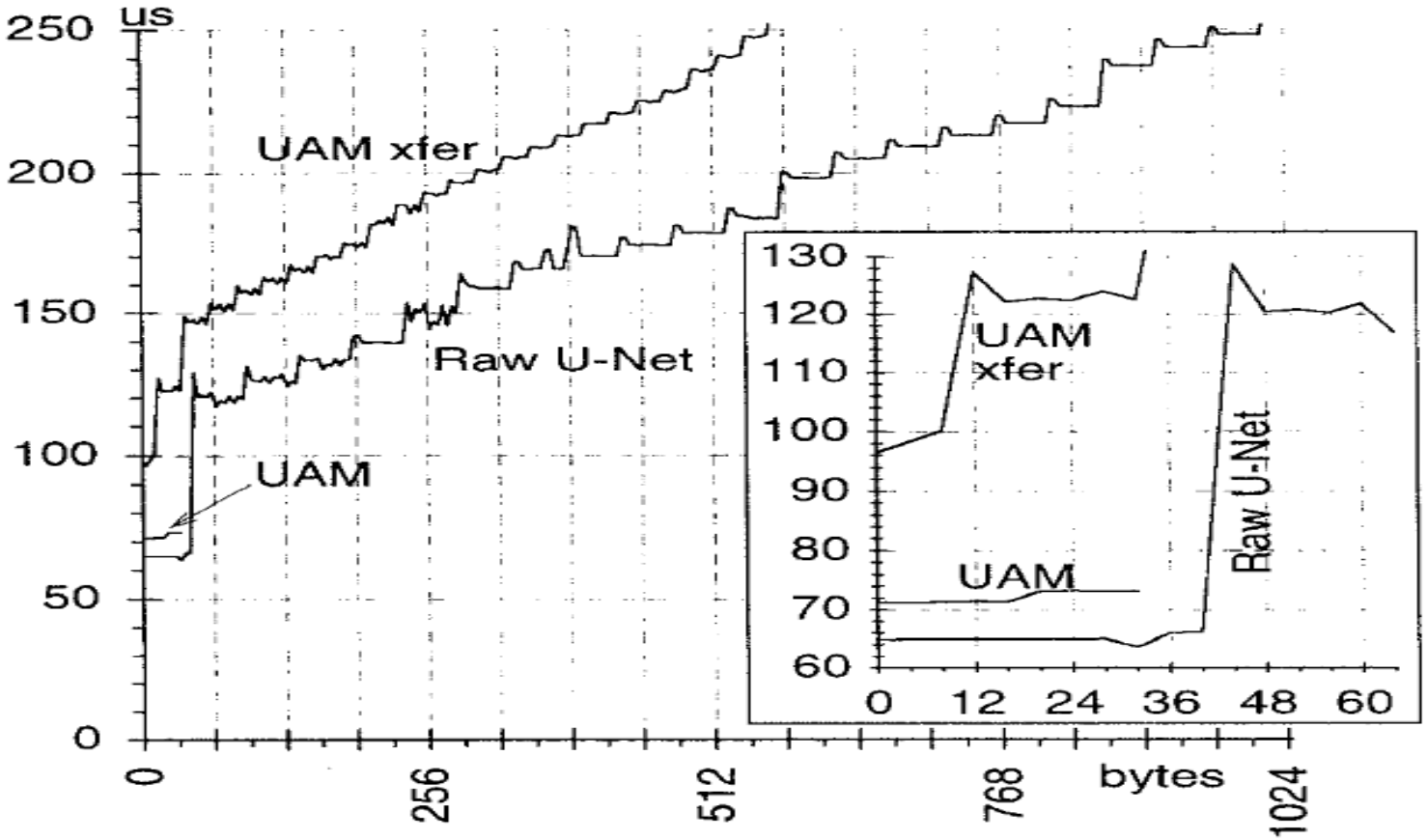


Figure 3: U-Net round-trip times as a function of message size. The *Raw U-Net* graph shows the round-trip times for a simple ping-pong benchmark using the U-Net interface directly. The inset graph highlights the performance on small messages. The *UAM* line measures the performance of U-Net Active Messages using reliable single-cell requests and replies whereas *UAM xfer* uses reliable block transfers of arbitrary size.

Performance Raw-U-Net and UAM

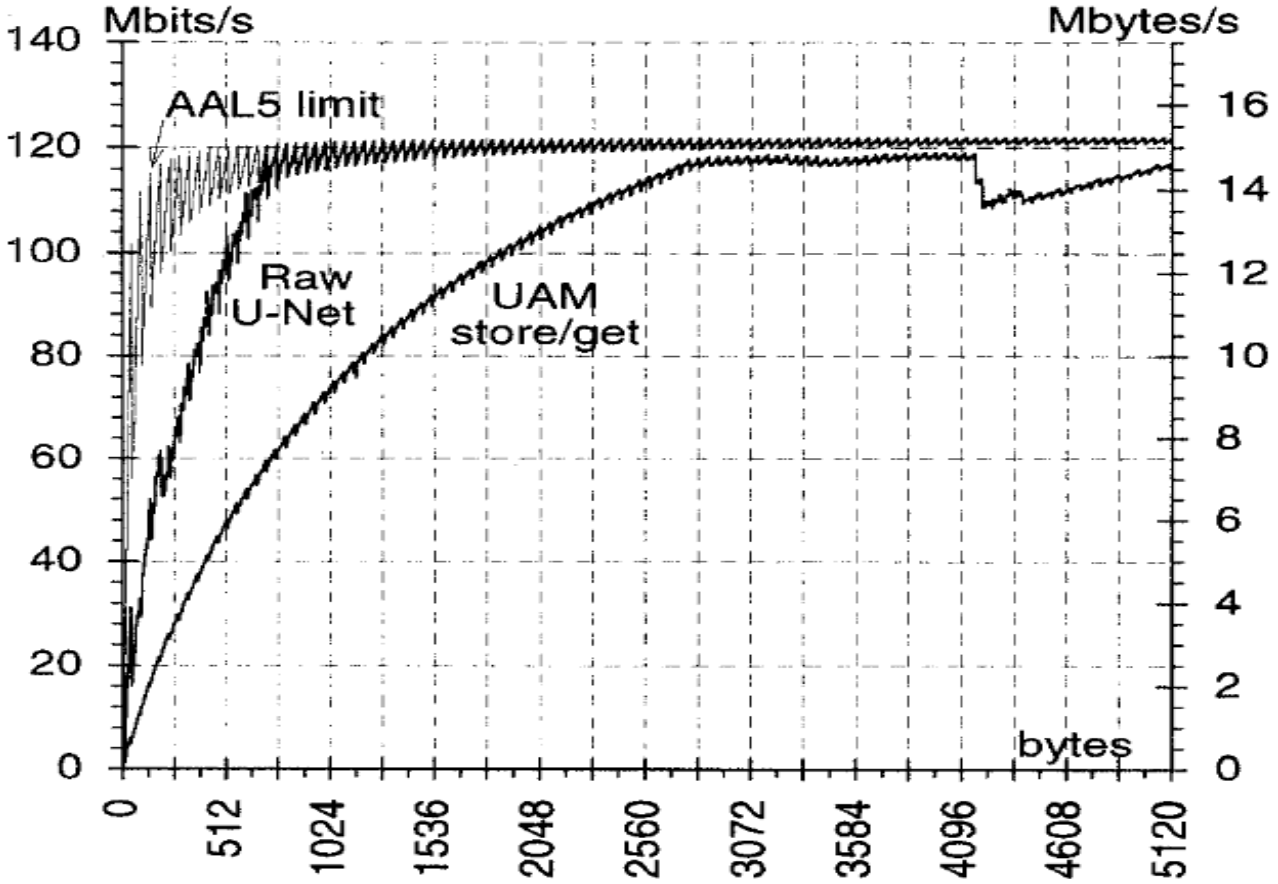


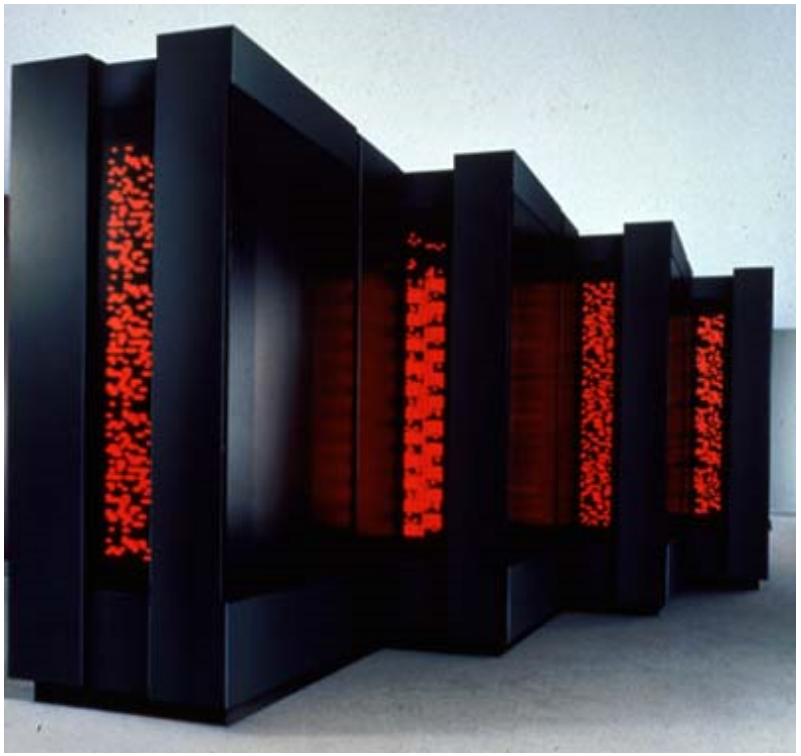
Figure 4: U-Net bandwidth as a function of message size. The *AAL-5 limit* curve represents the theoretical peak bandwidth of the fiber (the sawtooths are caused by the quantization into 48-byte cells). The *Raw U-Net* measurement shows the bandwidth achievable using the U-Net interface directly, while *UAM store/get* demonstrate the performance of reliable U-Net Active Messages block transfers.

Split-C Evaluation

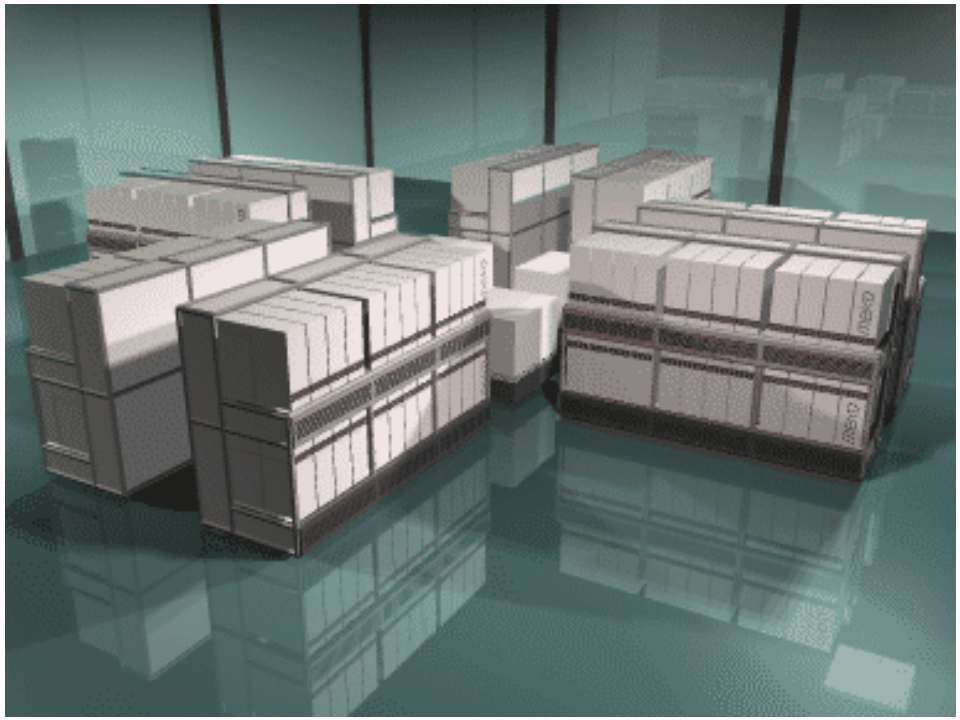
- Split C based on UAM
- Vs.
- CM-5
- Meiko CS-2

Machine	CPU speed	message overhead	round-trip latency	network bandwidth
CM-5	33 Mhz Sparc-2	3µs	12µs	10Mb/s
Meiko CS-2	40Mhz Supersparc	11µs	25µs	39Mb/s
U-Net ATM	50/60 Mhz Supersparc	6µs	71µs	14Mb/s

CM-5

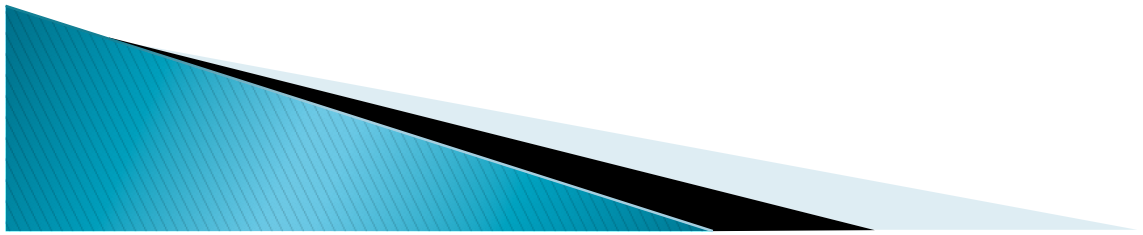


Meiko CS-2

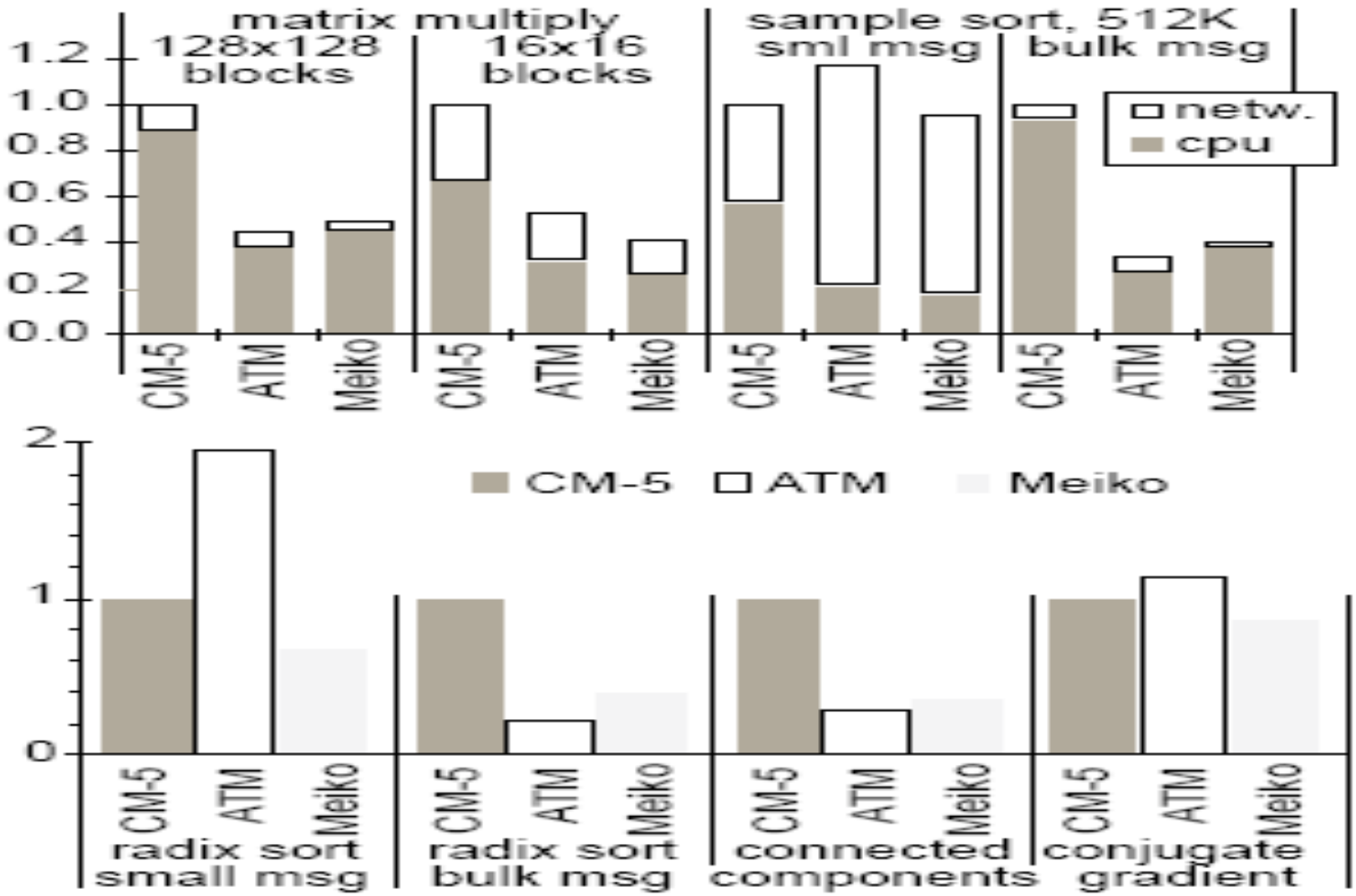


Split-C application benchmarks

- Block matrix multiply
- Sample sort (2 versions)
- Radix sort (2 versions)
- Connected component algorithm
- Conjugate gradient solver



Performance



IP Suite Evaluation

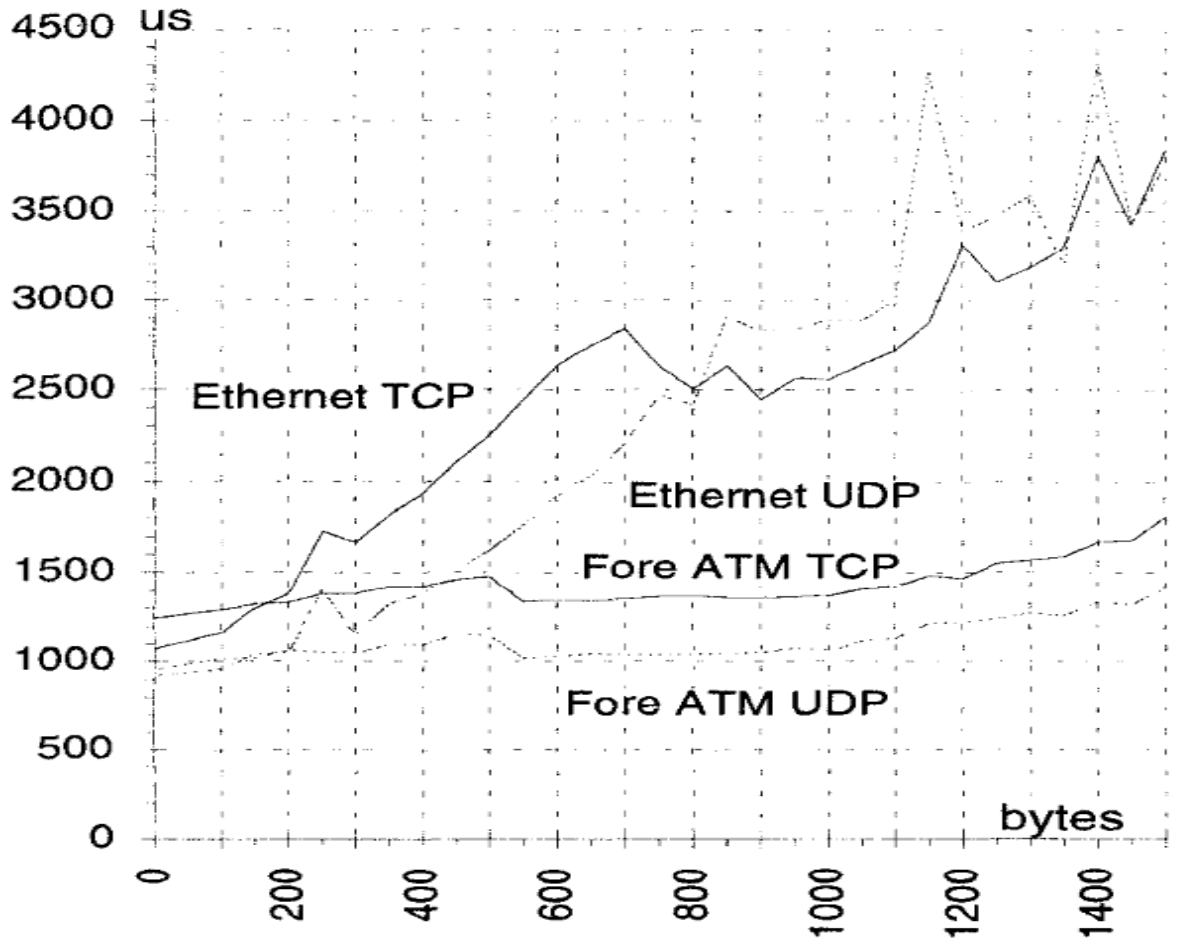


Figure 6: TCP and UDP round-trip latencies over ATM and Ethernet. as a function of message size.

IP Suite Evaluation

TCP max bandwidth

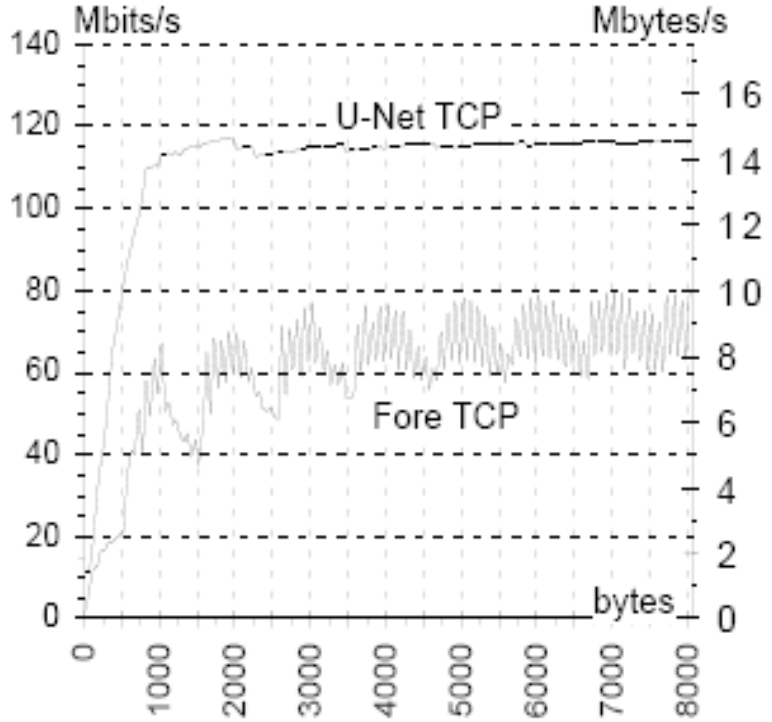


Figure 8: TCP bandwidth as a function of data generation by the application.

UDP max bandwidth

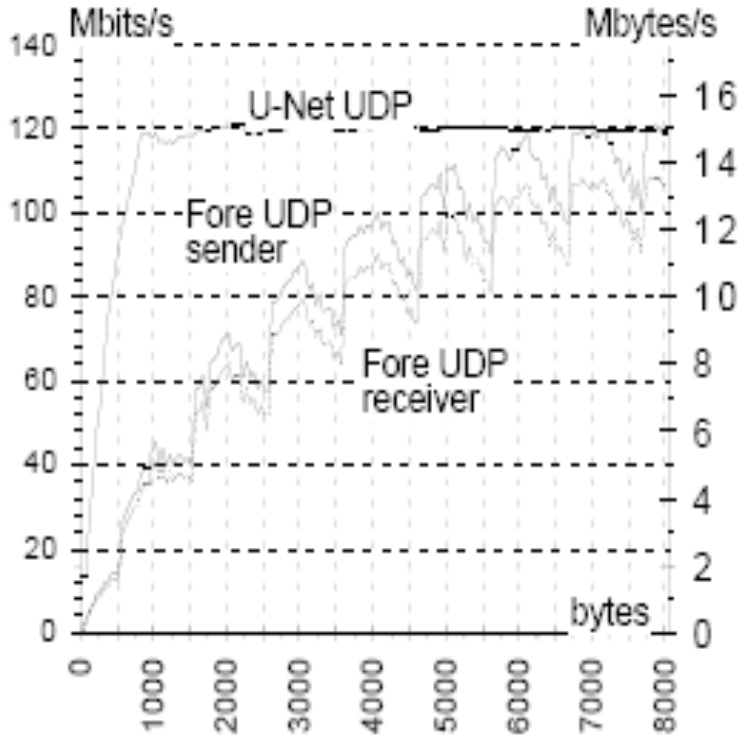
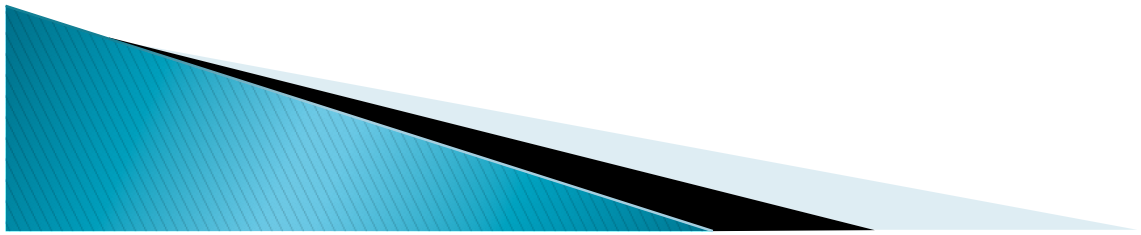


Figure 7: UDP bandwidth as a function of message size.

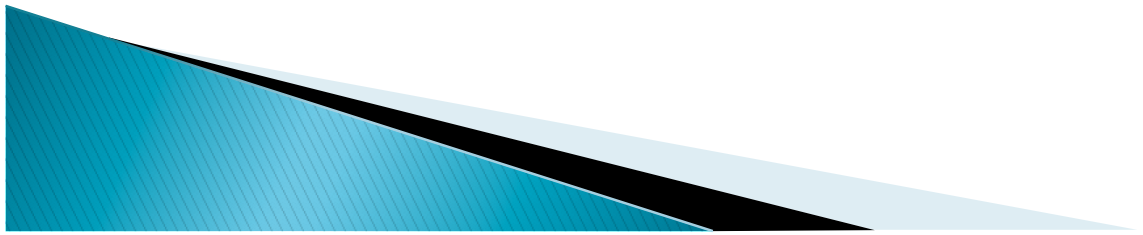
Outline

- Motivation
- Design
- Implementation
  - SBA-100
  - SBA-200
- Evaluation
  - Active Messages
  - Split-C
  - IP Suite
- Conclusion

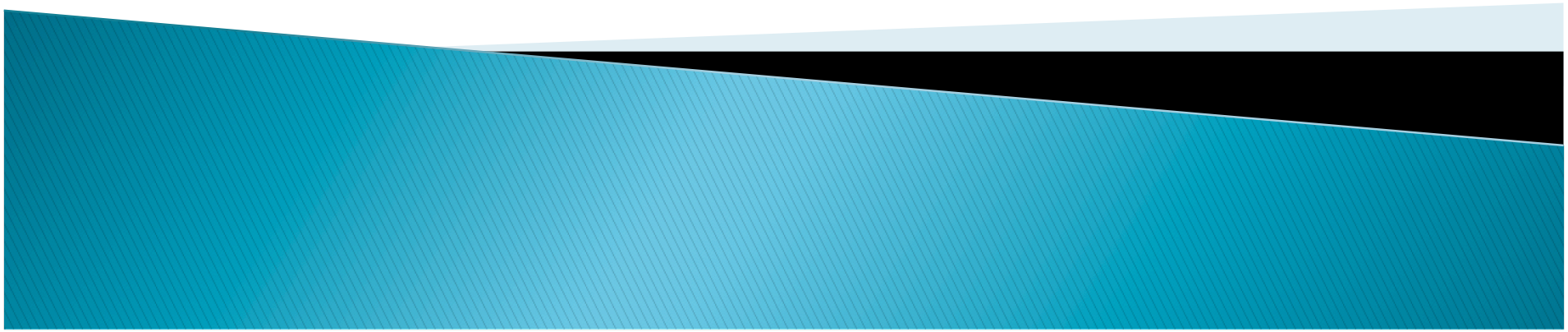


Conclusion

- U-Net main objectives achieved:
  - Provide efficient low latency communication
  - Offer a high degree of flexibility
- U-Net based round-trip latency for messages smaller than 40 bytes: Win!
- U-Net flexibility shows good performance on TCP and UDP protocol

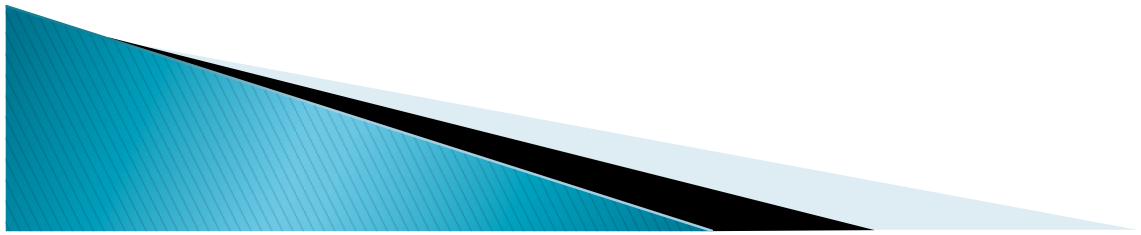


Active Messages: a Mechanism for Integrated Communication and Computation



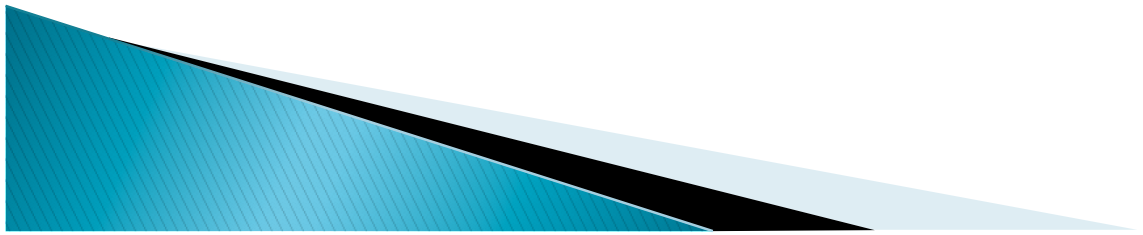
Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions



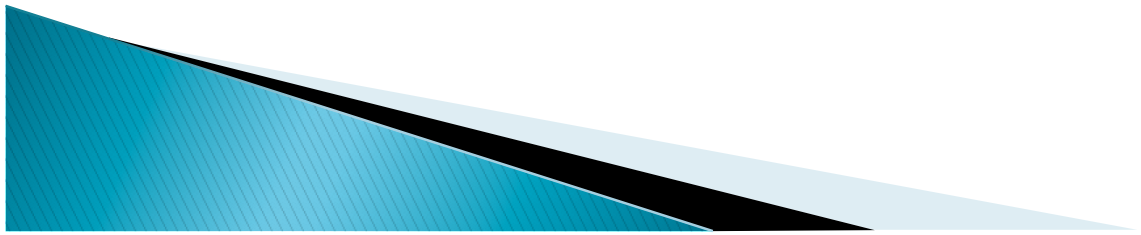
Large-scale multiprocessors design's key challenges

- Minimize communication overhead
- Allow communication to overlap computation
- Coordinate the two above without sacrificing processor cost/performance



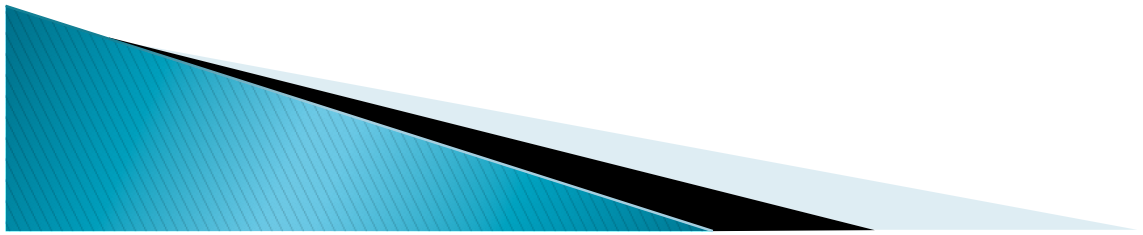
Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions



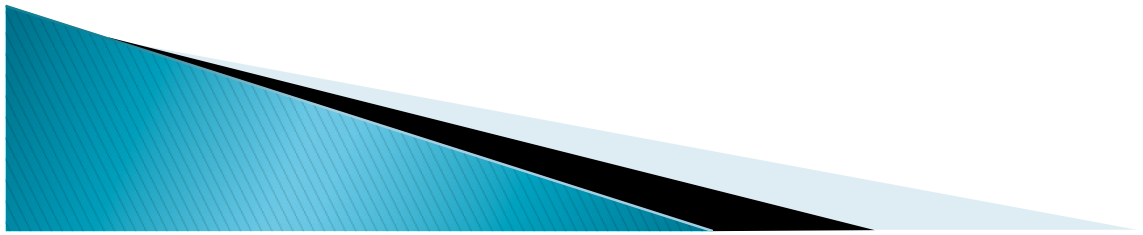
Active Messages

- Mechanism for sending messages
  - Message header contains instruction address
  - Handler retrieves message, cannot block, and no computing
  - No buffering available
- Making a simple interface to match hardware
- Allow computation and communication overlap



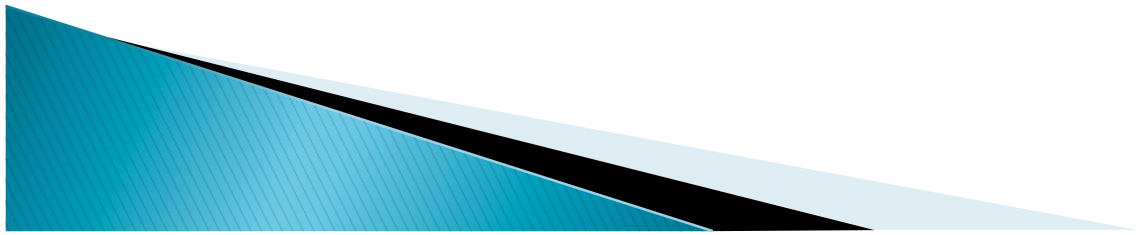
Active Message Protocol

- Sender asynchronous sends a message to a receiver without blocking computing
- Receiver pulls message, integrates into computation through handler
  - Handler executes without blocking
  - Handler provides data to ongoing computation, but not does any computation



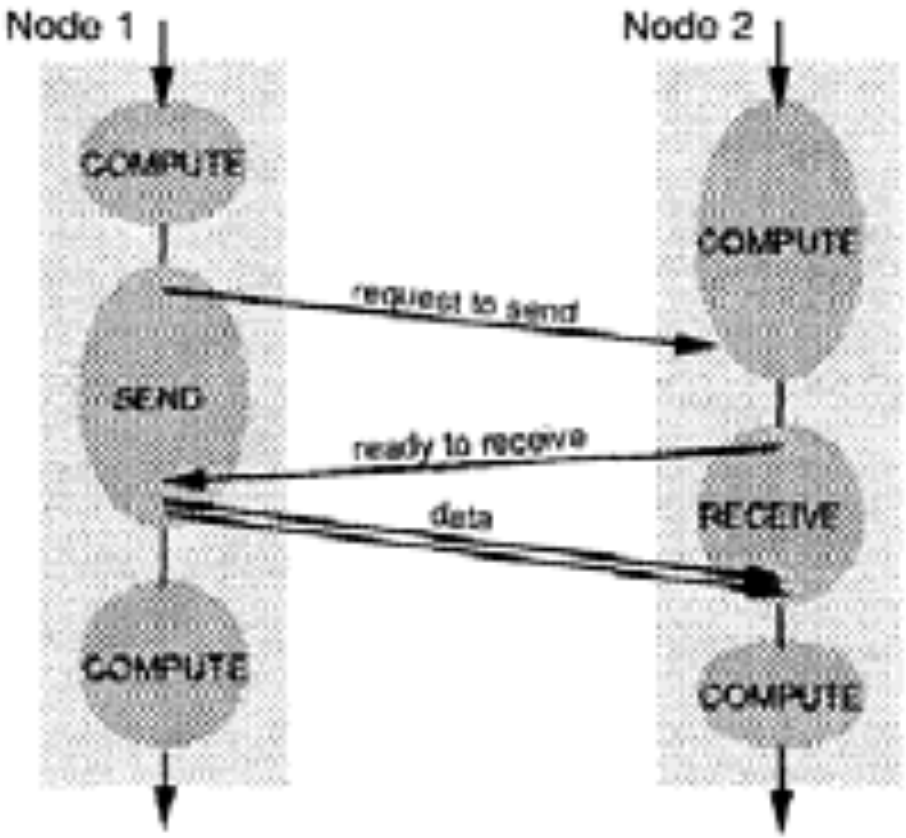
Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions



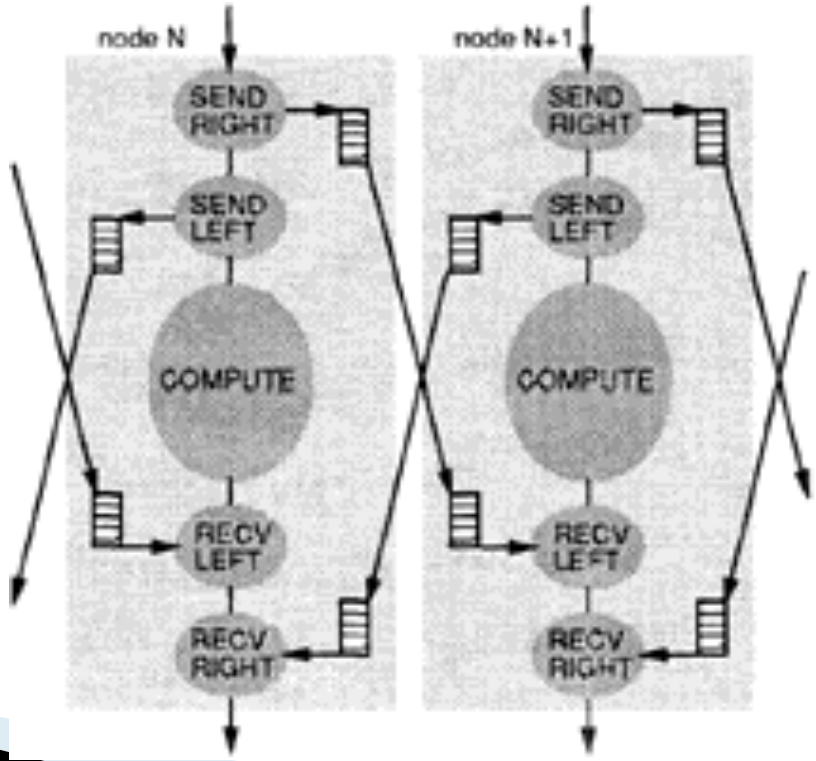
Blocking Send/Recv Model

- 3-Phase Protocol
- Simple
- Inefficient
- No buffering needed



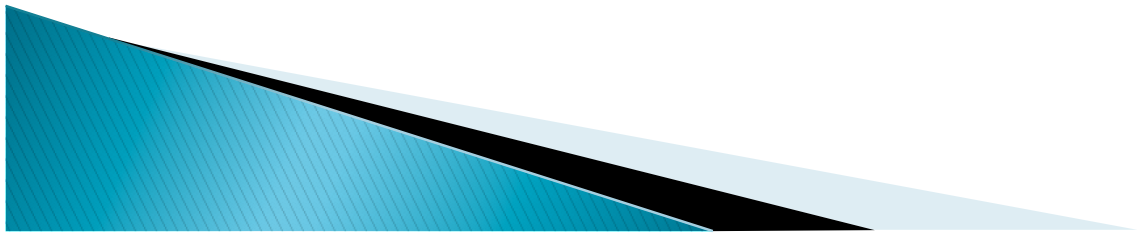
Asynchronous Send/Recv Model

- Communication can have overlap with computation
- Buffer space allocated throughout computation

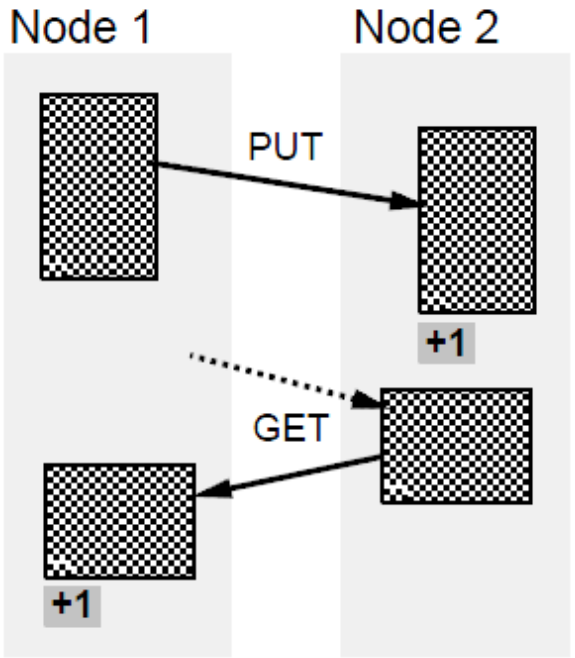


Split-C

- Extension of C for SPMD Programs
  - Global address space is partitioned into local and remote
  - Maps shared memory benefits to distributed memory
  - Split-phase access
- Active Messages serve as interface for Split-C



PUT / GET in Split-C



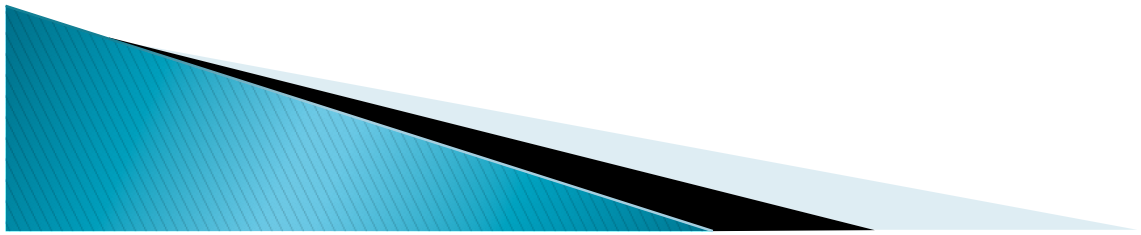
a)

PUT message	GET request
remote node	remote node
put handler	get handler
remote addr	local addr
data length	data length
flag addr	flag addr
data	local node
...	remote addr
...	

b)

Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions

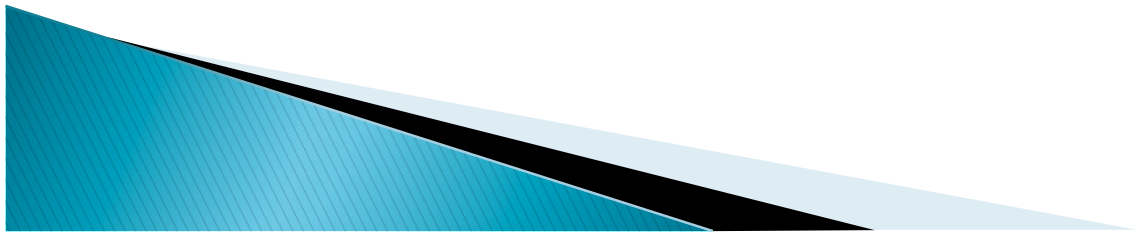


Message Driven Machines

- To support languages with dynamic parallelism
- Integrate communication into the processor
- Computation is driven by messages, which contain the name of a handler and some data
- Computation is within message handlers
- May buffer messages upon receipt
  - Buffers can grow to any size depending on amount of excess parallelism
- Less locality

Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions

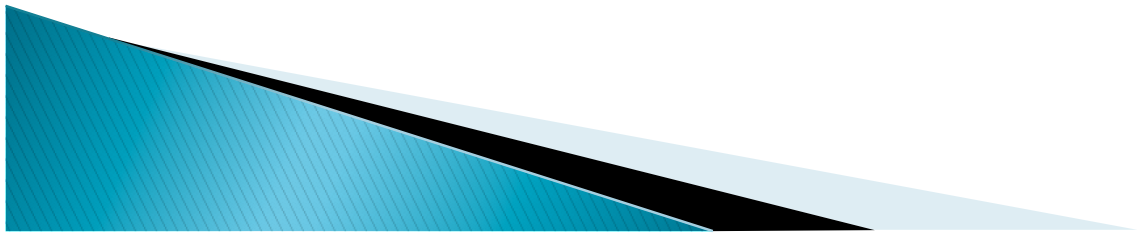


Potential Hardware support

- Network Interface Support
  - Large messages
  - Message registers
  - Reuse of message data
  - Single network port
  - Protection
  - Frequent message accelerators
- Processor Support
  - Fast polling
  - User-level interrupts
  - PC injection
  - Dual processors

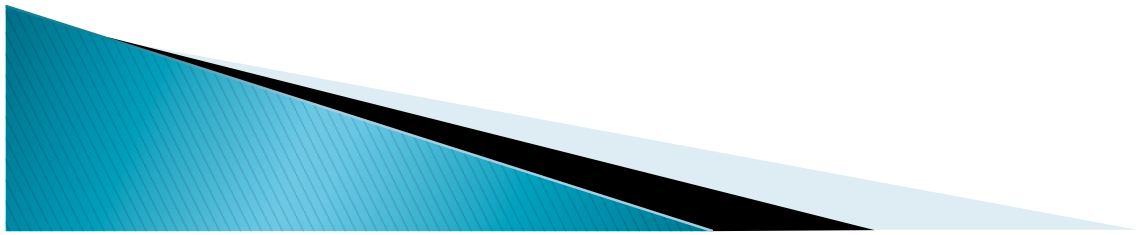
Outline

- Large-scale multiprocessors design's key challenges
- Active messages
- Message passing architectures
- Message driven architectures
- Potential hardware support
- Conclusions



Active Messages' Benefit

- Asynchronous communication
- No buffering
- Improved Performance
- Handlers are kept simple



Thanks!

Questions?