AN O/S PERSPECTIVE ON NETWORKS

Adem Efe Gencer[1]

October 4th, 2012

[1]Department of Computer Science, Cornell University

Papers

- Active Messages: A Mechanism for Integrated Communication and Control, Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser. In Proceedings of the 19th Annual International Symposium on Computer Architecture, 1992.
- U-Net: A User-Level Network Interface for Parallel and Distributed Computing, Thorsten von Eicken, Anindya Basu, Vineet Buch and Werner Vogels. 15th SOSP, December 1995.

Authors - I

- Thorsten von Eicken
  - Founder of RightScale
  - Ph.D, CS at University of California, Berkeley
  - Assistant Prof. at Cornell (1993-1999)
- David E. Culler
  - Chair, Electrical Engineering and Computer Sciences at Berkeley
- Seth Copen Goldstein
  - Associate Professor at Carnegie Mellon University
- Klaus Erik Schauser
  - Associate professor at UCSB

Authors - II

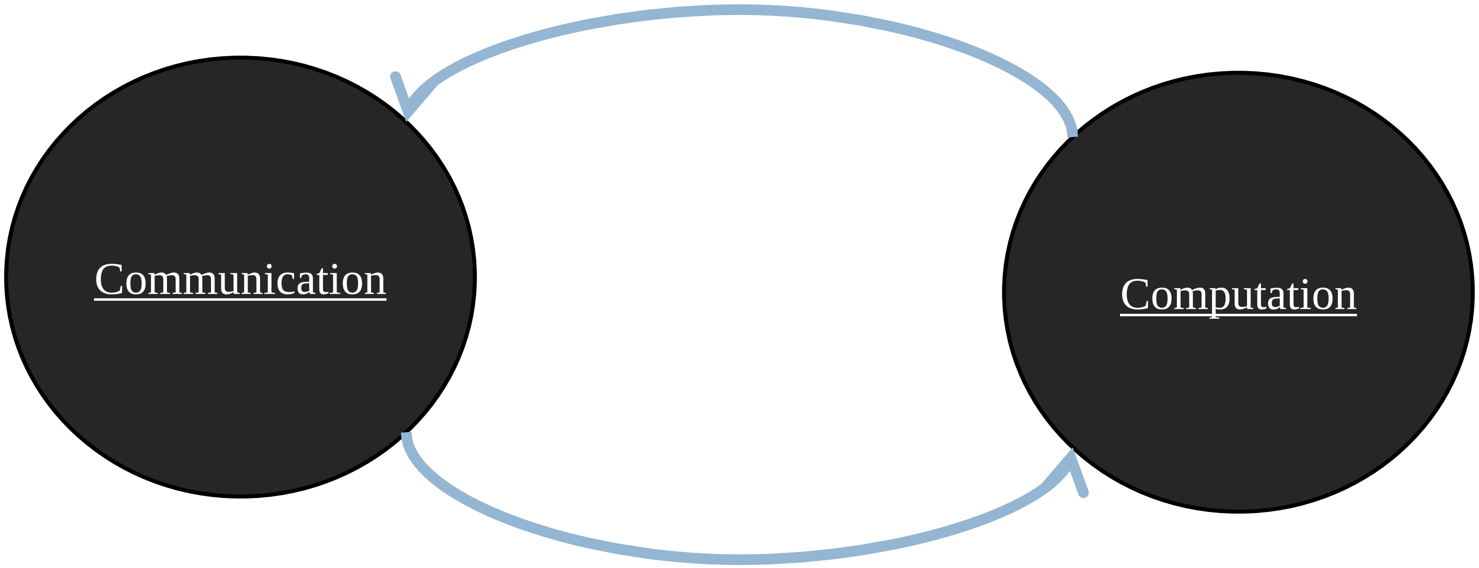
- Thorsten von Eicken
- Anindya Basu
  - Advised by von Eicken
- Vineet Buch
  - Google
  - MS from Cornell
- Werner Vogels
  - Research Scientist at Cornell (1994-2004)
  - CTO and vice President of Amazon

Active Messages

- Introduction
- Preview: Active Message
  - Traditional programming models
- Active Messages
- Environment
- Split-C
- Message Driven Model vs. Active Messages
- Final Notes

Introduction

- Inefficient use of underlying hardware
  - Poor overlap between communication and computation
  - High communication overhead

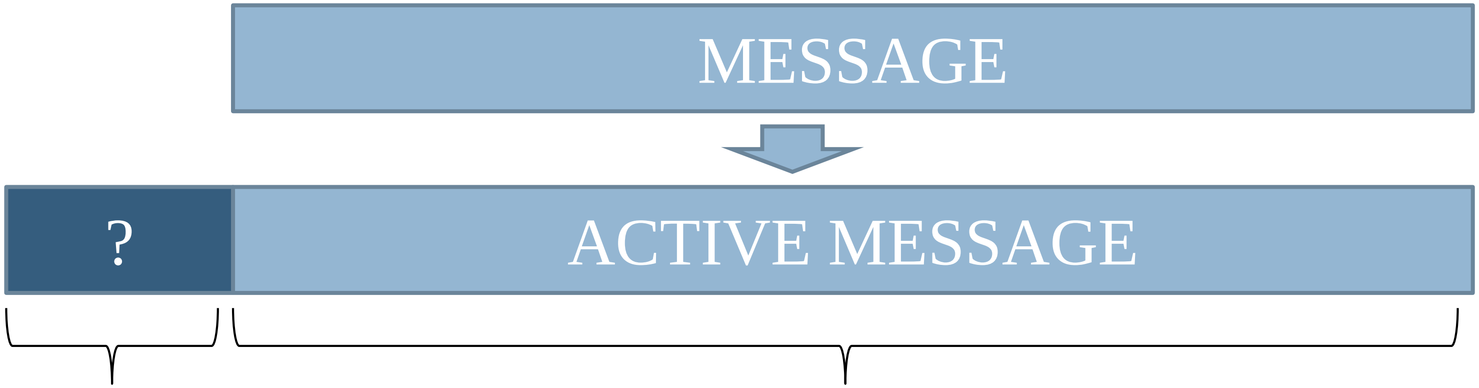


Introduction: Objective

- Eliminate inefficiency by increasing the overlap between *communication* & *computation*!

Preview: Active Message

- Solution: A novel asynchronous communication mechanism! **<u>Active messages</u>**



<u>HEADER</u>

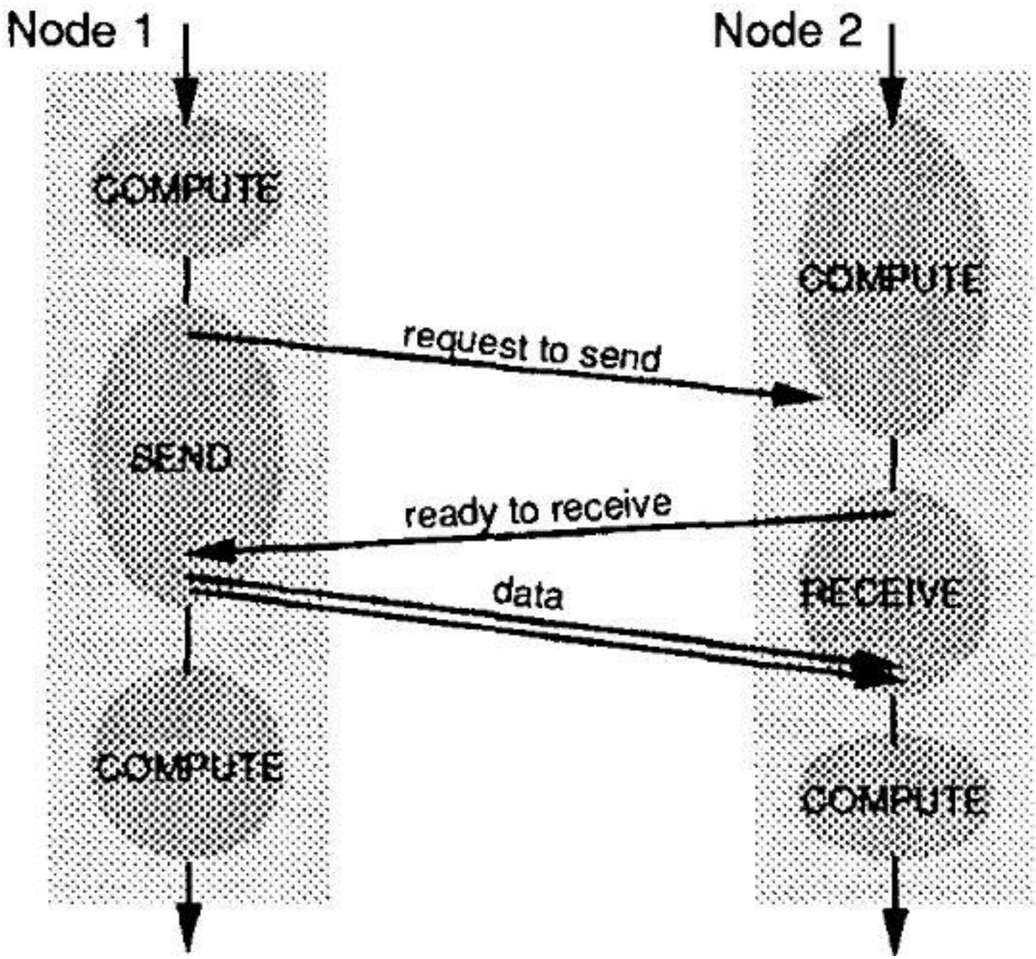
Address of a userspace handler

<u>BODY</u>

Argument of userspace handler

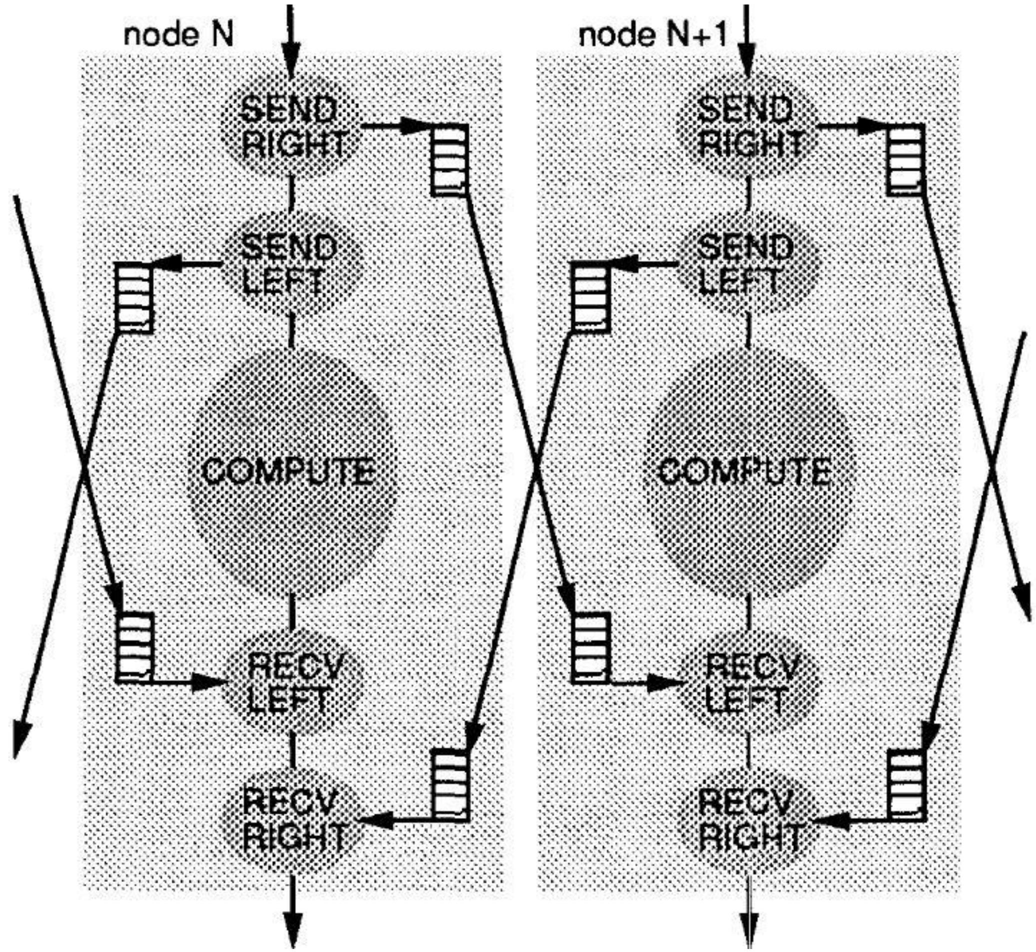
Traditional Programming Models

- Synchronous comm.
  - 3-phase protocol
  - Send / receive are blocking
- Simple
- No buffering
- High comm. latency
- Underutilized network bandwidth



Traditional Programming Models

- Asynchronous comm.
  - Send / receive are non-blocking
  - Message layer buffers the message
- Communication & Computation can overlap



Traditional Programming Models

	Synchronous	Asynchronous
PROS	No Buffering	Overlap
CONS	No Overlap High Comm. Latency	Buffering

Active Messages

Active Messages

- Requires SPMD programming model
- Key optimization: not buffered
  - Except for network transport buffering
  - For large messages buffering is needed on sender side
  - Receiving side pre-allocates structures
- Handlers do not block - deadlock avoidance

Active Messages

- Sender
  - Packs the message with a header containing handler's address
  - Pass the message to the network
  - Continue computing
- Receiver
  - Message arrival invokes handler
  - Handler Extracts the data & interrupts computation
  - Passes it to the ongoing computation

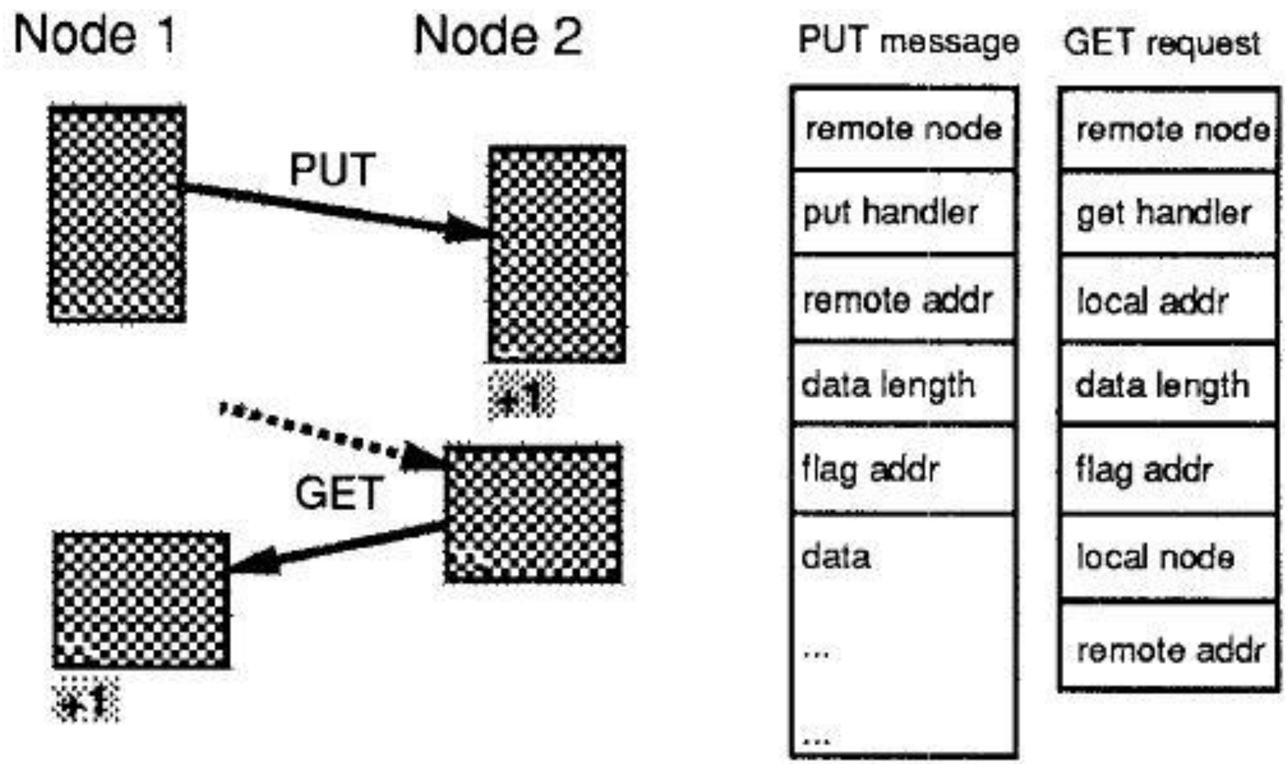
- Implementation of active messages on different platforms is not the same (nCube/2 vs. CM-5).

Environment

- Implementation on parallel supercomputers
  - *nCUBE/2*
    - Binary hypercube network
    - NI with 28 DMA channels
  - *CM-5 (connection machine-5)*
    - Hypertree network
- Split-C
  - A parallel extension of C

CM-5 machine of NSA

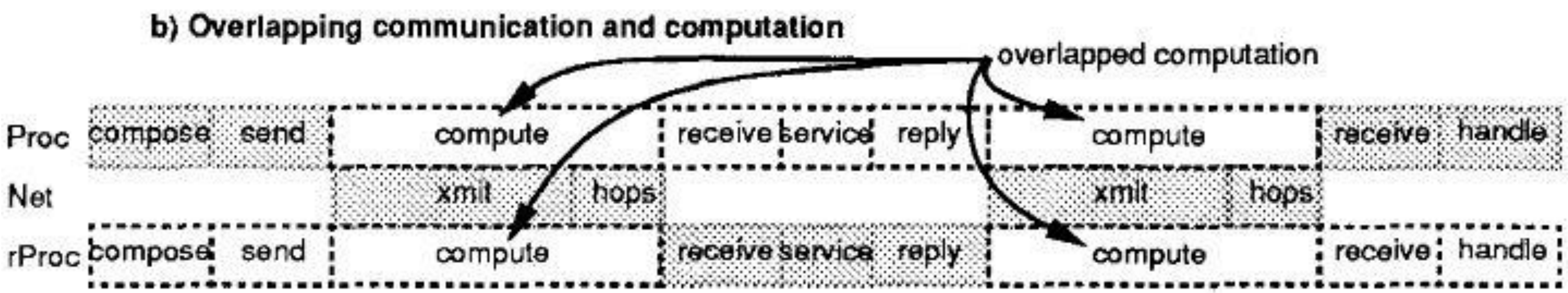
Split-C: PUT and GET



- Nonblocking
  - Message handler
  - Message formatter
- Used in matrix multiplication to show the performance of active message

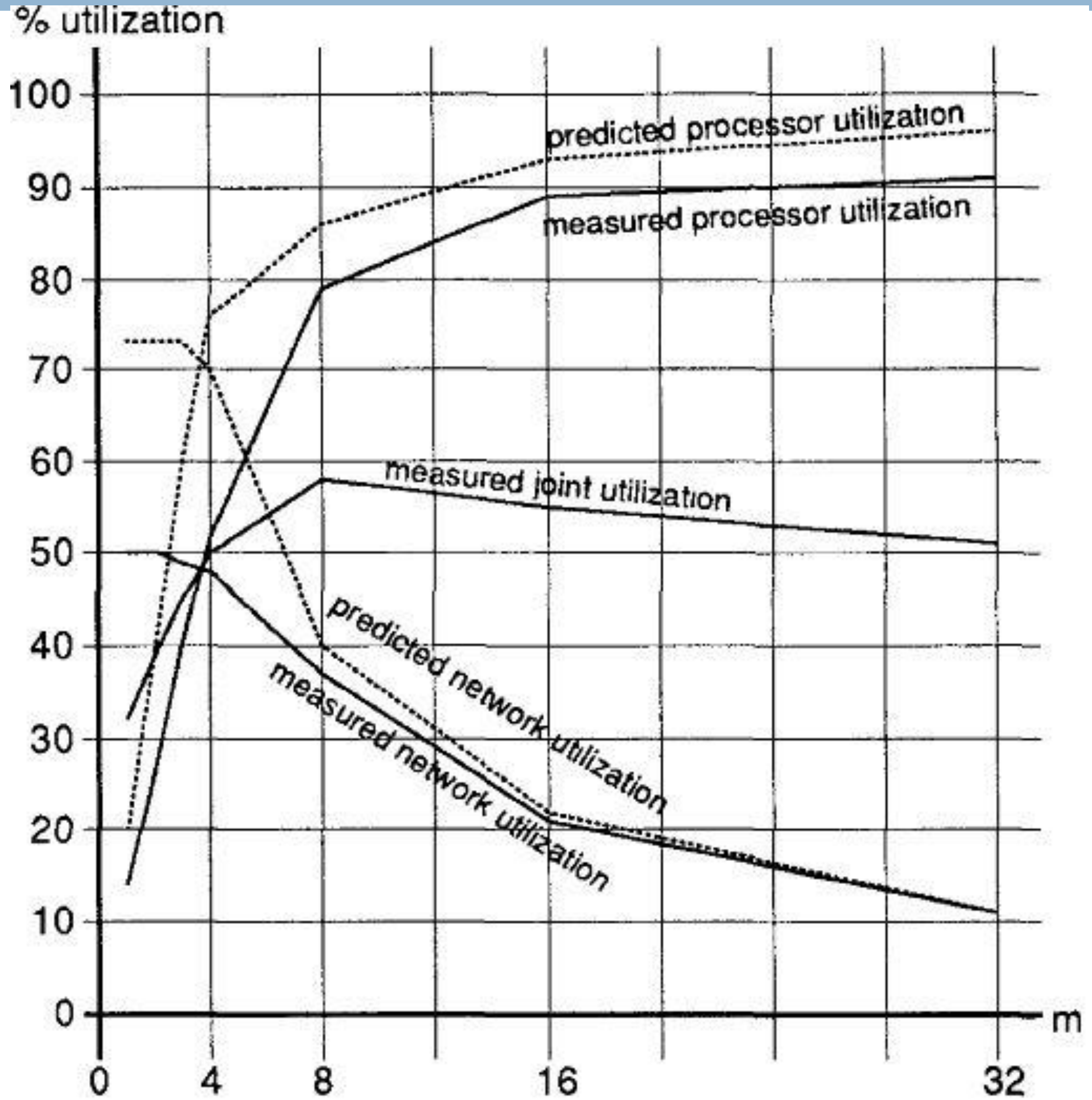
Split-C: GET

- Matrix multiply achieves 95% performance in nCUBE/2 configuration.



- Xmit: time to inject the message info the network
- Hops: time for the network hops

Split-C: Matrix Multiply



- **m**: # of columns per processor
- # of nodes is constant (N=128)

Message Driven Model vs. Active Messages

- Message driven model
  - Computation in message handlers
  - Handler may suspend
  - Memory allocation & scheduling on message arrival
- Active Messages
  - Computation in background (handler extracts message)
  - Immediate execution

Final Notes

- Not a new parallel programming paradigm!
  - A primitive communication mechanism
  - Used to implement them efficiently
- The need for userspace handler address leads to programming difficulty
- Requires SPMD model (but not required on some modified versions)

U-Net

- Introduction
- U-Net
  - Remove the Kernel from Critical Path
- Communication in U-Net
- Protection
- Zero Copy
  - True zero copy vs. zero copy
- Environment
- Performance
- Final Notes

Introduction

- Low bandwidth and high communication latency
  - No longer hardware issues!
  - Problem is the message path through the *kernel*
- Small messages
  - Common in various applications
  - Requires low round-trip latency

Latency = processing overhead + network latency

Introduction: Objective

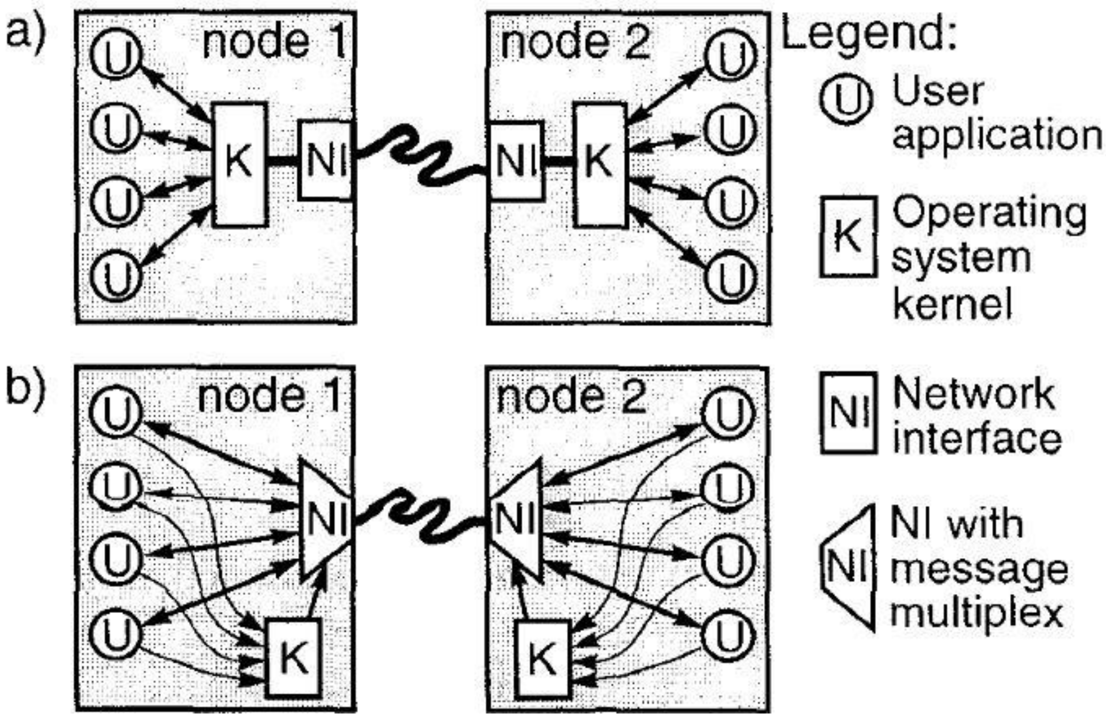
Make communication ***FASTER***!

U-Net

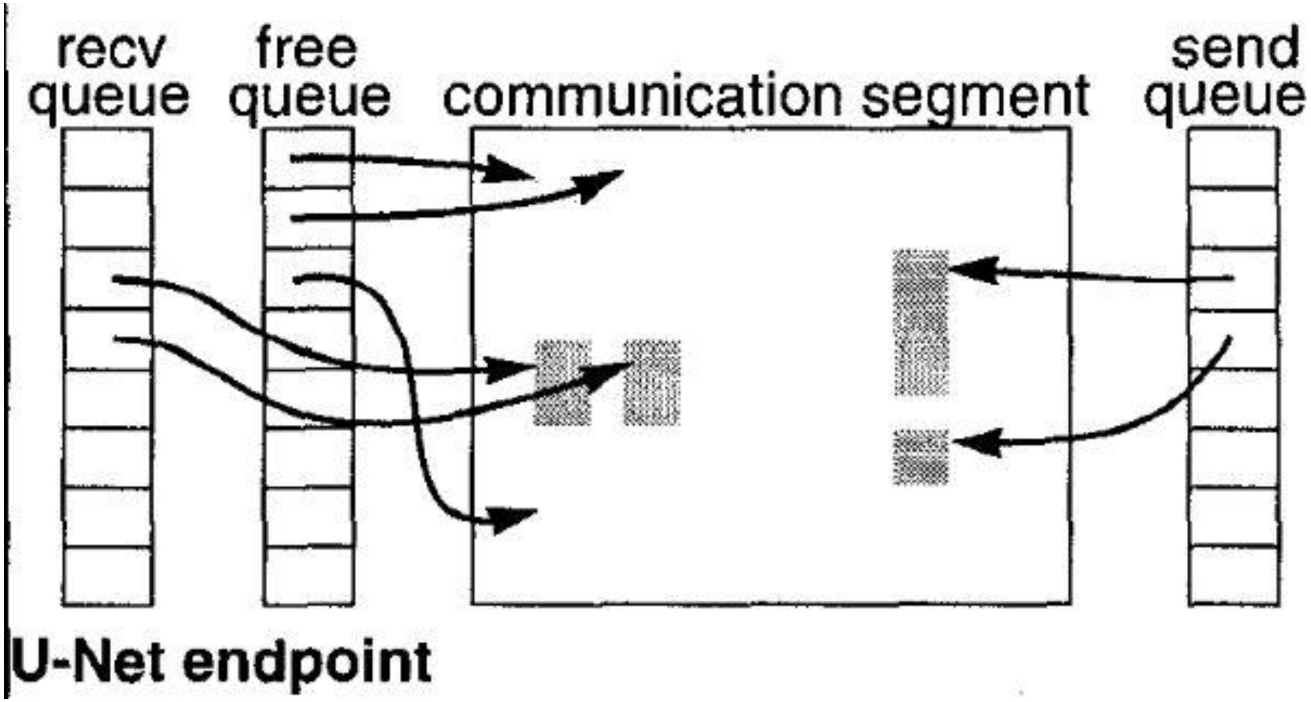
- Low communication latency
- High bandwidth
  - even with small messages!
- Support for workstations with off-the shelf network
- Keep providing full protection
  - Kernel controlled channel setup / tear-down
- TCP, UDP, Active Messages, … can be implemented

Remove the kernel from critical path

- Solution: Enable user-level access to NI to implement user-level communication protocols
  - Does not require OS modification or custom HW
  - Provides higher flexibility (app. specific protocols!)



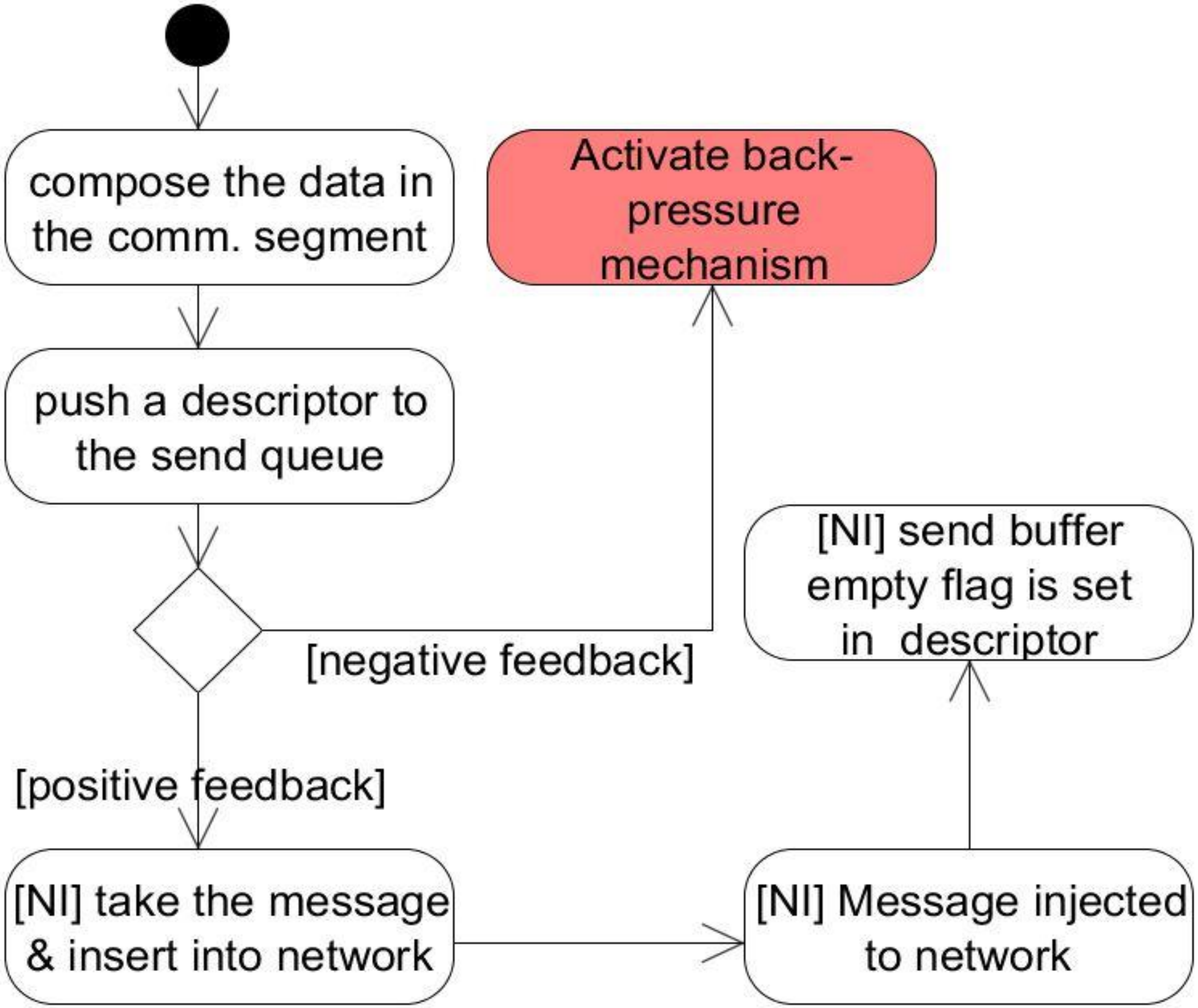
Communication in U-Net



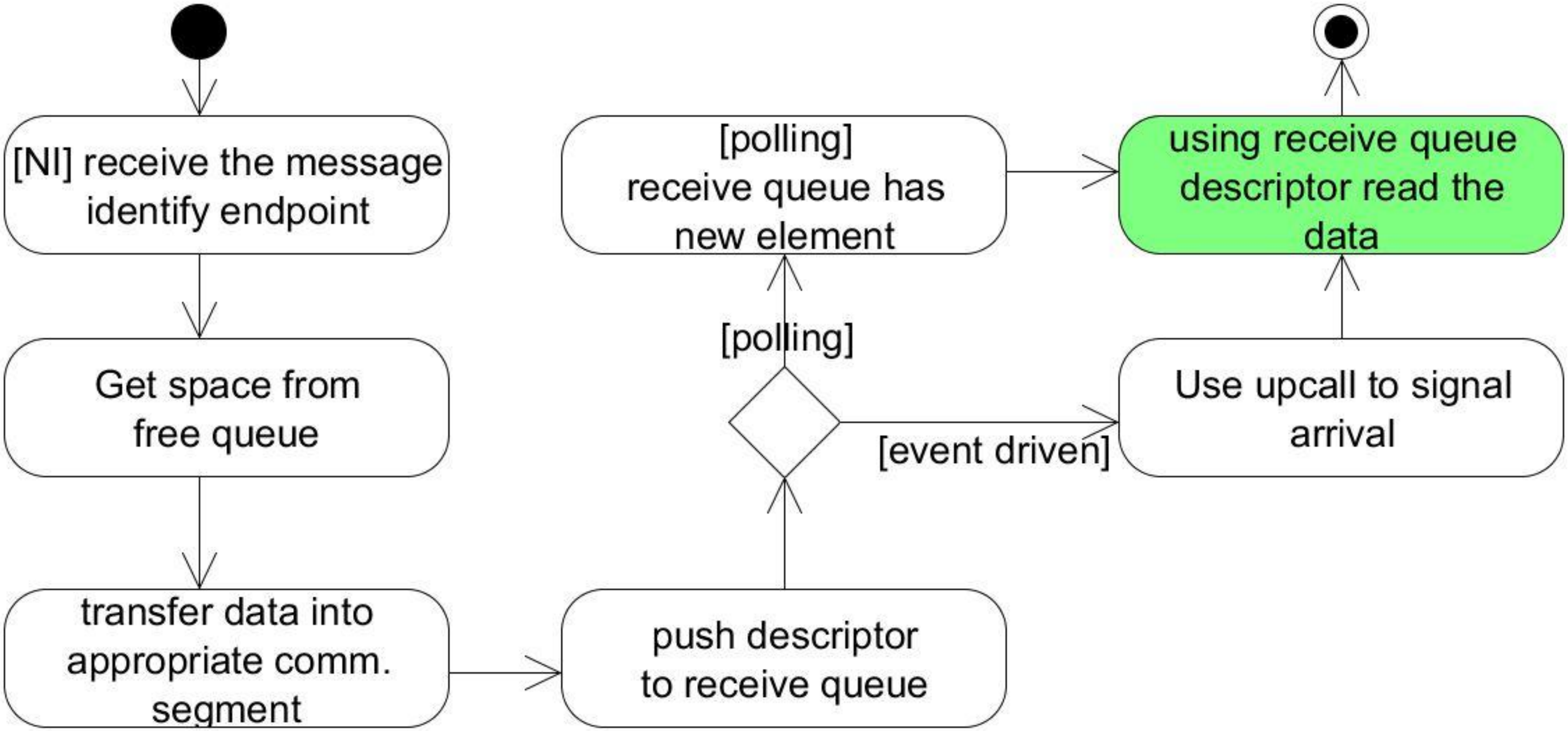
Each process creates endpoints to access network

- **U-net endpoint:** application handle into the network
- **Communication segment**: regions of memory that hold message data
- **Send/recv/free queues**: hold message descriptors

Communication in U-Net: Send



Communication in U-Net: Receive



Protection

- Owning process protection:
  - Endpoints
  - Communication Segments
  - Send/Receive/Free Queues

Accessible by owning process only!

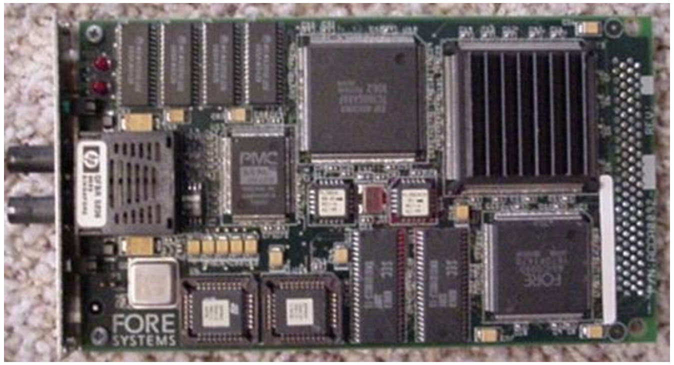
- Tag protection:
  - Outgoing message
    - tagged with originating address
  - Incoming message
    - delivered to correct destination point only!

Zero Copy

- Base level U-Net (zero copy)
  - Send/receive needs a buffer (not really "zero" copy!)
  - Requires a copy between app. data structures and a buffer in the comm. segment
- Direct Access U-Net (true zero copy)
  - Spans on entire address space
  - Has special hardware requirements

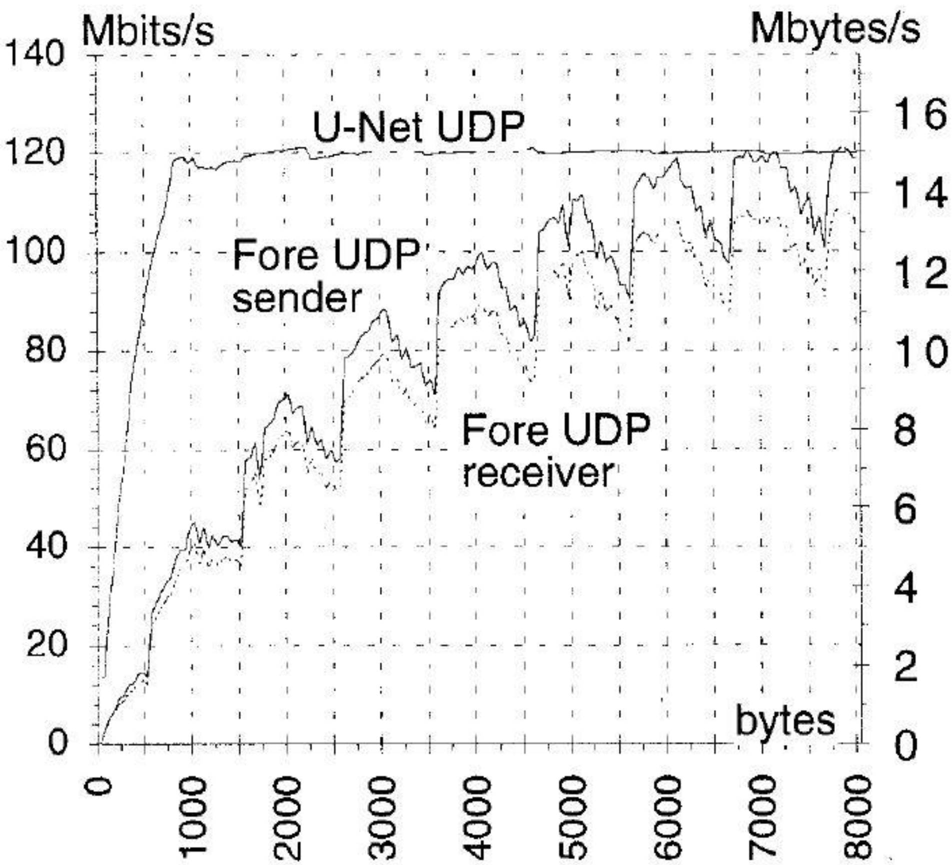
Environment

- Implementation on off-the-shelf hardware platform
  - *SPARCStations*
    - Fore SBA-100 NIC
    - Fore SBA-200 NIC
- SunOS 4.1.3
  - BSD-based Unix OS
  - Ancestor of Solaris (after version 5.0)
- Split-C
  - A parallel extension of C

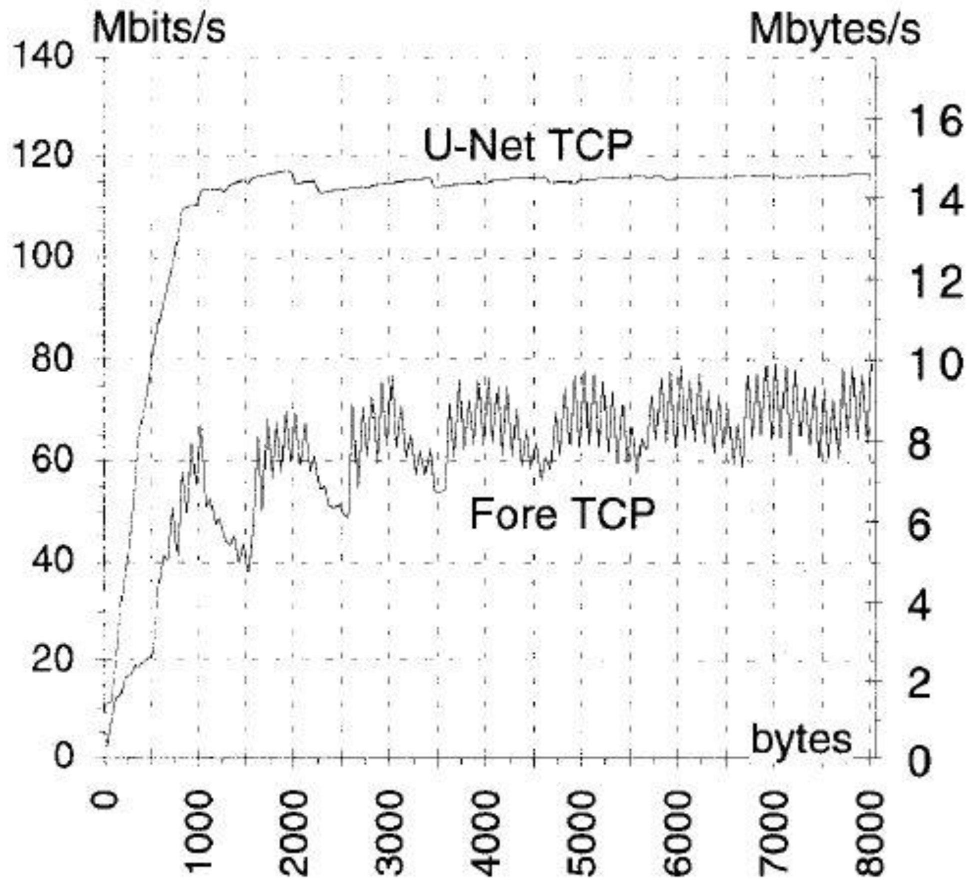


Fore SBA-200 NIC

Performance: UDP vs. TCP



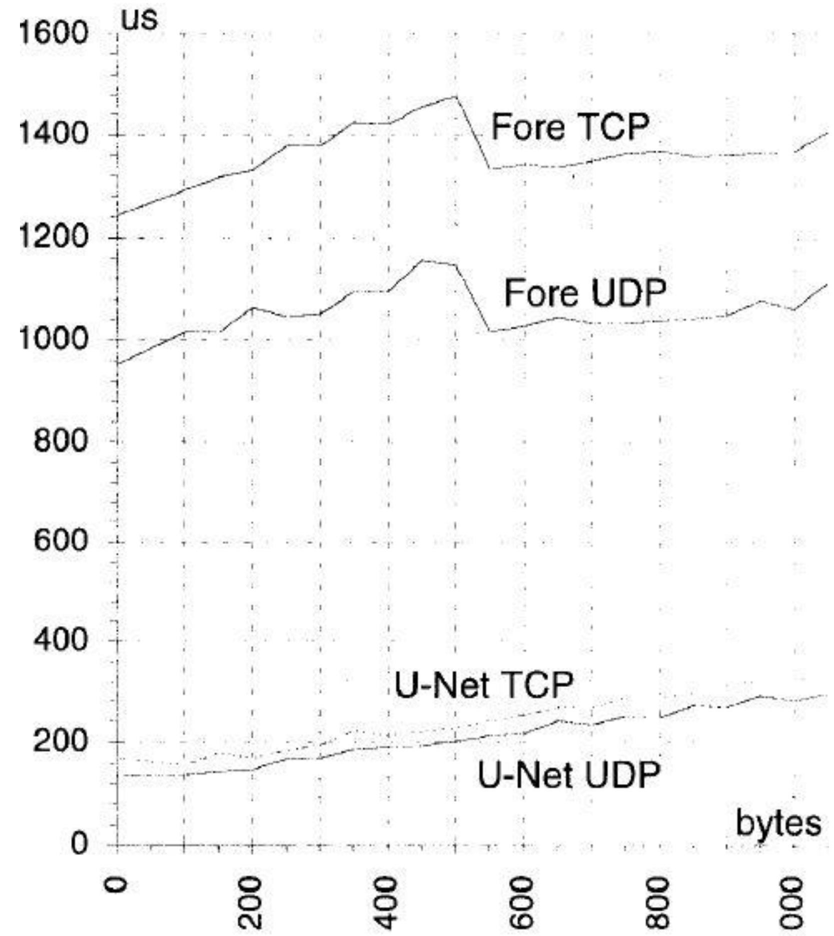
UDP bandwidth as a function of message size



TCP bandwidth as a function of data generation by application

Performance: UDP vs. TCP

- Fast U-Net roundtrips:
  - Apprx. 7x speedup for TCP
  - Apprx. 5x speedup for UDP
- Fast U-Net TCP roundtrips let use of small window size.



End-to-end round-trip latencies as a function of message size

Final Notes

- Low comm. latency
- High bandwidth
- Good flexibility

Protocol	Round-trip latency	Bandwidth 4K packets
Raw AAL5	65µs	120Mbits/s
Active Msgs	71µs	118Mbits/s
UDP	138µs	120Mbits/s
TCP	157µs	115Mbits/s
Split-C store	72µs	118Mbits/s

Table 3: U-Net latency and bandwidth Summary.

Cluster of workstations vs. supercomputers

- Comparable performance using U-Net
- Additional system resources (parallel process scheduler, file systems, …) are needed