ATM and Fast Ethernet Network Interfaces for User-Level Communication

Matt Welsh
Department of Computer Science
Cornell University
mdw@cs.cornell.edu, http://www.cs.cornell.edu/Info/Projects/U-Net/

Joint work with Anindya Basu and Thorsten von Eicken

User-level Network Interfaces: Motivation

Motivation 1: Performance

- *Utilize high-speed nets*
- *Support parallel processing on NoW's*
- *Finer comm granularity*

fi That means: Low latency and high bandwidth

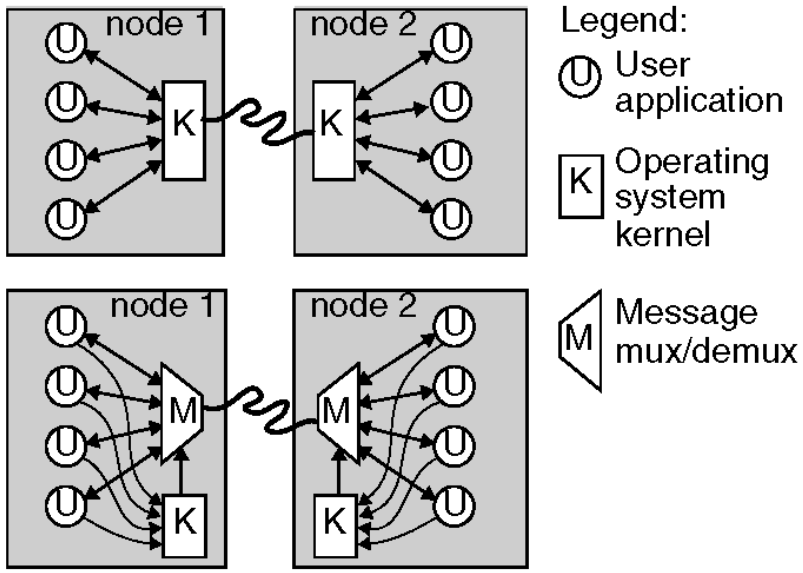
Motivation 2: Flexibility

- *Variants of standard protocols*
- *New communication semantics*
- *That means: Abandon in-kernel protocol stacks*

Proposed solution: User-level network access

- *Provide minimal interface enabling communication*
- *Application implements protocols directly*
- *Need to ensure protection between processes*

U-Net: Basic Idea



Traditional:

- All communication via kernel

U-Net:

- Applications send/recv directly via simple MUX in NI

- Kernel only involved in connection set-up/shut-down

Generic communication architecture

- Can be implemented in hardware, software, or both

Invariants:

- Off-the-shelf hardware and software

- No compromise on protection:

A cannot inspect or corrupt B's messages, A cannot impersonate B

Inspiration: MPP Systems

Key idea: User-level access to NI

- *Examples: TMC CM-5, IBM SP-2, Meiko CS-2*

Advantages

- *Bypasses the kernel for send/recv*
- *No copy: DMA direct to/from user memory*
- *Application-specific protocols*

Shortcomings

- *Custom network and NI*
- *Assumes homogeneous nodes*
- *Restricts multi-user capabilities*

Overview

Summary: Explore design space of user-level NI's

- What is the hardware/software tradeoff?

- Does user-level communication require expensive/complex NI's?

Our method:

Comparing implementations of U-Net

- NIs with and without a programmable co-processor

- Explore Fast Ethernet as an alternative to ATM for commodity interconnect

Detailed performance analysis

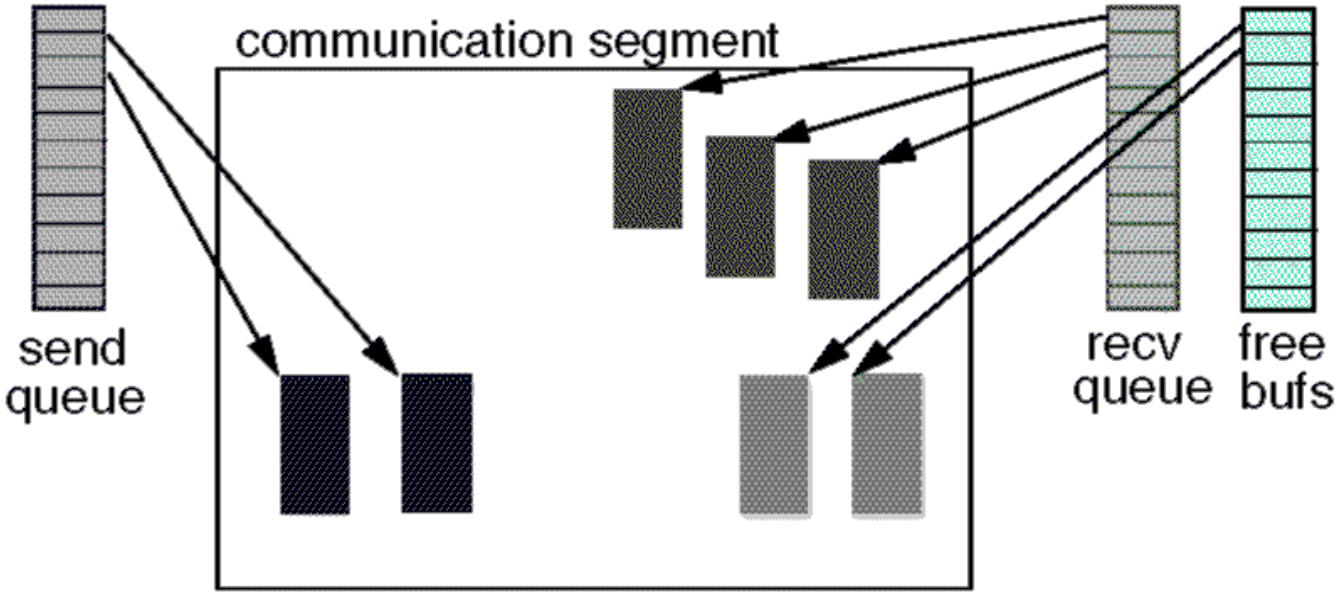
- Careful instrumentation of U-Net/FE implementation

- Micro-benchmarks for latency/bandwidth performance

Application performance

- Set of parallel benchmarks measured over FE and ATM workstation clusters

The U-Net Interface



U-Net Endpoint: Virtual device interface

- Message buffers and send/recv/free queues

Transmit operation:

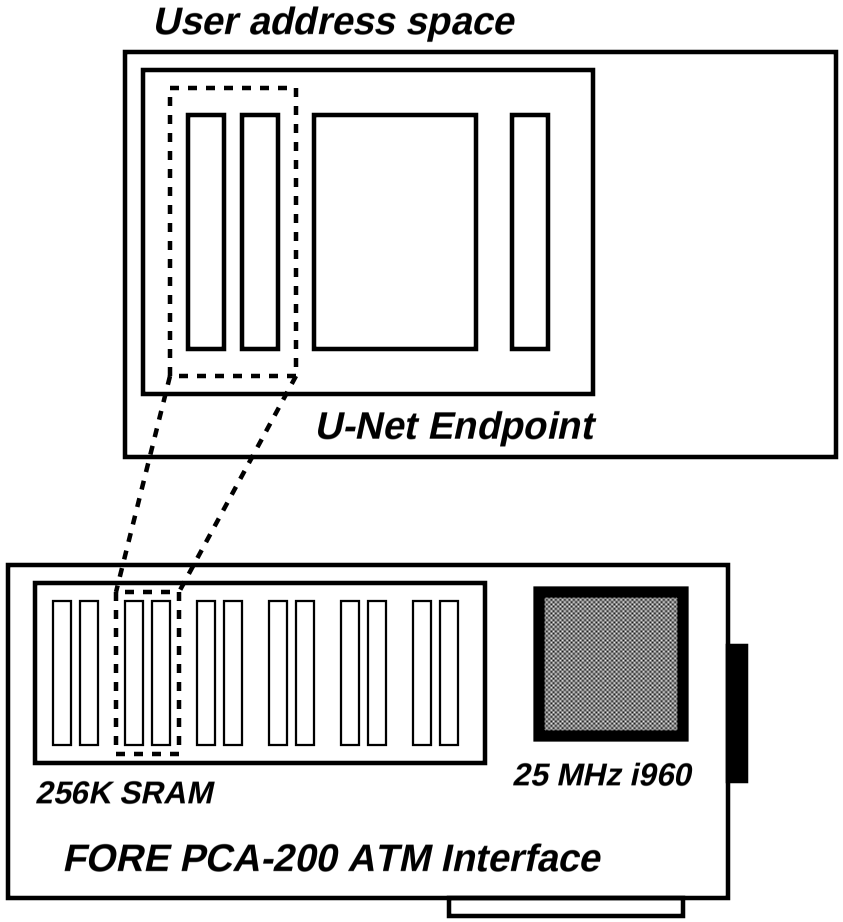
- User constructs msg in buffer area, pushes Tx descriptor onto send queue

Receive operation:

- Msg arrives, data in buffer from free queue, Rx descriptor pushed onto recv queue

U-Net ATM Implementation

- *Original implementation of U-Net*
- *Programmable co-processor, ATM as "obvious choice" for interconnect*



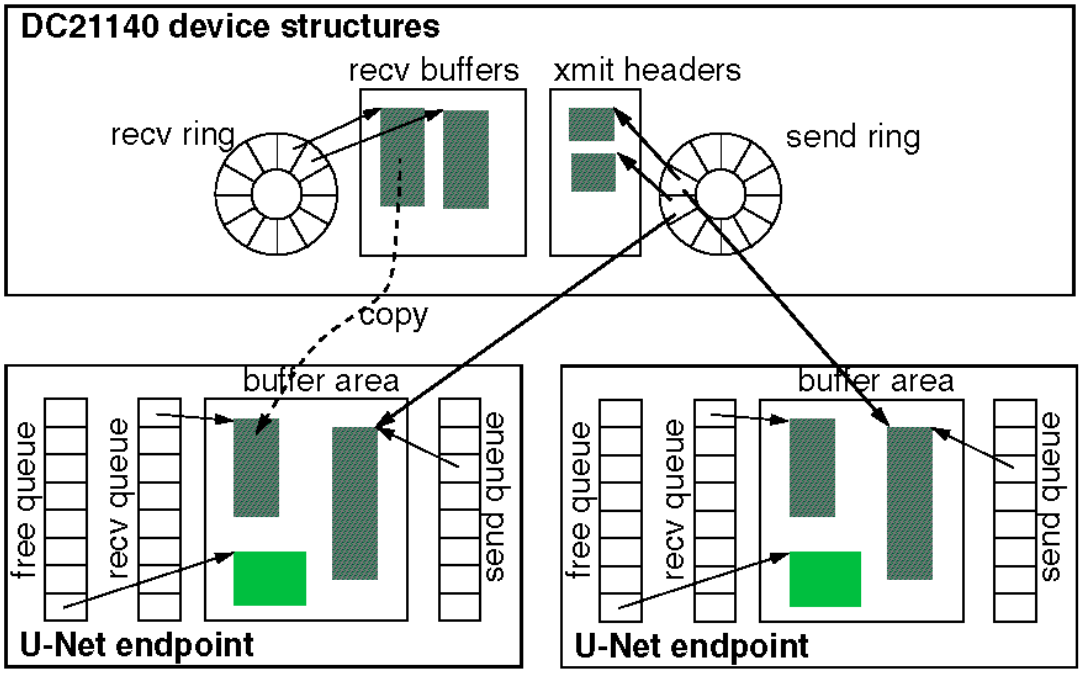
FORE Systems PCA-200

- *PCI bus 155 Mbps OC-3 ATM NI*
- *25 MHz i960, 256K SRAM*
- *Pentium 133 WS, Linux 1.3.97*

- *U-Net implemented on i960*
- *Tx/Free rings mapped from i960 RAM*
- *Buffers, Rx ring in pinned memory segments*

  ... always DMA-able by the i960
- *No O/S, CPU intervention in Tx/Rx*

U-Net Fast Ethernet Implementation



DECchip 21140 FE controller

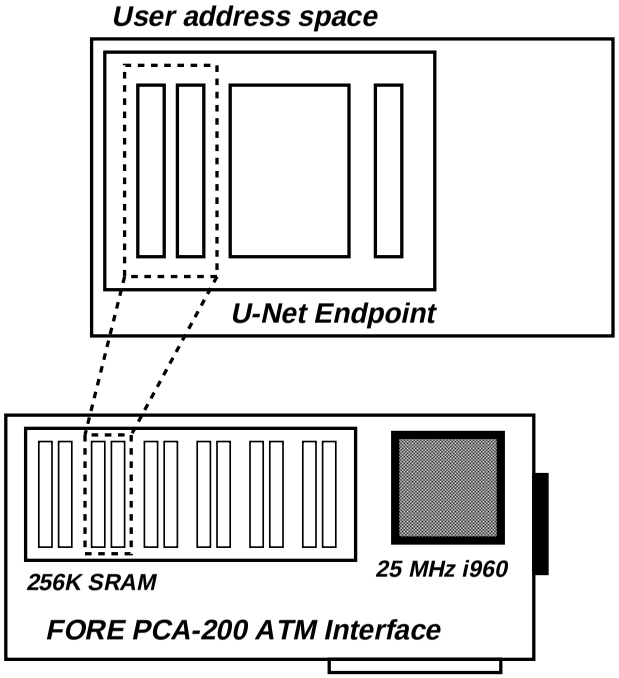
- *100 Mbps UTP5 or fiber*
- *PCI busmastering interface*
- *But, not programmable*
- *Low cost: $150/board*
- *Pentium 133 WS, Linux 1.3.97*

- *Single, shared Tx and Rx rings, buffer pool*
- *Assumes single O/S agent to mux the queues*
- *U-Net implemented in kernel trap and interrupt routines*

Transmit Operation

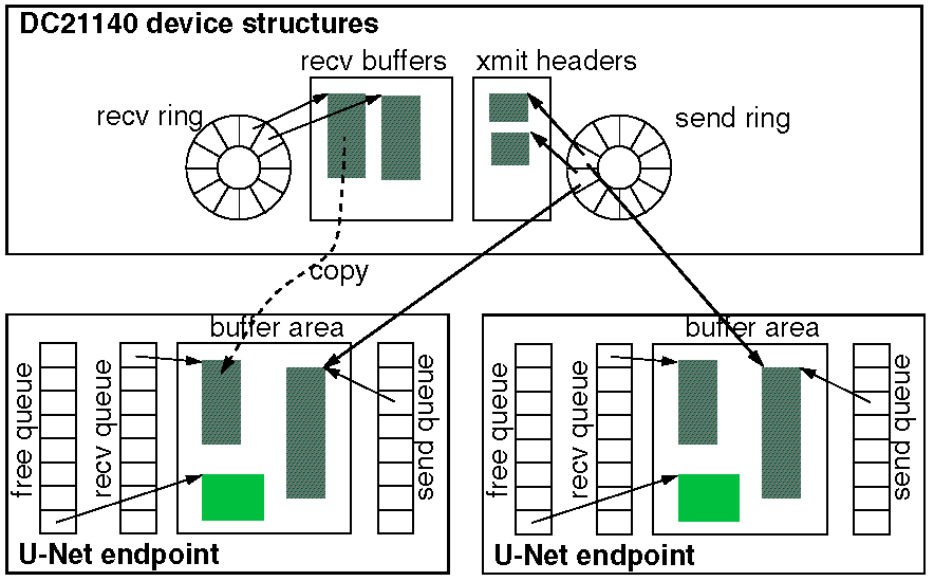
U-Net/ATM

1. User constructs data in buffer region
2. User pushes Tx descr into Tx Ring
3. i960 polls Tx rings, fetches descriptor
4. i960 initiates DMA to fiber output
5. i960 sets Tx descr done flag



U-Net/Fast Ethernet

1. User constructs data
2. User pushes Tx descr
3. User calls trap
4. Trap pushes descr to device Tx Ring
5. On Tx done, trap sets Tx descr done flag



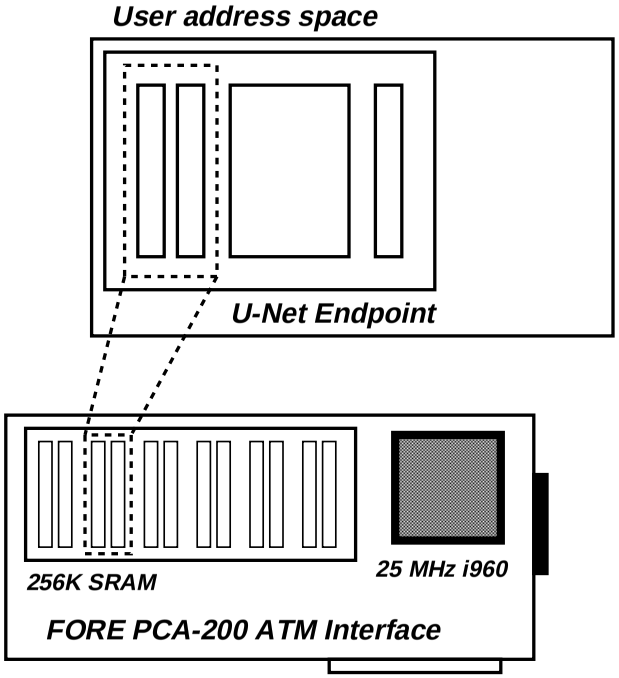
Receive Operation

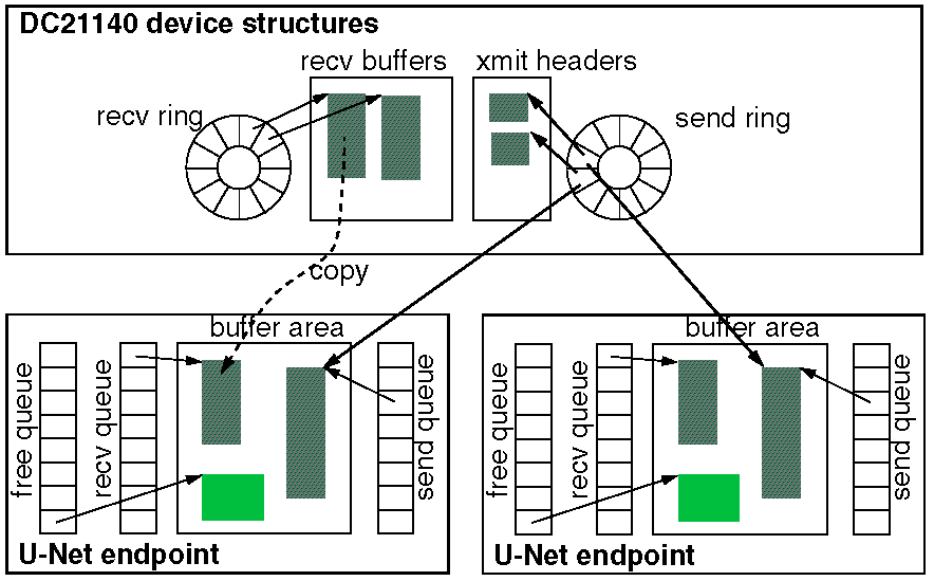
U-Net/ATM

1. *AAL5 PDU cells arrive at fiber input*
2. *i960 fetches free buffer descr*
3. *i960 initiates DMA to free buffer*
4. *At End-of-PDU, i960 writes Rx descr*
5. *User polls Rx FIFO, or upcall*

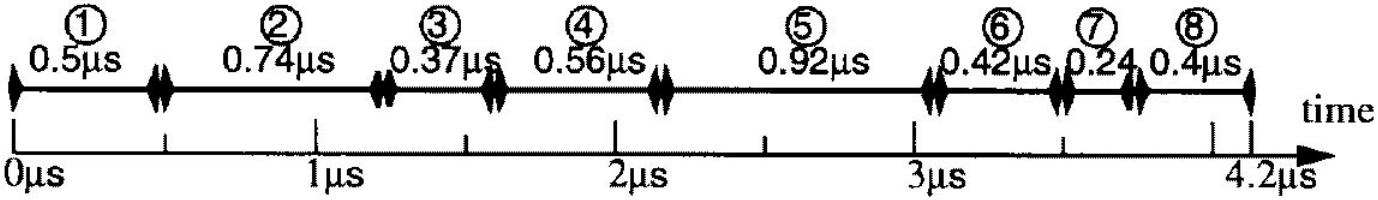
U-Net/Fast Ethernet

1. *FE packet arrives, interrupt raised*
2. *Intr fetches free buffer descr*
3. *Intr copies from device buffer to user buffer*
4. *Intr writes Rx descr into Rx FIFO*
5. *User polls Rx FIFO, or upcall*





U-Net/FE Transmit operation



1. trap entry overhead
2. U-Net send param check
3. Ethernet header set-up
4. device send ring descr set-up
5. issue poll demand to DC21140
6. free send ring descr of prev message
7. free U-Net send queue entry of prev message
8. return from trap

Figure 3: Fast Ethernet transmission time-line for a 40 byte message (66 bytes with the Ethernet and U-Net header).

Fast trap to start transmit, 4.2 usec any size packet

- *Null trap: 1 usec*
- *PCI access time dominates*
- *Trap semantics are 'service U-Net Tx queue'*
- *Trap seen as 'protected co-routine'*
- *Take (small) slice of main CPU time to mux U-Net*

U-Net/FE Receive operation

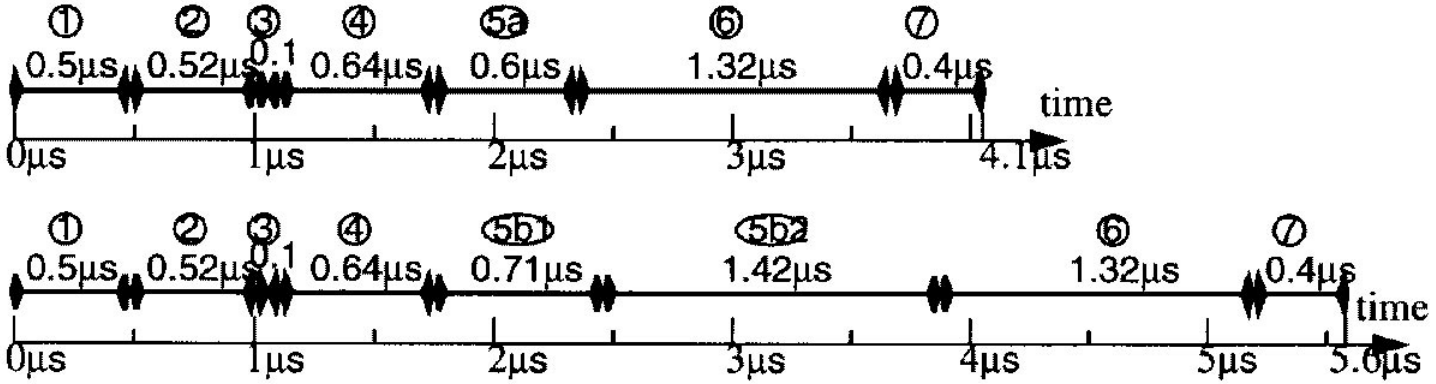


Figure 5: Fast Ethernet reception time-line for a 40-byte and a 100-byte message.

Interrupt handler on Rx, copy time dominates

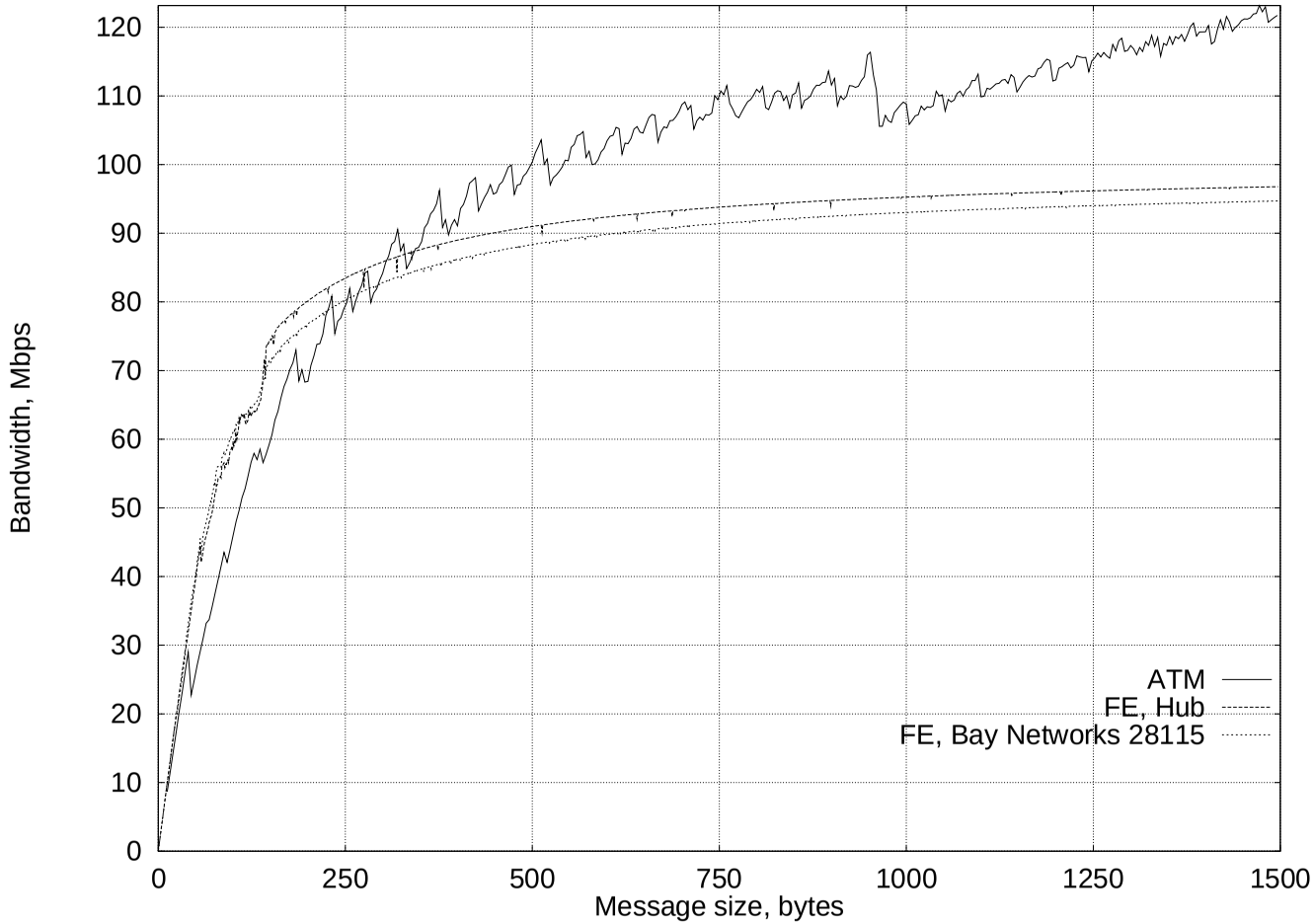
- Msg arrives in fixed buffer pool in kernel, copy to user

Mux/Demux:

- U-Net 'protocol ID' in Ethernet header, plus 'channel number' and length field

- Need to integrate with IP/packet filtering

Performance: Bandwidth



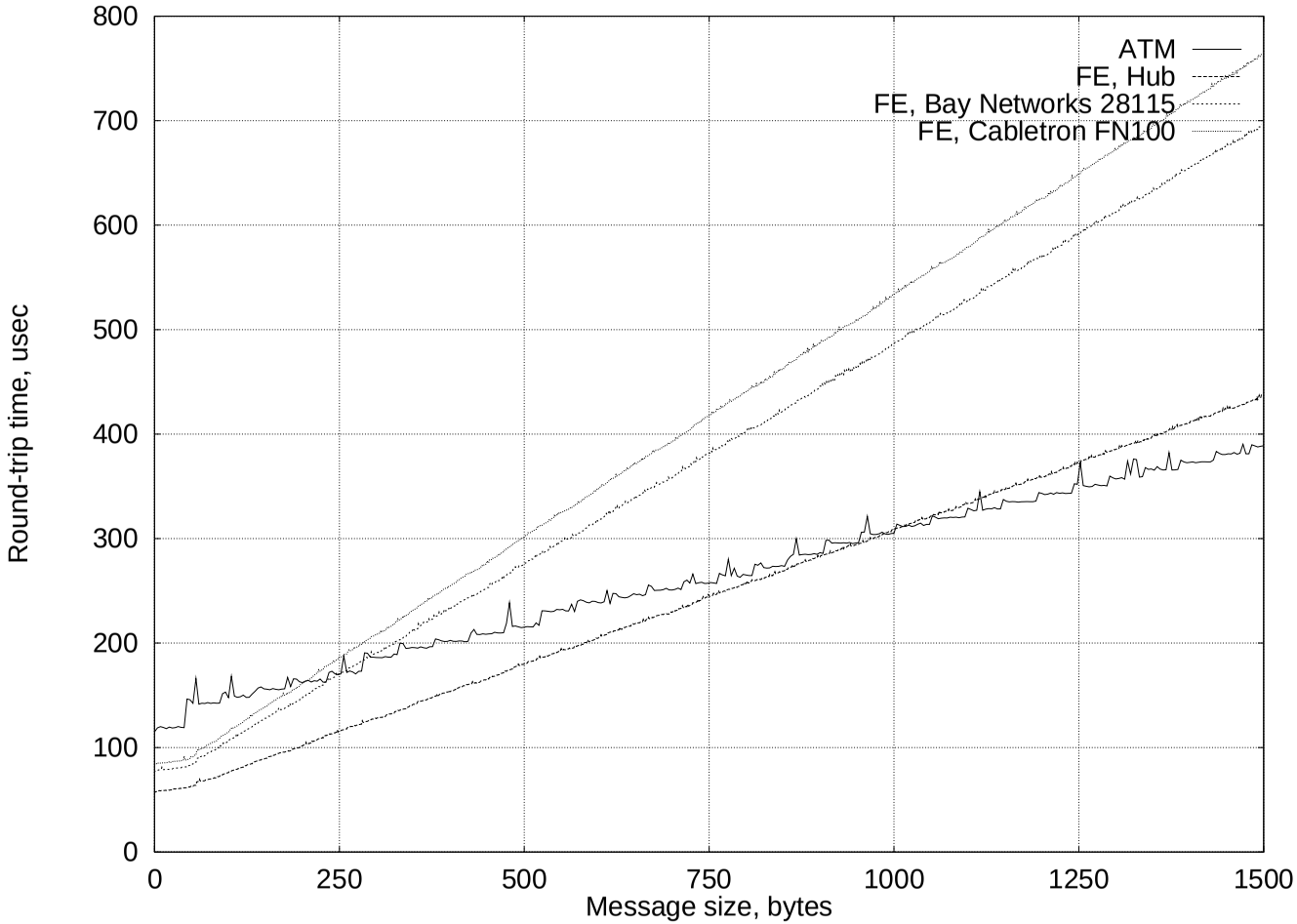
ATM

120 Mbps TAXI used as receiver

Fast Ethernet

90 Mbps+ with 500-byte messages ... but switch shaves off some b/w?

Performance: Latency

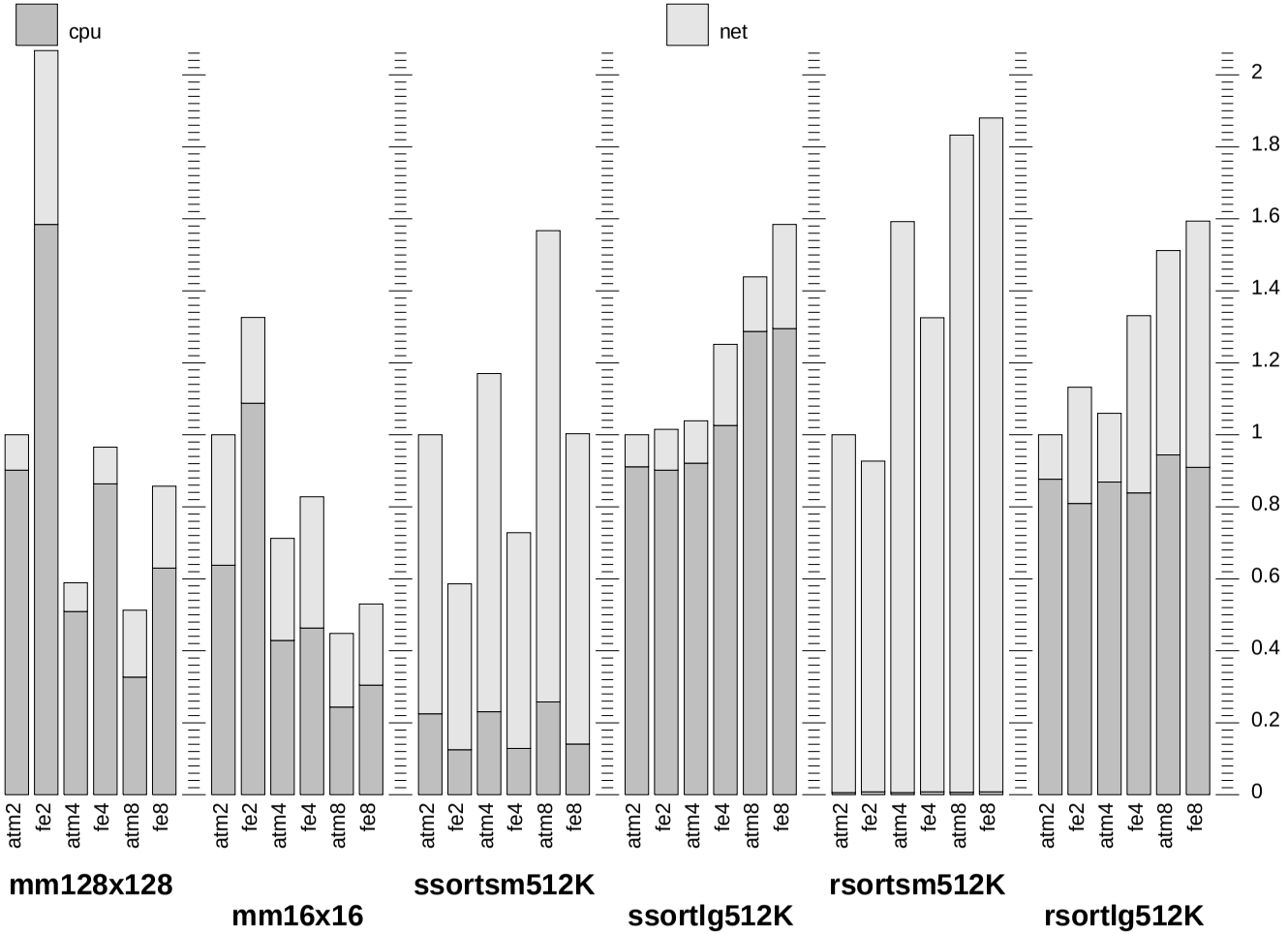


FE has lower latency than ATM!

... for small messages, anyway

- *FE switches add 17 usec one-way*

Split-C Benchmarks: ATM vs. FE



mm: Matrix Multiply

ssort: Sample sort

rsort: Radix sort

SparcStations used in ATM cluster

- *ATM faster for large msgs*
- *FE faster for small msgs*
- *SPARC fp faster than Pentium*
- *Pentium int ops faster than SPARC*

Split-C

- *Novel parallel language based on C*
- *Use of 'global pointers' to access other proc addr space*

Current work work: Memory Management

Pinned buffers and queues

- Locked into physical memory for lifetime of process

- Required to allow direct DMA to/from user space

Paging Endpoints

- On-demand paging of U-Net buffers

- Uses software TLB to cache page mappings

- TLB miss causes kernel interrupt to fetch page

- Pages discarded on TLB capacity miss

Issues

- Writeable pages are easy to get

- What about swapped-out read page? (Can't swap in interrupt...)

- Lazy read-page retrieval: Tell the NI to try again

Implementations for PCA-200 & Linux, DC21140, Zeitnet & Windows NT

Summary

U-Net extended to non-programmable NICs

- Implementation using DC21140 FE interface

- Hardware requires kernel trap and copy on receive

U-Net extended to Fast Ethernet

- Round-trip latency starts at 57 usec, 40 byte ping-pong

- Lower latency than OC-3 ATM (120 usec, 40 byte ping-pong)

- Bandwidth reaches > 90 Mbps with 500-byte messages

Conclusions

- U-Net model extends to other networks and NI architectures

- Split-C benchmarks demonstrate comparable app performance

- Fast Ethernet is an excellent price-performance point for workstation clusters