Thorsten von Eicken
tve@cs.cornell.edu

Dept of Computer Science
Cornell University

THE U-NET USER-LEVEL NETWORK ARCHITECTURE

**or: it's easy to buy high-speed networks
but making them work is another story**

NoW retreat
June 7th-9th,1995

Joint work with Werner Vogels, Anindya Basu, and Vineet Buch

Why ATM & U-Net goals

- Why ATM?
  - could be decent LAN
  - standard
  - ok if one ignores 99% of the standards
  - ok if one ignores 99% of the vendor software
  - shoot yourself in the foot and then try to run?
    - yup, but building your own hardware is even worse...
- Why U-Net?
  - need user-level access to NI (for all the good ol' reasons)
  - not everyone has bought into Active Messages (yet :-)
  - provide simple abstraction over network
    - send + receive queues
    - flexible buffers managed by user
    - enable (but not require) "true zero copy"
  - build Active Messages over U-Net efficiently

Experimental Set-up

Standard workstations

• 4 SS-20 @60Mhz, 64MB mem	each $19’250
• Total cost	**$77’000**

ATM network

• Switch chassis (1/4 Fore Systems ASX-200)	each $5500
• Network module for switch	$6000
• Network interfaces (4x Fore Sys SBA-200)	each $1900
• Fiber (4x lab fiber)	each $100
• Total cost	**$19’500**
• Fraction of total cost	20%

- list prices july ‘94

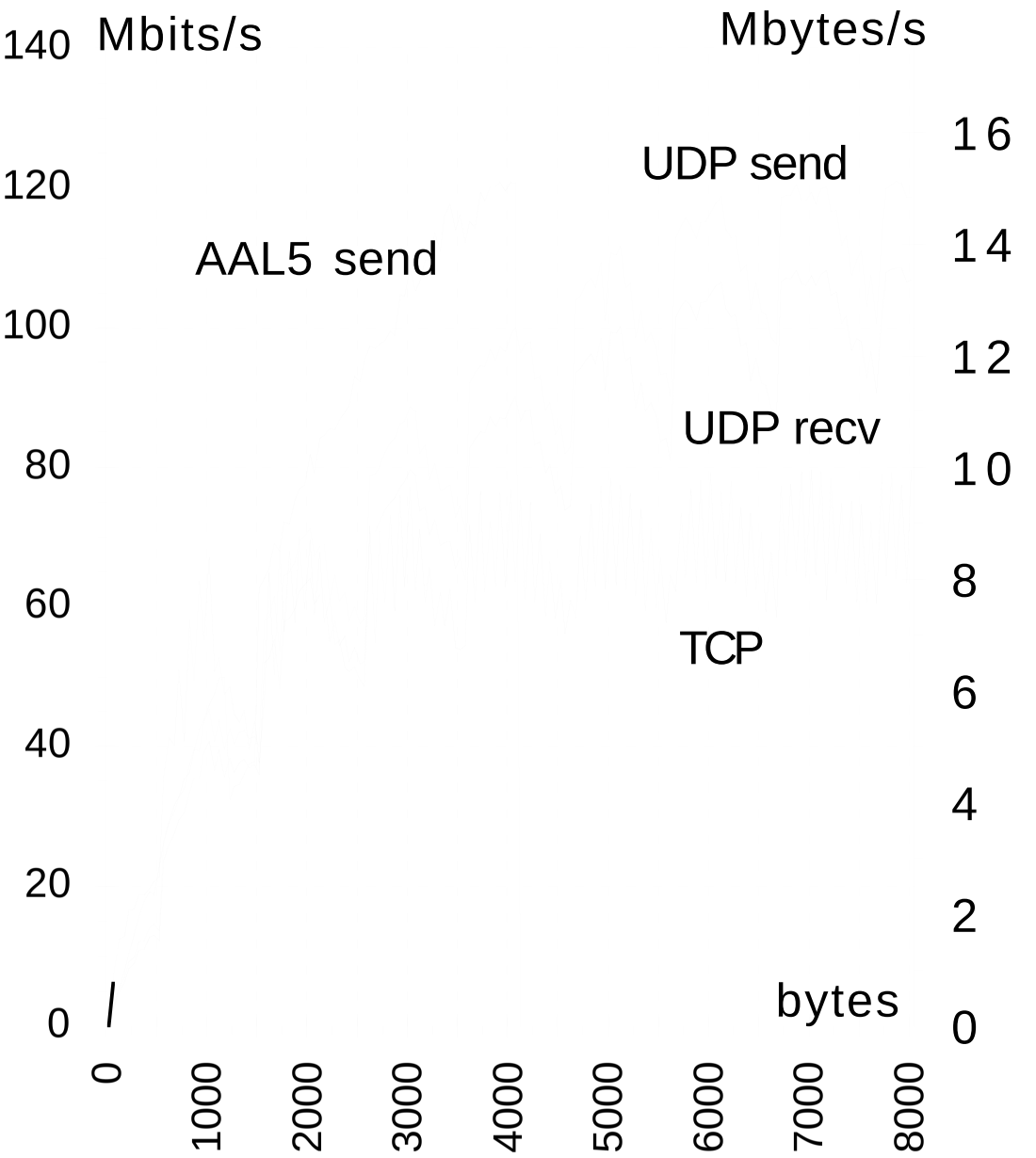
ATM bandwidth straight out of the box

Test 1:

- measure bandwidth
- using infinite stream out of cached buffers

Results:

- TCP at most 60% of bw
- UDP >80% only with large buffers, 20% drops if “wrong” buffer size
- AAL5 >80% only with buffers 3K<size<4k



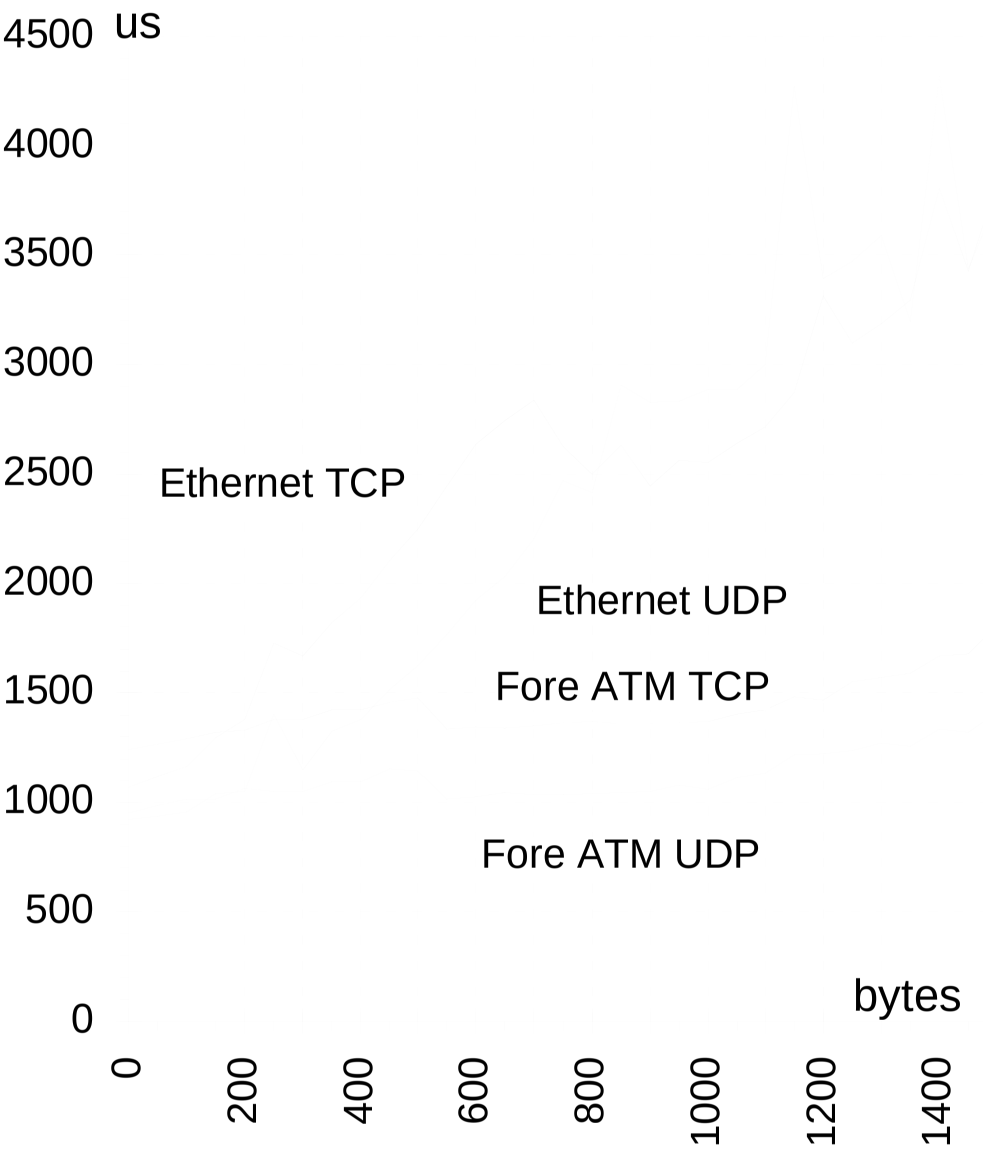
ATM latency straight out of the box

Test 2:

- measure round-trip
- using 1000 ping-pongs

Results:

- worse than ethernet,
- unless bandwidth matters



It must be the ATM network's fault!

Fore Systems ASX-200 switch:

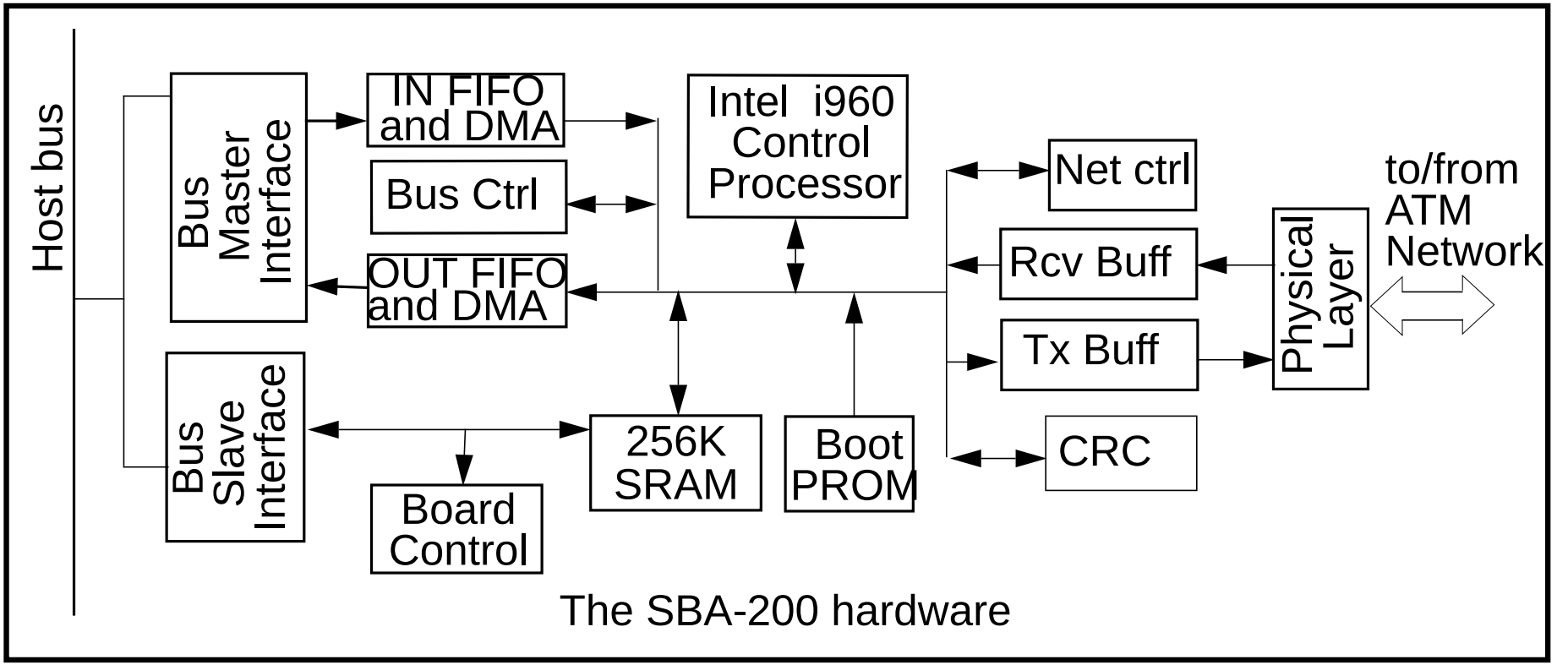
- up to 16 140/155Mbit ports
- full bandwidth broadcast architecture (equiv. to xbar)
- about 7µs latency per switch

140 Mbit fiber

- TAXI chip-set
- 175Mhz clock
- 4b/5b bit encoding -> 140Mbit/s
- 55bytes/cell -> 122.2Mbit/s payload bandwidth
- about 3µs to serialize cell
- about 1µs optical conversion delay (unlike when using SONET!)

It must be the ATM interface's fault!

SBA-200 Network Interface



- 25 Mhz i960 dual-issue processor
- burst DMA onto host bus
- AAL5 CRC calculation in hardware

... But no, it's the Software — Good ol' UNIX

UNIX Networking layers

- regular TCP/IP stack
- accounts for ~70% of the round-trip latency
  - > Werner Vogels will explain...

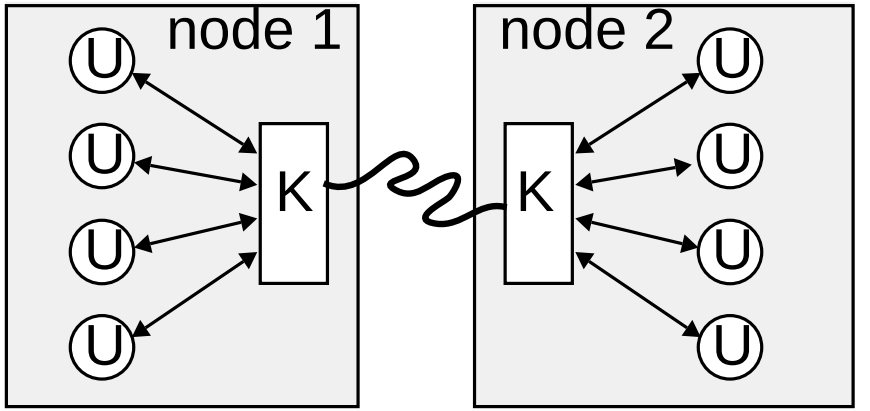
Device layer

- SBA-200 device driver
  - maps or copies mbufs into DMA space
  - sends by pointing SBA-200 at PDU descriptors
  - receives by handling interrupts and getting PDU descriptors
- SBA-200 firmware
  - deals with AAL5 segmentation/reassembly
  - sends queue of PDU descrs pointing to buffer descrs pointing to buffers
  - receives into queue of free buffer descrs pointing to buffers
  - provides queue of PDU descrs pointing to buffer descrs pointing to buffers + interrupt

U-Net: Basic Idea

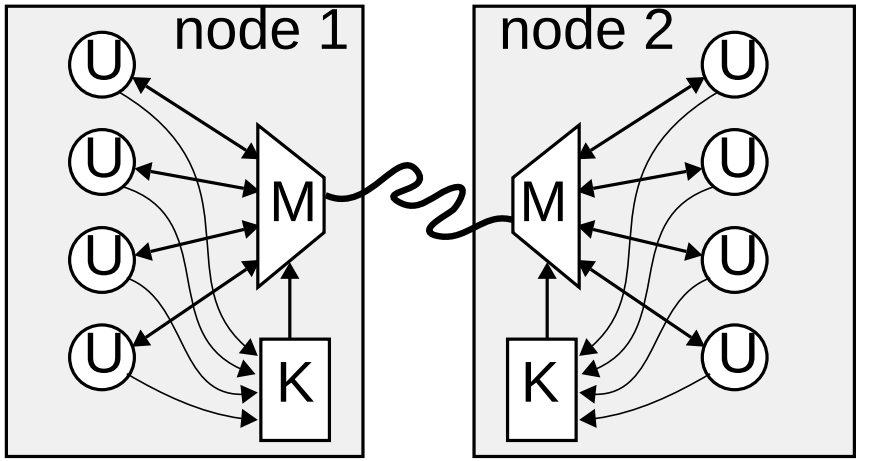
Traditional:

- kernel controls the network
- all communication goes via kernel



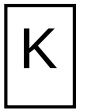
U-Net:

- applications access network directly via simple mux
- kernel only involved in connection set-up



Legend:

User application

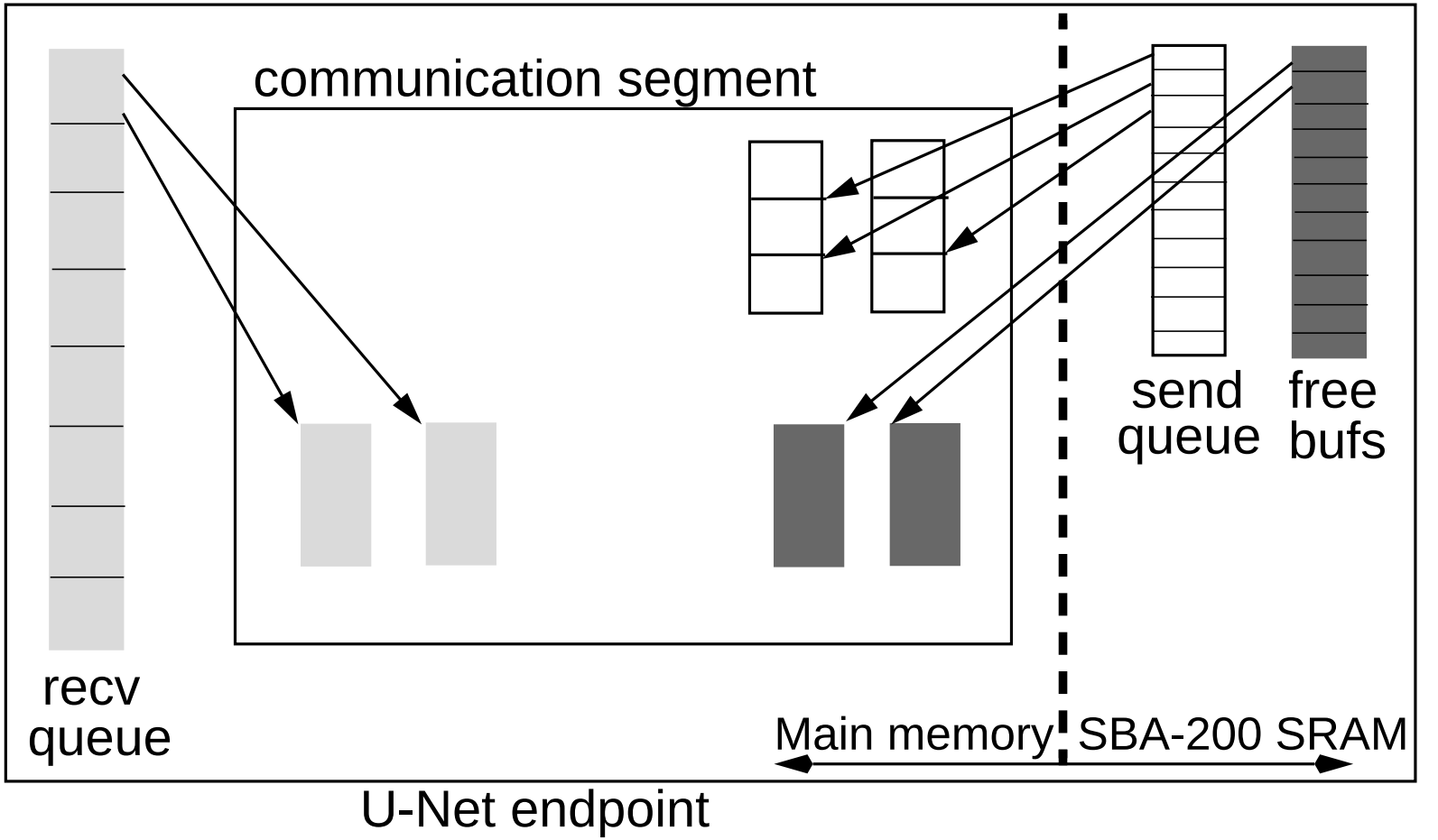


Operating system kernel

Message mux/demux

U-Net Building Blocks

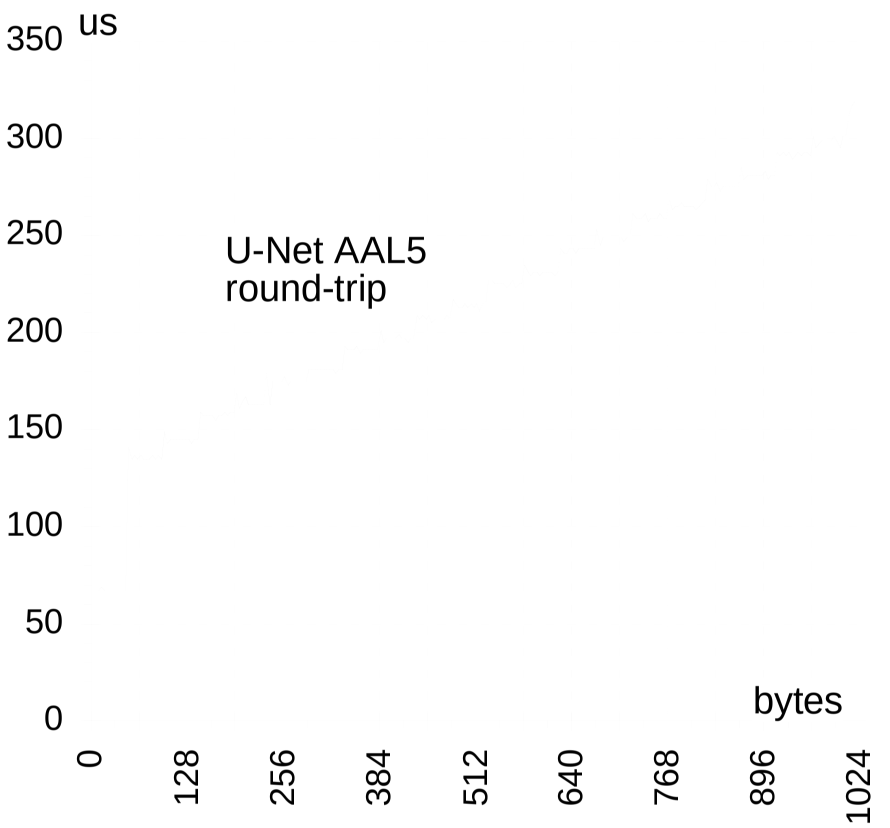
U-Net: User-Level Network Interface

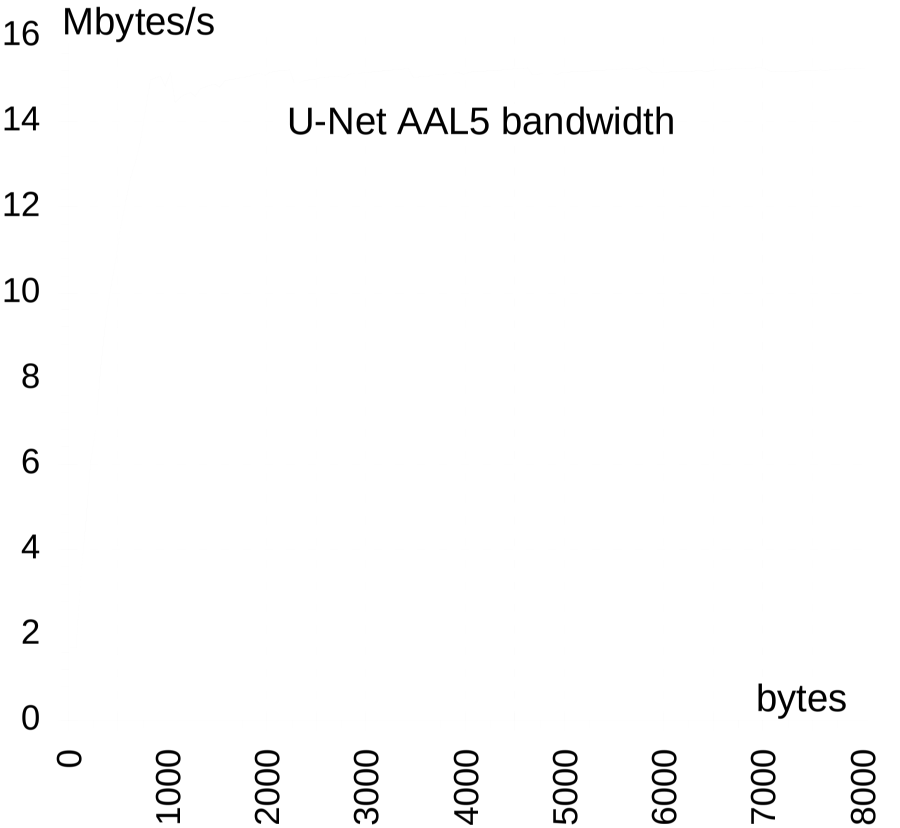


U-Net Characteristics

- Each user process communicates directly with NI
  - per-process queues and comm segment, protected from other processes
  - per process U-Net “channels”, converted to/from VCI in NI
- Connection set-up still handled via kernel
  - kernel informs NI about per-process channel<->VCI mappings
  - kernel can enforce protection/authorization/authentication
- Optimized short messages
  - single packet sned is optimized (ATM: single cell = 40 bytes payload)
  - single packet receives fit in receive queue - no buffer alloc necessary
- Supports scatter/gather
  - one PDU can consist of multiple buffers
- Various reception models
  - polling the receive queue
  - going to sleep and waking up (blocking read or select)
  - getting an interrupt (UNIX signal)

Raw U-Net Performance





- Expected low latency
- Expected high bandwidth, even with small messages

U-Net issues

How much memory per process?

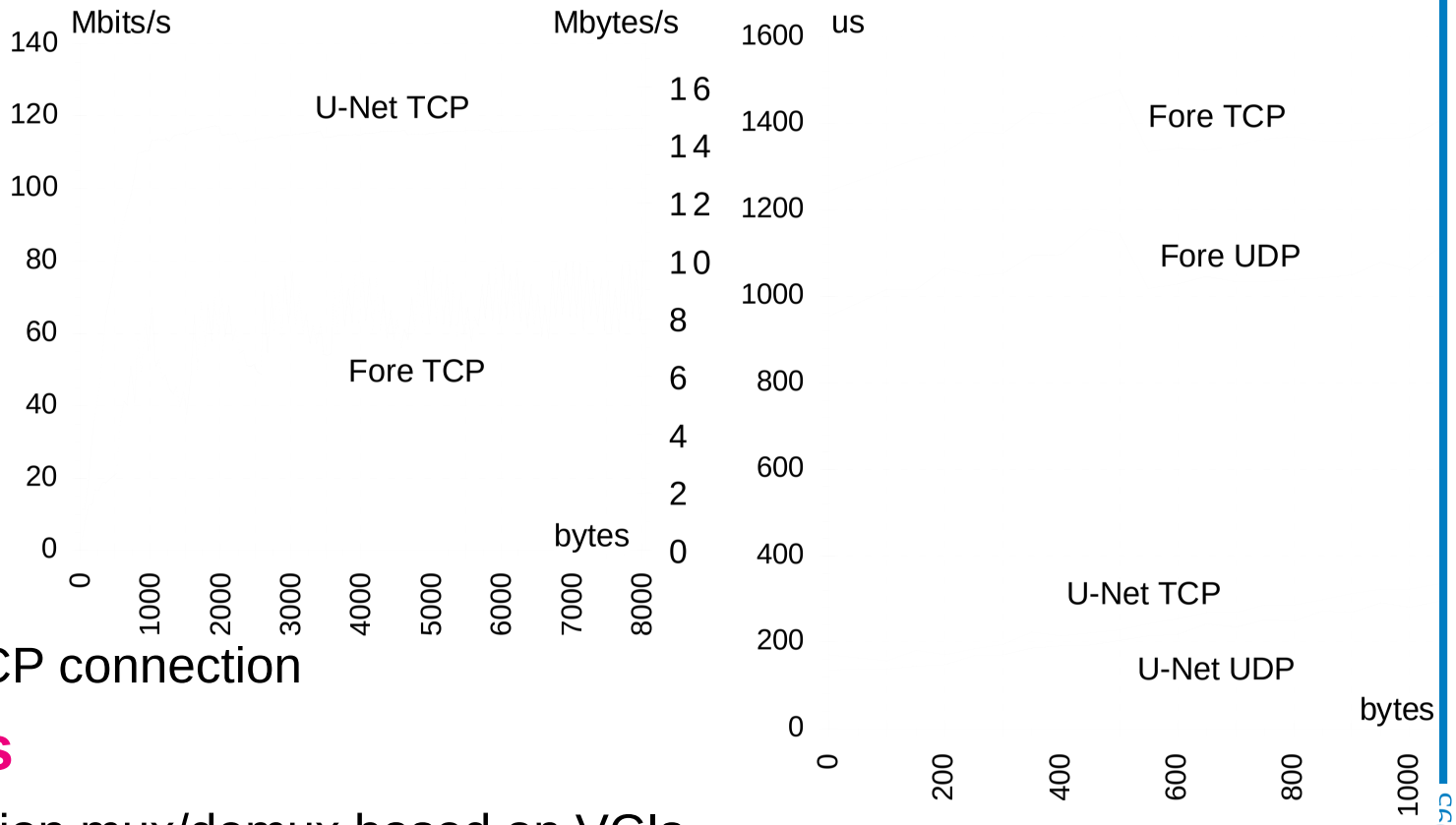
- send queue is small, receive queue should be larger
- communication segment could be huge
- all this memory is pinned
- what are the limiting factors?
  - main memory size? DMA space? Sbus address space? SBA-200 SRAM?

How about a cheap NI that doesn't "do U-Net"?

Solution: *emulated* U-Net endpoints

- for applications which don't need the high performance
- same interface, but serviced by the kernel, not by the NI
- kernel muxes all emulated endpoints over its own endpoint
  - involves system call + copy

TCP/IP over U-Net



Parameters

- full IP & TCP headers
- regular TCP checksum (in addition to AAL5 CRC!)
- one VCI per TCP connection

Improvements

- simple connection mux/demux based on VCIs
- custom buffering:
  no buffer copy, no strange fragmentation, simple allocation, pre-aligned
- straight-forward acks: no strange delays
- flow-control: can provide feed-back to application
- few buffers: 8Kb window & 2Kb PDUs

UNAM: Micro-benchmarks

Performance

- small message round trip time: 66µs (AM=10%).
- bulk xfer bandwidth: 15MB/sec @3Kbytes.
- comparisons:
  - CM5 — 12µs round-trip, 10MB/sec bandwidth
  - SP-2 — 52µs round-trip, 35MB/sec bandwidth
  - CS-2 — ~25µs round-trip, ~20MB/sec bandwidth

Issues remaining to be resolved

- improving the flow-control
- reducing the memory requirements

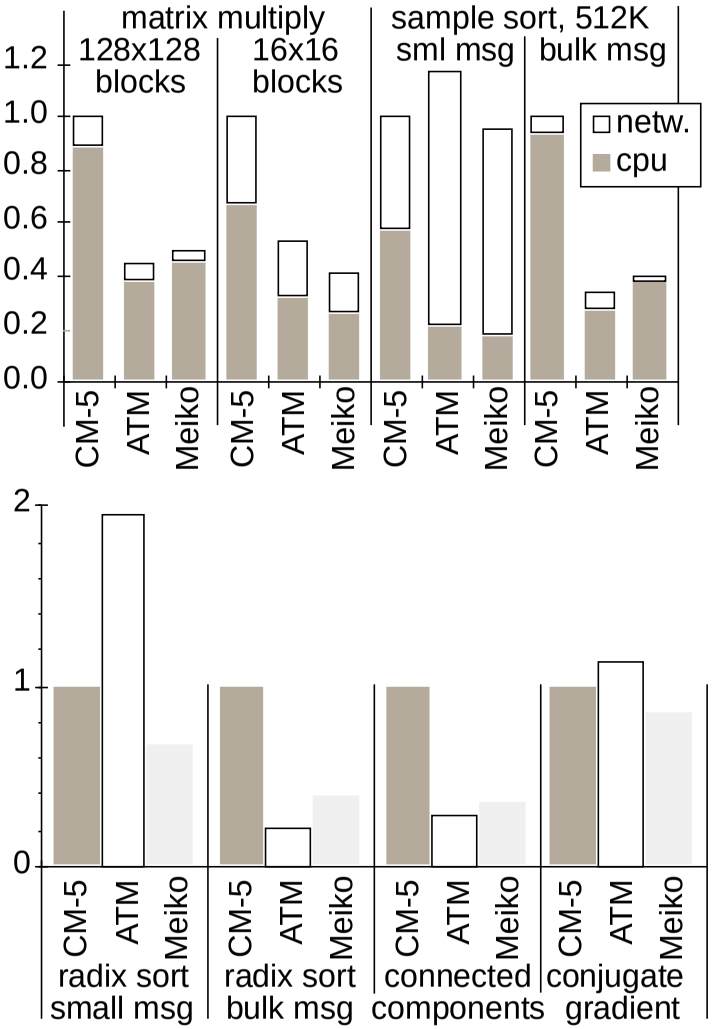
Split-C: Application benchmarks

Machines

- CM-5: 33Mhz SS-2
- CS-2: 40Mhz Supersparc
- ATM: 50/60Mhz Supersparc

Results

- on 8 processors
  normalized to the CM-5.
- compute phases:
  ATM > CS-2 > CM-5
- small msg comm phases:
  CM-5 > CS-2 ≥ ATM
- large msg comm phases:
  CS-2 > ATM > CM-5



Caveat: ATM cluster has no coordinated scheduling

Other uses of U-Net (Student Projects)

Real-time video transport

- snarf X-window data direct from 8-bit frame buffer
  xmit to remote workstation
  paste into X-window direct on frame buffer
- over 90Mbit/sec bandwidth using custom (broken)protocol
- needs more research into “real-time” communication protocols

Distributed Shared Virtual Memory

- port of Quarks DSM from UDP to U-Net replaces comm module
- U-Net works fine, but optimizations in Quarks are for slow nets
- most of the time is spent in sending page deltas instead of raw data

Remote Procedure Call

- ground-up implementation using DCE stub compiler
- avoid complex marshalling is same arch, reliable packet stream
- approx 200µs round-trip RPC

Summary

Order of magnitude increase in network bandwidth requires *system-wide* rethinking!

- Networking layers
  - In conventional systems the kernel is in the way
    - 1. kernel layers cannot be optimized for all networks (from SLIP to ATM)
    - 2. kernel layers cannot be optimized for all applications (telnet to video)
    - 3. protection boundary crossings cost
  - UDP/TCP are not a problem, but they're not cheap either
- Application layers
  - The network ceases to be the bottleneck
  - First got to undo a decade of optimizations against slow UDP/TCP ethernet
  - Then got to think hard about compute phases & overall scheduling

Summary (cont.)

U-Net offers the full performance of ATM networks

- required redesign of all network-related software
  - the hardware is not a problem (could be faster though...)
  - ATM is not a problem (could be better though...)
- Result: simple user-level network interface access
  - model is independent of ATM and independent of communication model
  - full ATM performance without dedicating all the memory and all the processor to it
  - supports hot parallel languages, as well as legacy protocols
  - tremendous protocol flexibility at the application level, enabling new modes of use

Next:

- true zero copy: communication segment = user-space
- 622Mbit/sec (?)
-