An O/S perspective on networks: Active Messages and U-Net

Theo Jepsen

Cornell University

17 October 2013

Brief History Overview of Parallel Computing

- 1962: Burroughs Corporation introduces D825 (4 CPUs connected to 16 memory modules)
- 1967: Amdahl's Law: predicting the maximum speedup when new CPUs are added
- 1969: Multics introduced (8 CPUs)
- 1970: C.mmp multiprocessor (16 CPUs)
- 1976: ILLIAC IV (up to 256 CPUs): "perhaps the most infamous of Supercomputers"

Source: `en.wikipedia.org`

Networked Parallel Computers

- Tightly coupled multiprocessor machines
- Task parallelism
- Communication should be responsive
- Communication should not have high overhead

Connection Machine 5 (CM-5)

- Hypercubic routing network
- Intended for AI, but used for scientific computing
- communication channel over a hypercubic routing network

Source: Photographer: Tom Trower, 1993

Networking Problems Faced by Parallel Computers

- High-latency
- Throughput optimization based on big messages
- Disconnect between software and hardware design
- Synchronous messaging

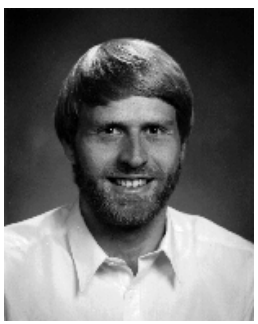
Active Messages: a Mechanism for Integrated Communication and Computation

In Proceedings of the 19th Annual International Symposium on Computer Architecture, 1992

Thorsten von Eicken

David E. Culler

Seth Copen Goldstein



Klaus Erik Schauser

Outline

- Motivation for Active Messages
- Solution – how do they achieve the goals
- Implementation
- Evaluation of Active Messages
- Perspective

Active Messages: Motivation

- CPU design based on raw performance, not so much on networking
- Synchronous messaging is slow due to high latency
- Communication and computation should be combined

Active Messages: Solution

- Take advantage of fast DMA and specific network interface capabilities
- Use asynchronous messaging paradigm; latency becomes less of a problem
- Implement queues to buffer data so that it is ready for computation

Asynchronous Messaging

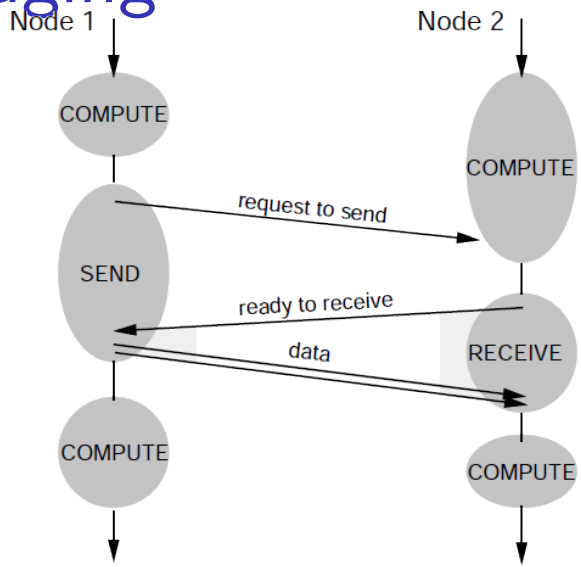


Figure 1: *Three-phase protocol for synchronous send and receive. Note that the communication latency is at best three network trips and that both send and receive block for at least one network round-trip each.*

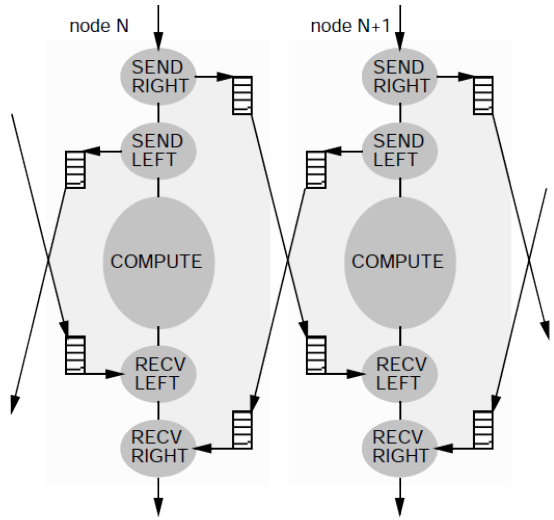


Figure 2: *Communication steps required for neighboring processors in a ring to exchange data using asynchronous send and receive. Data can be exchanged while computing by executing all sends before the computation phase and all receives afterwards. Note that buffer space for the entire volume of communication must be allocated for the duration of the computation phase!*

Source: same paper

Asynchronous Messaging

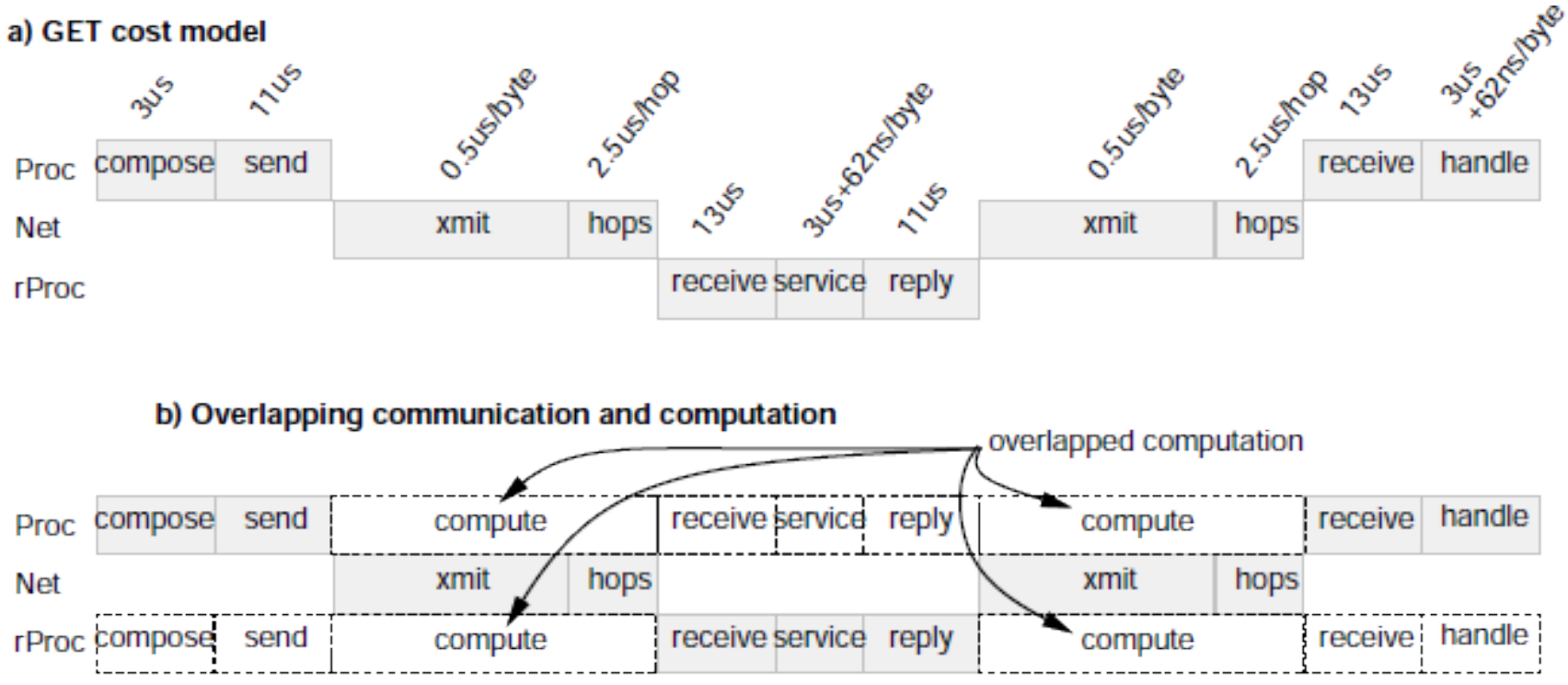


Figure 6: *Performance model for* GET. Compose *accounts for the time to set-up the request.* Xmit *is the time to inject the message into the network and* hops *is the time taken for the network hops.* Service *includes for copying the data into the reply buffer and* handle *for the time to copy the data into the destination memory block.*

Source: same paper

Utilizes resources (almost) as predicted

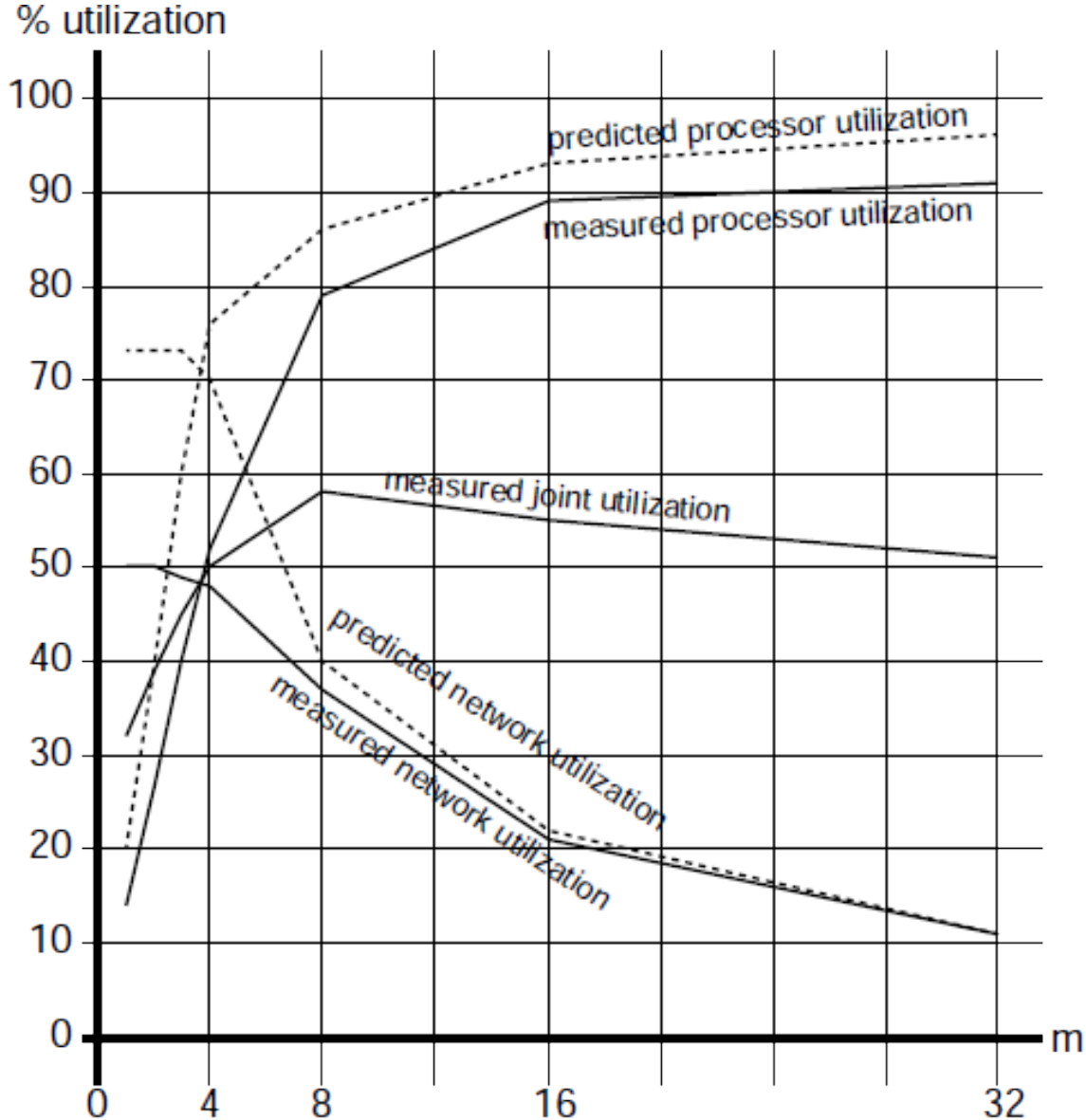


Figure 5: *Performance of Split-C matrix multiply on 128 processors compared to predicted performance using the model shown in Figure 6.*

Source: same paper

Active Messages: Conclusion

- Case for hardware to better support networking
- Good basis for parallel computing (Split-C)
- Design is specific to parallel computing

Active Messages: Perspective

- What happened to Active Messages?
- Where is parallel computing? Niche computing?
- Change in hardware? We stopped increasing CPU frequency

More Than Low Latency

- Low latency communication is essential in parallel computing
- What happens with multiple processes?
- What about isolation?
- Do you let the kernel handle isolation?

Networking Through Kernel

- Kernel abstracts networking device
- Processes must use the underlying protocols
- Kernel responsible for management and isolation (policy)

User Space Networking

- Remove kernel from critical path
- Reduce the latency between network interface and process
- Program can decide protocols

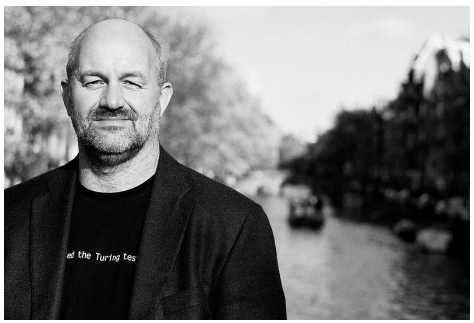
U-Net: A User-Level Network Interface for Parallel and Distributed Computing

15th SOSP, December 1995.

Thorsten von Eicken

Anindya Basu

Vineet Buch



Werner Vogels

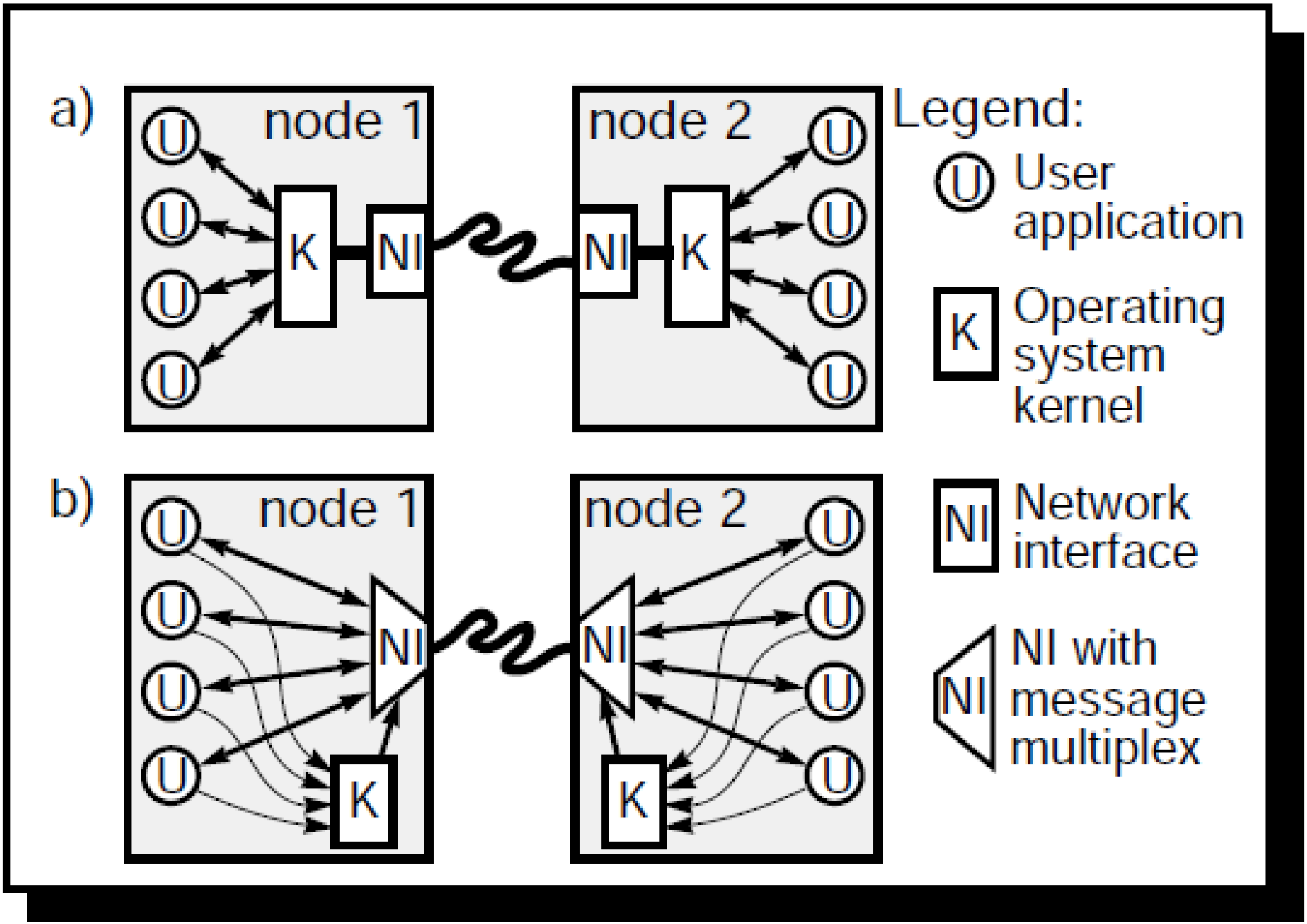
Outline

- Motivation/Goals for U-Net
- Design
- Implementation
- Evaluation of U-Net
- Perspective

U-Net Goals

- Low latency and high bandwidth with small messages
- Application access to underlying protocols
- Practical integration with existing systems

Solution: virtual network interfaces (endpoints)



Source: same paper

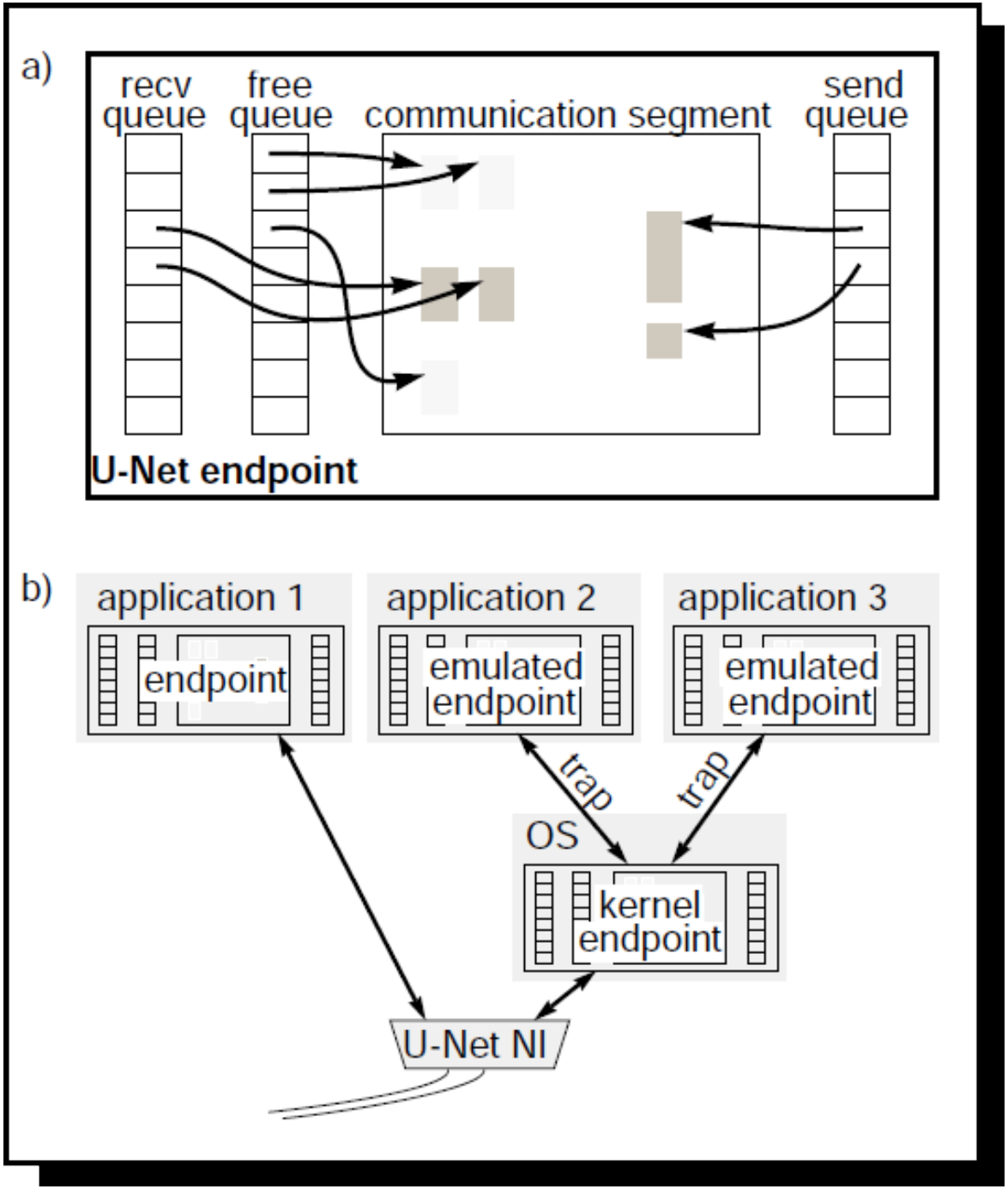


Figure 2: U-Net building blocks.
a) *Endpoints* serve as an application's handle into the network, *communication segments* are regions of memory that hold message data, and message queues *(send/recv/free queues)* hold descriptors for messages that are to be sent or that have been received.
b) Regular endpoints are serviced by the U-Net network interface directly. Emulated endpoints are serviced by the kernel and consume no additional network interface resources but cannot offer the same level of performance.

Source: same paper

Better utilization of available bandwidth

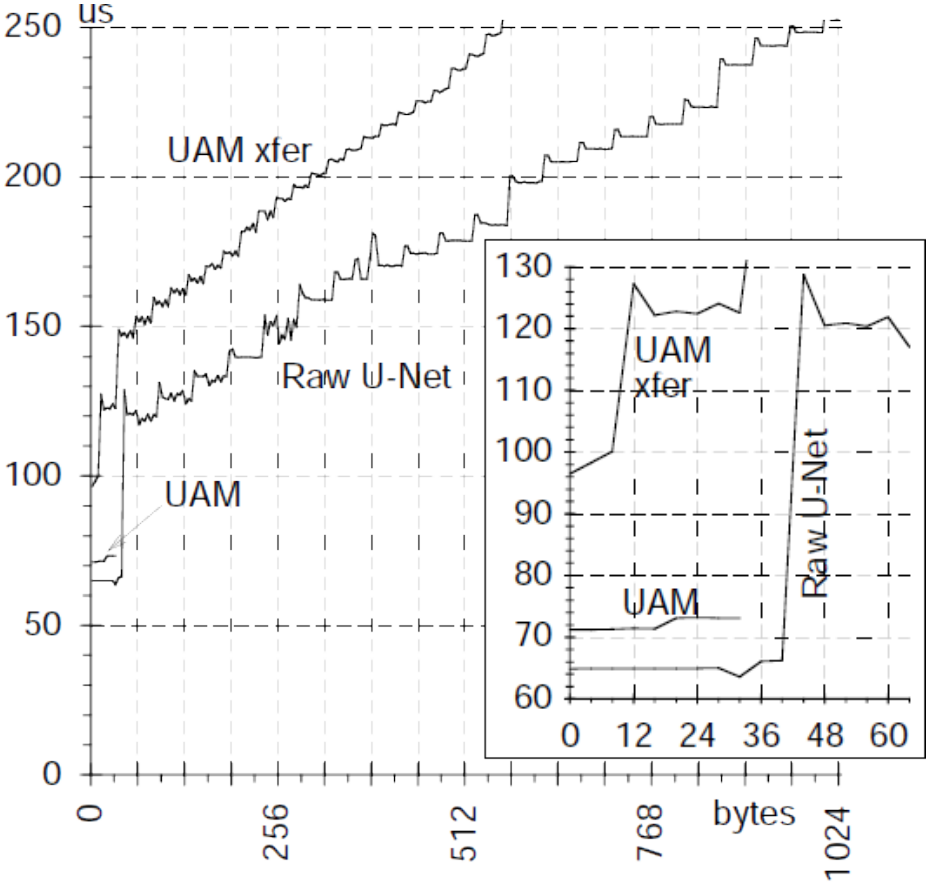


Figure 3: U-Net round-trip times as a function of message size. The *Raw U-Net* graph shows the round-trip times for a simple ping-pong benchmark using the U-Net interface directly. The inset graph highlights the performance on small messages. The *UAM* line measures the performance of U-Net Active Messages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

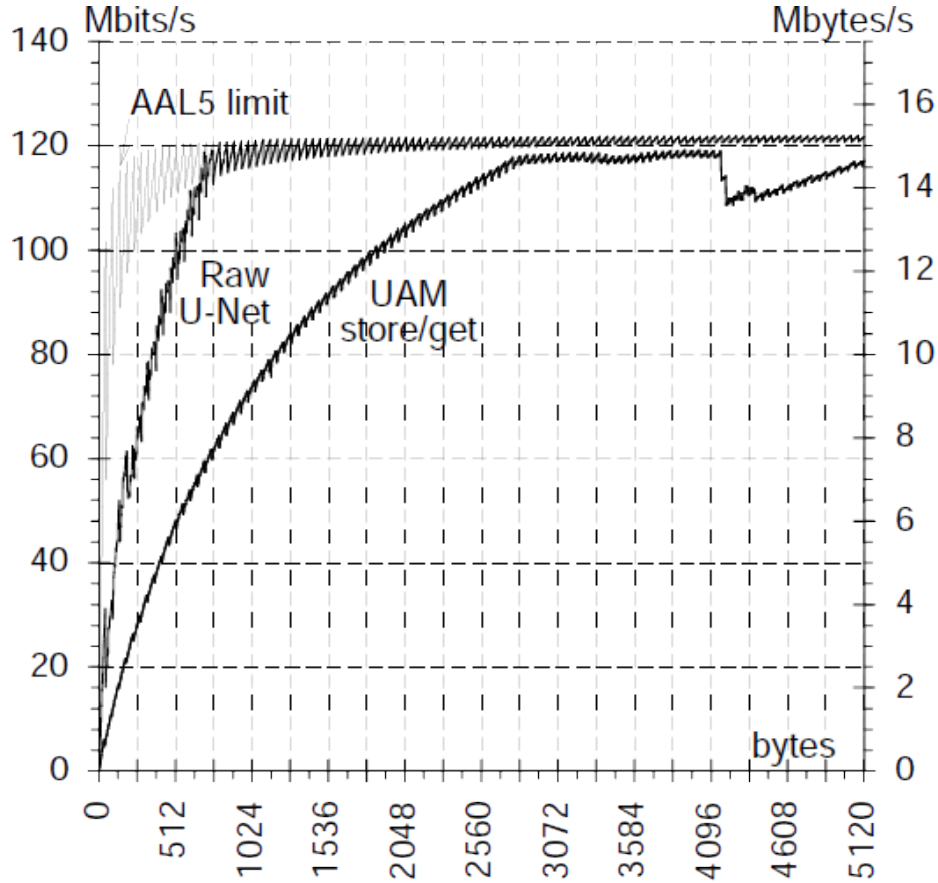


Figure 4: U-Net bandwidth as a function of message size. The *AAL-5 limit* curve represents the theoretical peak bandwidth of the fiber (the sawtooths are caused by the quantization into 48-byte cells). The *Raw U-Net* measurement shows the bandwidth achievable using the U-Net interface directly, while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

Source: same paper

Supports other protocols

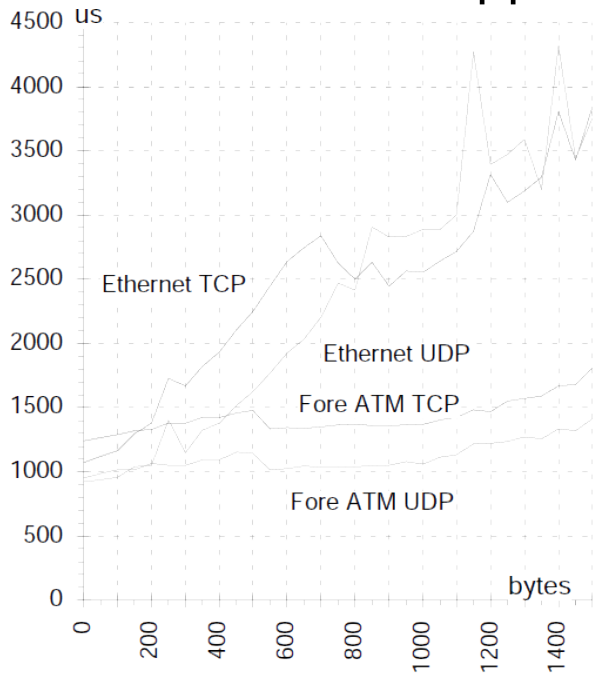


Figure 6: TCP and UDP round-trip latencies over ATM and Ethernet. as a function of message size.

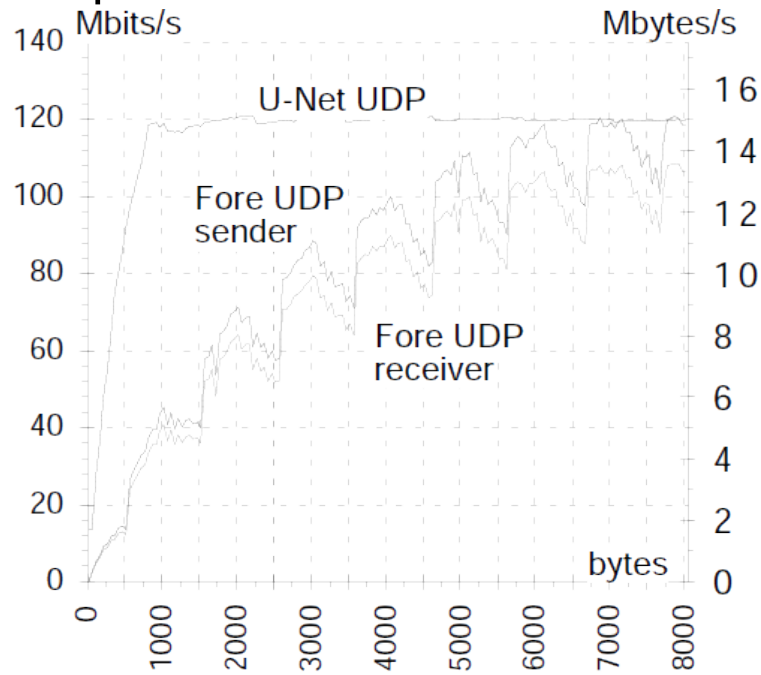
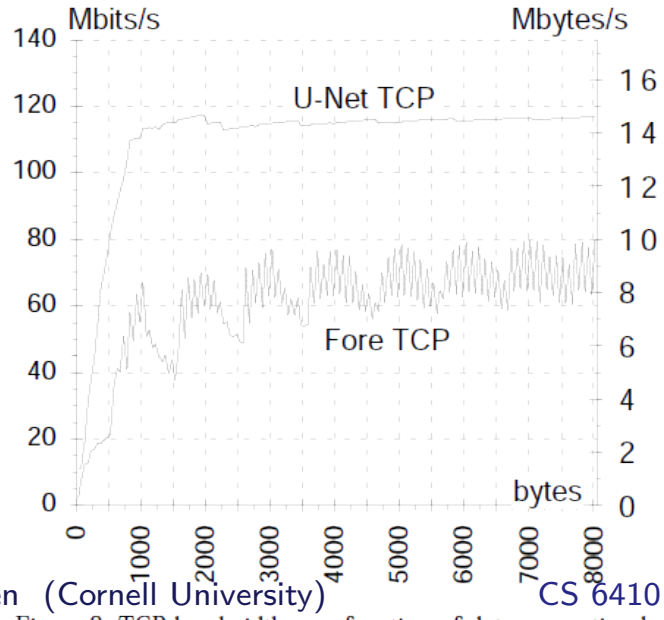
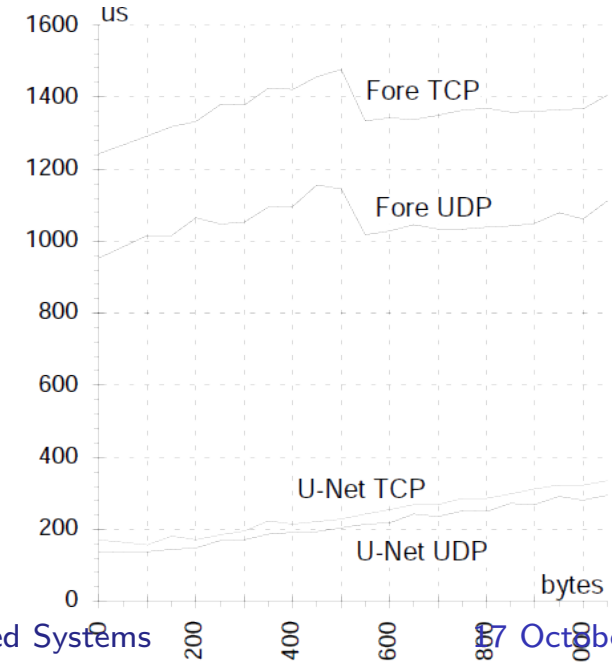


Figure 7: UDP bandwidth as a function of message size.





Source: same paper

Performs well compared to other systems (*run on different h/w)

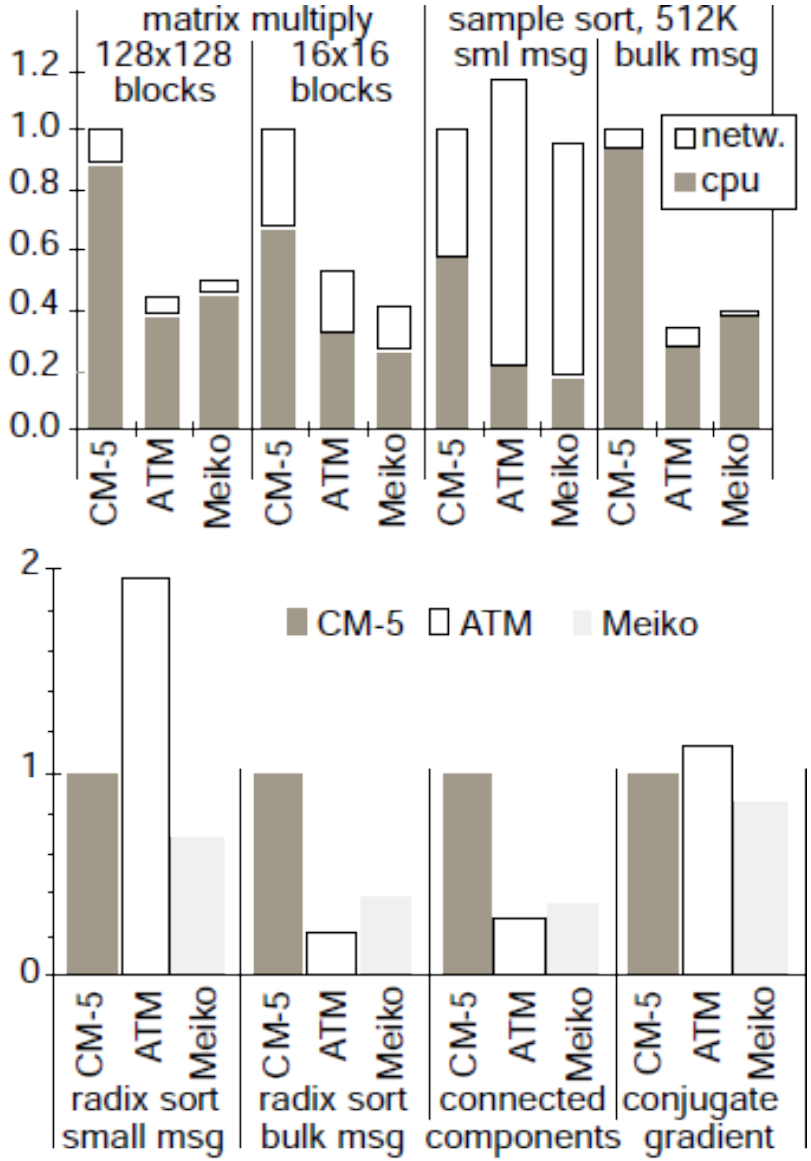


Figure 5: Comparison of seven Split-C benchmarks on the CM-5, the U-Net ATM cluster, and the Meiko CS-2. The execution times are normalized to the CM-5 and the computation/communication breakdown is shown for three applications.

Source: same paper

U-Net: Conclusion

- Reasonable performance
- Is there process isolation?
- TCP/IP implementation should facilitate integration with other networks
- Adoption? Applicability in existing systems?
- Approach reminiscent of a microkernel's: move networking to userspace

U-Net: Perspective

- Is there sufficient granularity of network resources?
- Do we have user-level interfaces today?

What happened to U-Net?

- Virtual Interface Network (1997) – userspace zero-copy networking
- RDMA – Remote DMA: zero-copy over network done by network card
- Myrinet (1995): two fibre optic cables, packet switching, lower protocol overhead than Ethernet, low latency
- Infiniband (1999): high speed I/O (storage)
- Virtualization: hypervisor kernel-bypass

Source: en.wikipedia.org

Active Messages and U-Net: Comparison

- Both are in parallel computing environment
- However, U-Net also focused on TCP and UDP
- Do we use this today?
- What happened to parallel computing

Perspective

- Less tightly-coupled processing today
- Active Networks in mid 90s
- Disliked for disobeying end-to-end argument
- 15 years later: SDN
- Is parallel computing still important today?