Unconventional Networking

Makoto Bentz

October 13, 2010

CS 6410

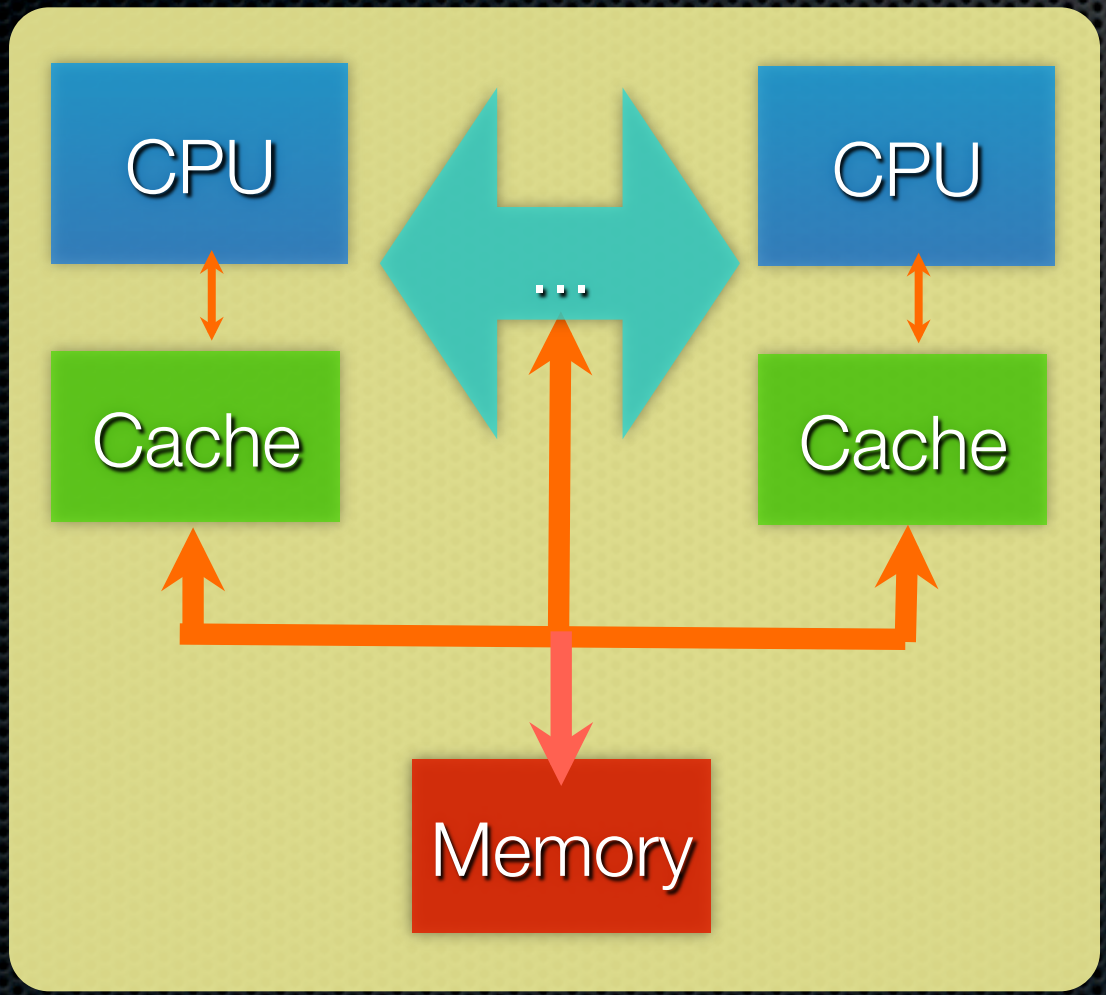
Papers

- U-Net: A User-Level Network Interface for Parallel and Distributed Computing, Von Eicken, Basu, Buch and Werner Vogels. 15th SOSP, December 1995.
- 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.
- Evaluation of the Virtual Interface Architecture (VIA). Xin Liu June 8, 1999 Department of Computer Science and Engineering University of California, San Diego

Overview

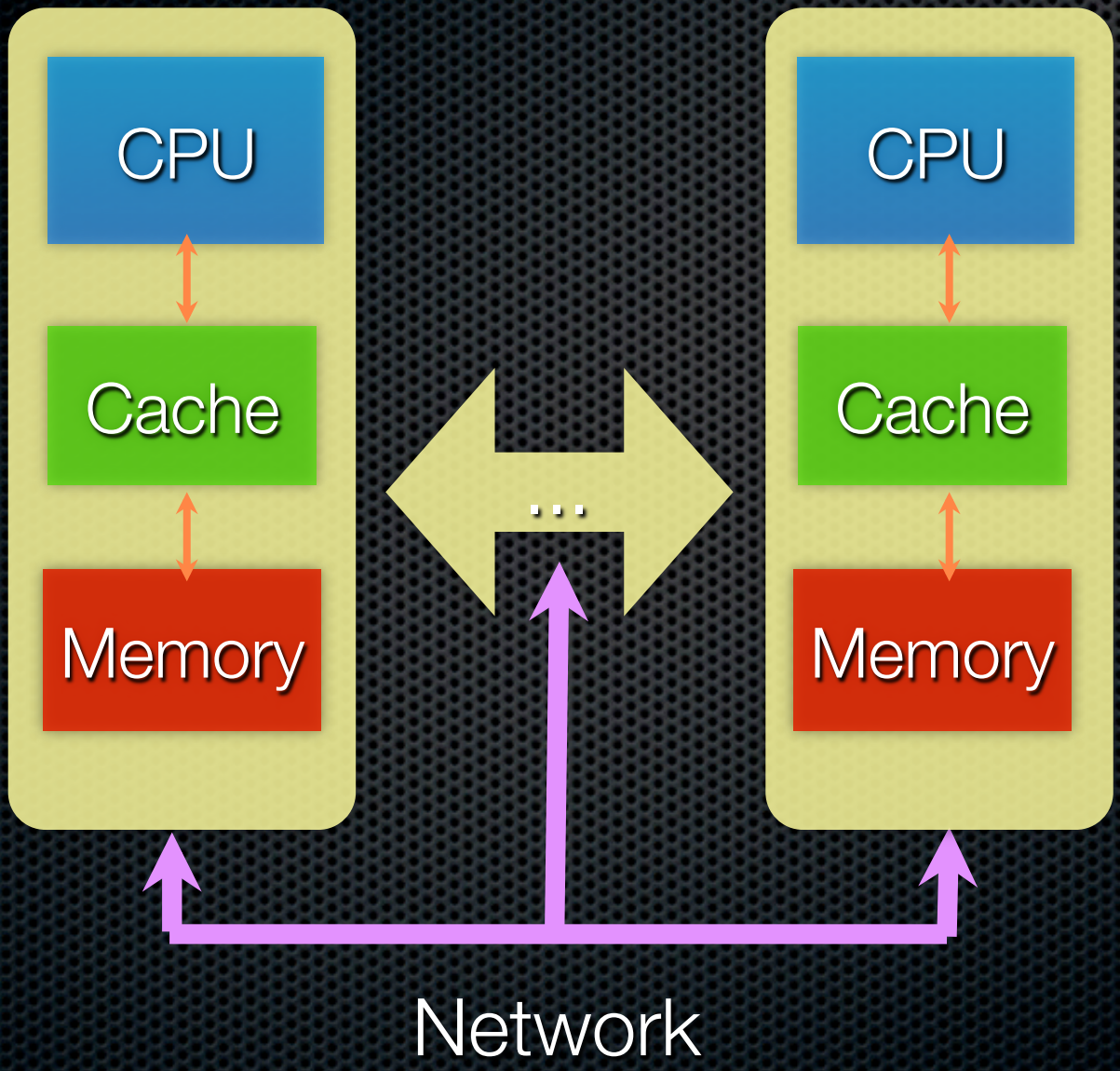
- **Active Messages:** Instructions between multiprocessors
- **U-Net:** Networking beyond the kernel, separation into the User Space
- **VIA:** Fast Messaging on virtualized standard network interfaces (VIA)

Parallel versus Distributed



- **Parallel Computing**
- Multiple processors connected in a computer
- Tightly coupled
- Shared memory

Parallel versus Distributed

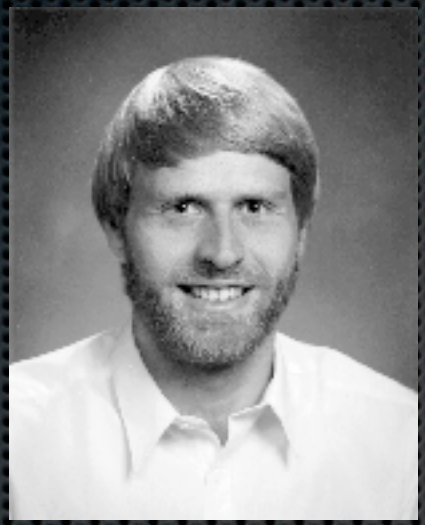


- **Distributed Computing**
- Multiple autonomous computers connected by network
- Loosely coupled
- Distributed memory

Active Messages

- Written at by von Eicken, Cohen and at University of California Berkeley, in David E. Culler's research group
- Other projects by group include Split-C, Castle and Ninja
  - Ninja

Active Messages: Authors



- Seth Copen Cohen is now a professor at CMU
  - Programmable matter

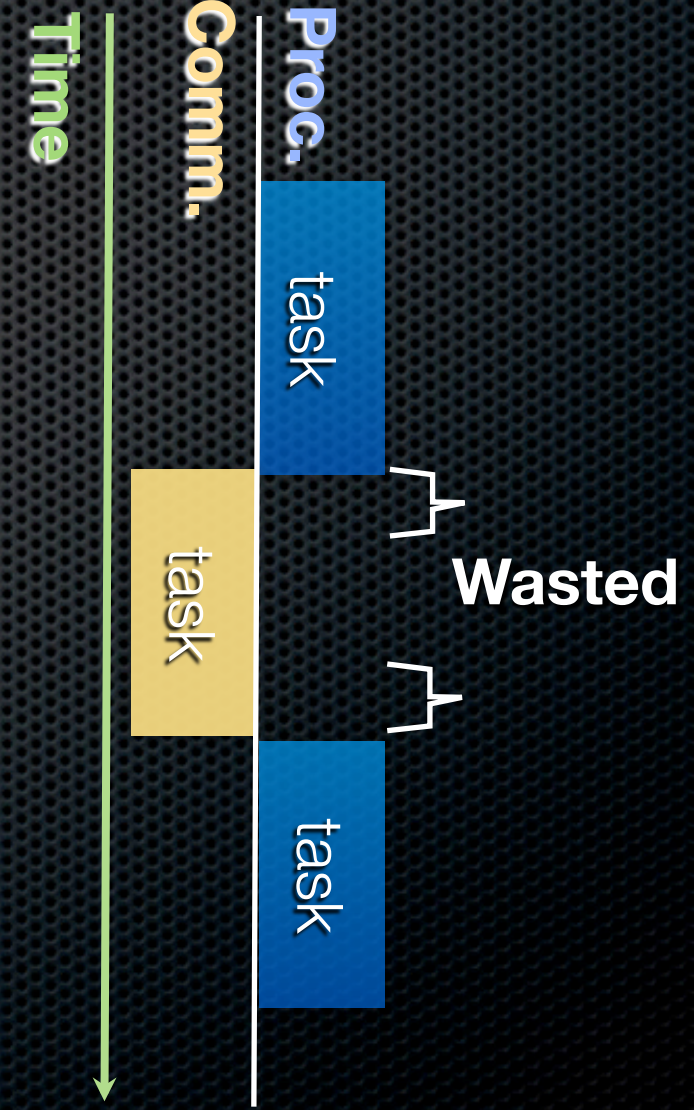
- Klaus Erik Schauser is now a professor at UCSB
  - Parallel computing

Thorston von Eicken

- Advised by David E. Culler (MIT, teaches at U of C Berkeley)
- Former Cornell Assistant Professor
- Active Messages becomes his Ph.D. thesis
- Now teaches at UCSB
- Worked at ExpertCity.com, worked on GoToMyPC and GoToMeeting
- Now works at RightScale, a cloud computing company

Active Messages: Motivation

- Message passing machines have reasonable hardware costs but suffer from
  - Poor overlap between communication and computation
  - High overhead for messages
- And this is not due to hardware



Active Messages:Hardware

- nCube-2
  - 1.9 Gflops/s
  - Hypercube layout
- Thinking Machines Connection Machines (CM)-5
  - Non-standard von Neumann architecture
  - Fat-tree network MIMD

NSA FROSTBURG a CM-5 machine

Active Messages:Design

- Get around the comm/proc delay
- Active messages are messaging objects that can perform its own processing
  - Each message carries a userspace handler or code
  - The message contains the arguments to pass to the handler
  - Single Program, Multiple Data

Active Messages:Design

- Program pre-allocates receiving structures, eliminating buffering
- The network itself acts as a buffer for small messages
- Creates low overhead on both sides
- Message handlers are not allowed to block

Active Messaging:Testing

- nCube/2
  - 11us send + 15us receive (nearly an order of magnitude reduction)
    - Almost near minimal instructions
- CM-5
  - 1.6us send + 1.7us receive

Active Messaging:Split-C

- Spit-C PUT and GET perform split-phase copies of memory blocks between nodes
- For high performance, should overlap communication and computation

```
int N, R, M;                /* matrix dimensions (see figure) */
double A[R/P][N], B[M/P][R], C[M/P][N]; /* matrices */
int i, j, k;                /* indices */
int j0, dj, nj;             /* initial j, delta j (j=j0+dj), next j */
int P, p;                   /* number of processors, my processor */
int Rp = R/P;
double V0[N], V1[N];        /* buffers for getting remote columns */
double *V=V0, *nV=V1, *tV; /* current column, next column, temp column */
static int flag = 0;        /* synchronization flag */
extern void get(int proc, void *src, int size, void *dst, int &flag);

j0 = p * Rp;                            /* starting column */
get(p, &A[0][0], N*sizeof(double), nV, &flag); /* get first column of A  */
for(dj=0; dj<R; dj++) {                 /* loop over all columns of A */
    j = (j0+dj)%R; nj = (j0+dj+1)%R;    /* this&next column index */
    while(!check(1, &flag)) ;           /* wait for previous get */
    tV=V; V=nV; nV=tV;                  /* swap current&next column */
    if(nj != j0)                        /* if not done, get next column */
        get(nj/Rp, &A[nj%Rp][0], N*sizeof(double), nV, &flag);
    for(k=0; k<M/P; k++)                /* accum. V into every col. with scale */
        for(i=0; i<N; ++)               /* unroll 4x (not shown) */
            C[i][k] = C[i][k] + V[i]*B[j][k];
}
```

Figure 4: Matrix multiply example in Split-C.

Active Messaging:Split-C

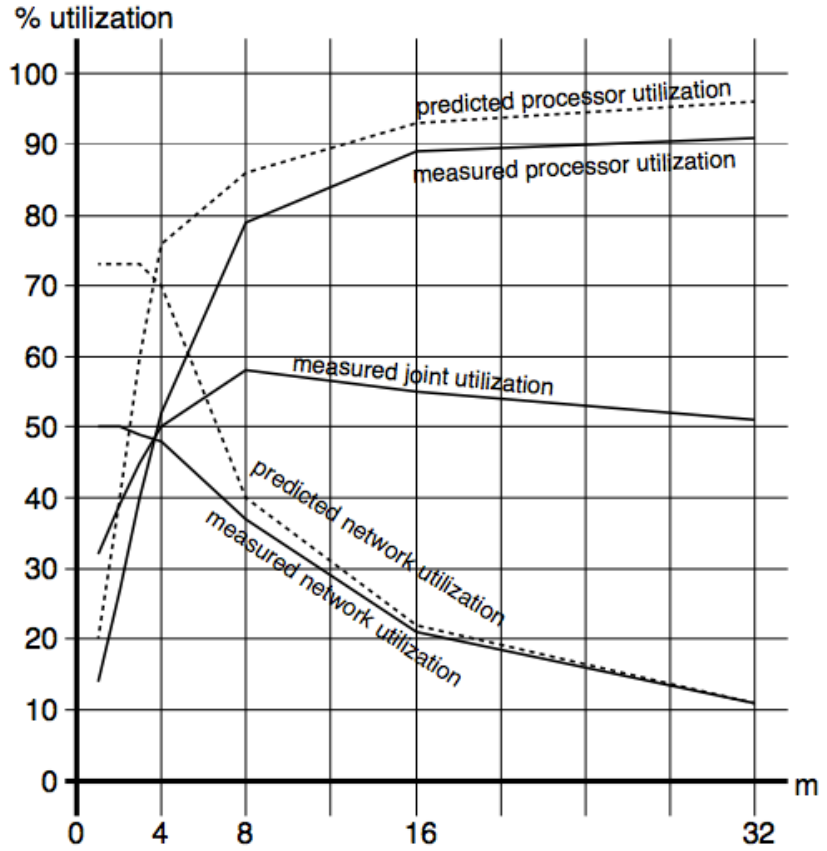


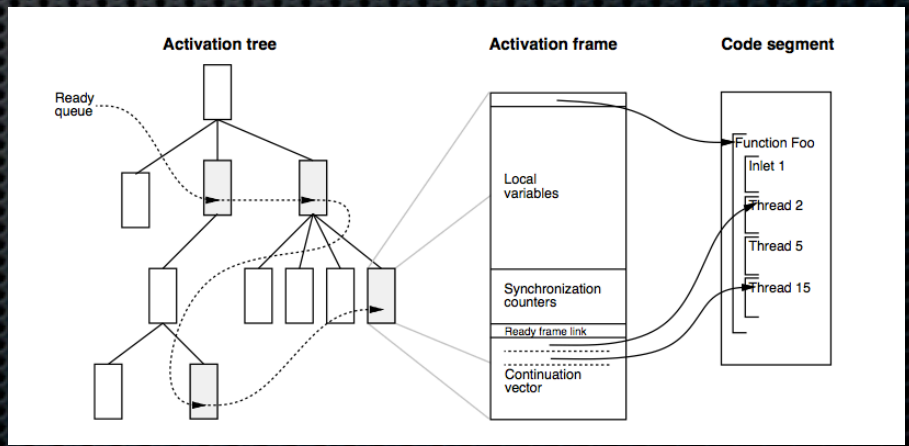
Figure 5: *Performance of Split-C matrix multiply on 128 processors compared to predicted performance using the model shown in Figure 6.*

Active Messaging:Message Driven Architectures

- Message Driven Architectures Support languages with support for dynamic parallelism
- Active Message handlers execute immediately upon message arrival, cannot suspend or block

Active Messaging:Further Work: TAM

- Threaded Abstract Machine is a fine-grained parallel execution model based on Active Message
- No testing results



Active Messaging: Discussion

- This paper becomes von Eicken's thesis
- I personally disliked the style of this paper, but liked the content
  - Too much in one article, works much better as a thesis

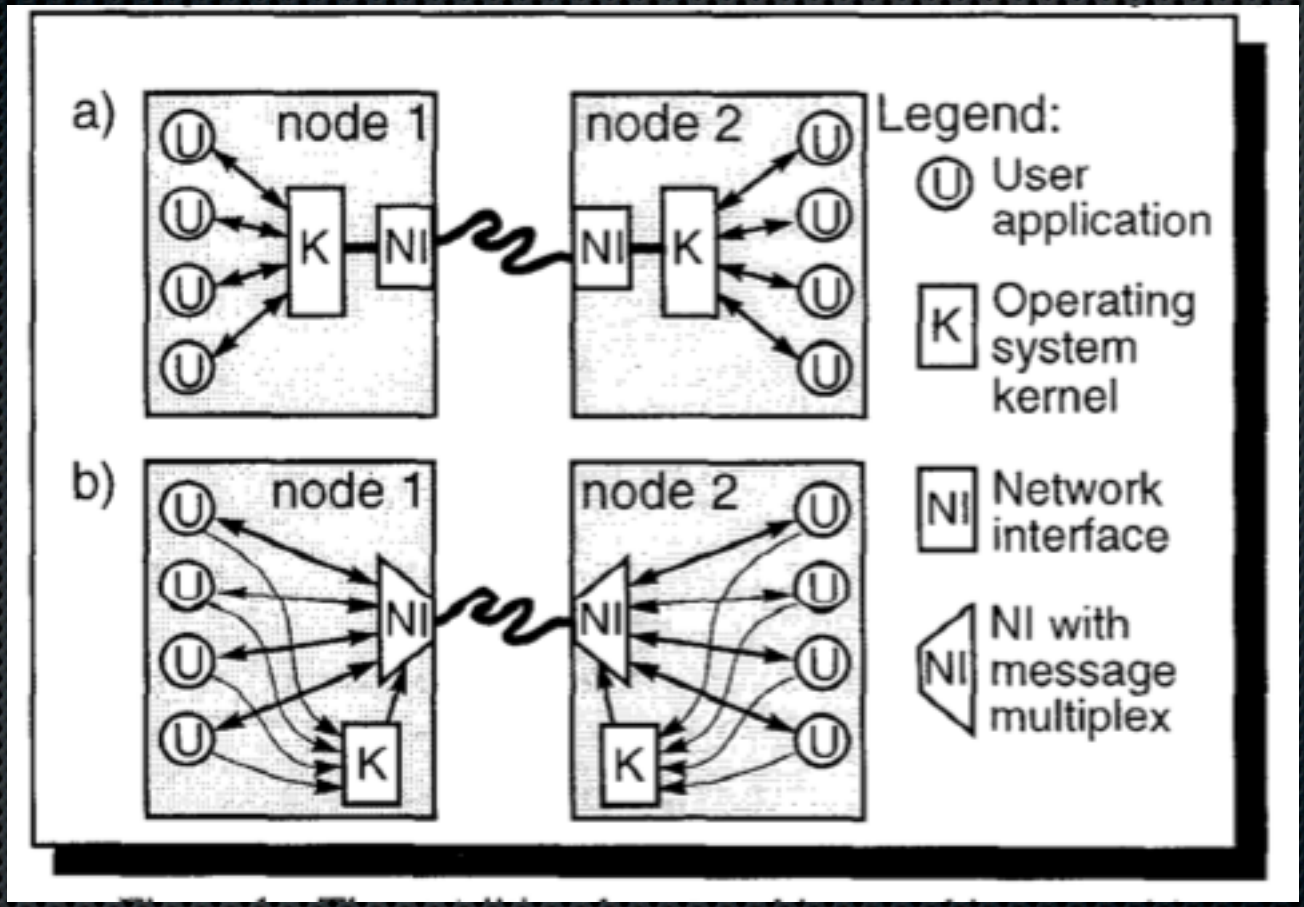
U-Net: Authors

- Assistant Professor von Eicken
- Anindya Basu (advised by von Eicken)
- Vineet Buch (M.S. from Cornell)
  - co-founds like.com (now part of Google)
  - worked with BlueRun (VC)
- Werner Vogels
  - former Researcher at Cornell
  - now Amazon CTO

U-Net: Motivation

- Networks are now so fast the biggest delay is kernel processing of the message stacks
- Developing in kernel space limits development of new message send/receive interfaces

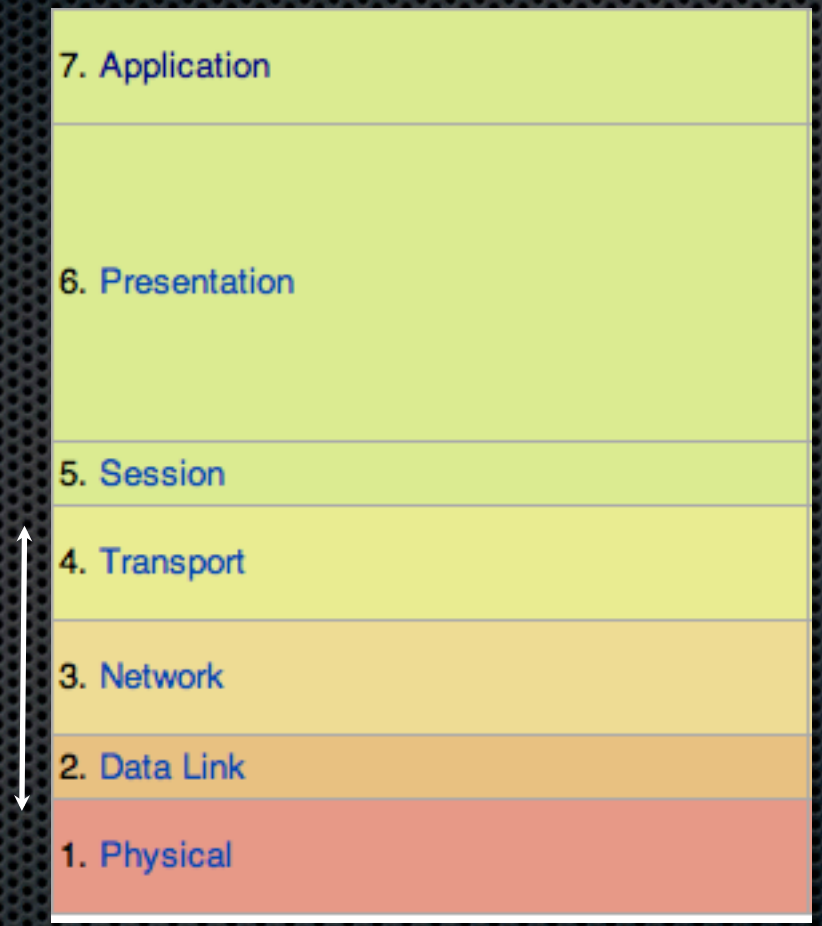
U-Net: Motivation



Removing the kernel from the critical path would allow specialization

U-Net: Motivation

Previously, Kernel's Responsibility, and opaque to the application



Want to make this application specific

Would like to be able to integrate application specific behavior to optimize performance

U-Net: Design

- U-Net must be able to
  - Multiplex the network between processes
  - Provide isolation and protection between processes
  - Manage resources without kernel paths
  - Allow the development of an interface

U-Net: Design Concerns

- Focus on low overheads to optimize small message transmission latency and bandwidth
- Emphasis on protocol design and integration
- To do both on standard off-the-shelf hardware

U-Net: Design: Low Overhead

- U-Net components map to
  - Real hardware in the NI
    - -and/or-
  - Memory locations interpreted by the OS
- Zero-Copy tries to limit intermediate buffering between NI and User-Level

U-Net: Design: Protocols

- Tags specific to the network substrate used for addressing
  - ATM (VCI)
  - Ethernet (Physical addressing)
- Endpoints, communication channels and queues are only accessible by owners

U-Net: Design: Hardware

- Implemented on SPARCstation
  - Fore SBA-100 interfaces
  - Fore SBA-200 interfaces
- Could use on-board processor on SBA-200, custom firmware due to poor bundled firmware

U-Net: Testing Results

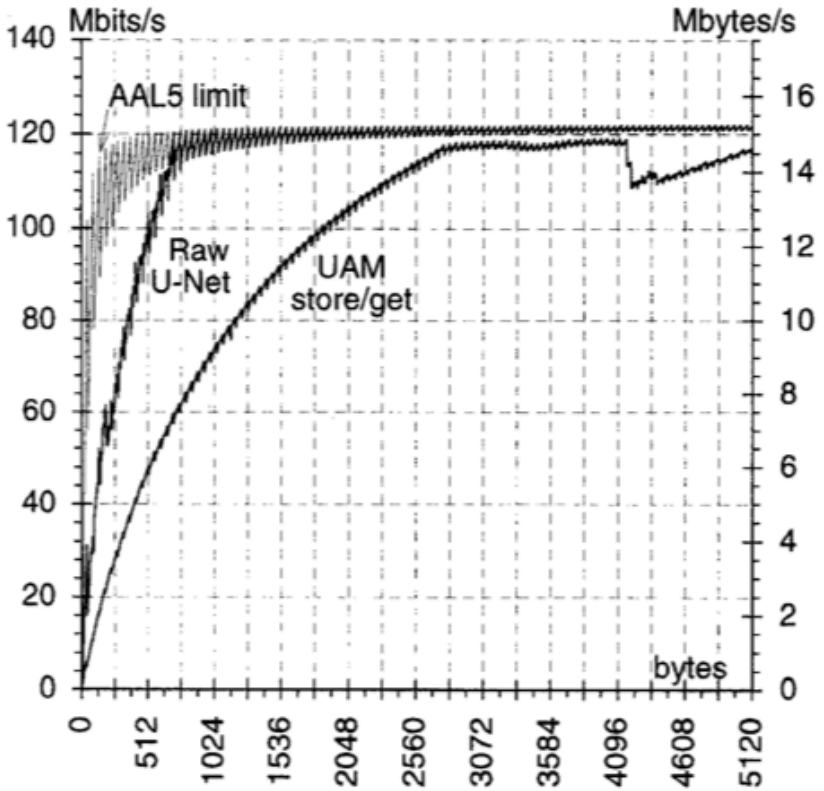


Figure 4: U-Net bandwidth as a function of message size. The *AAL-5 limit* curve represents the theoretical peak bandwidth of the fiber (the sawtooths are caused by the quantization into 48-byte cells). The *Raw U-Net* measurement shows the bandwidth achievable using the U-Net interface directly, while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

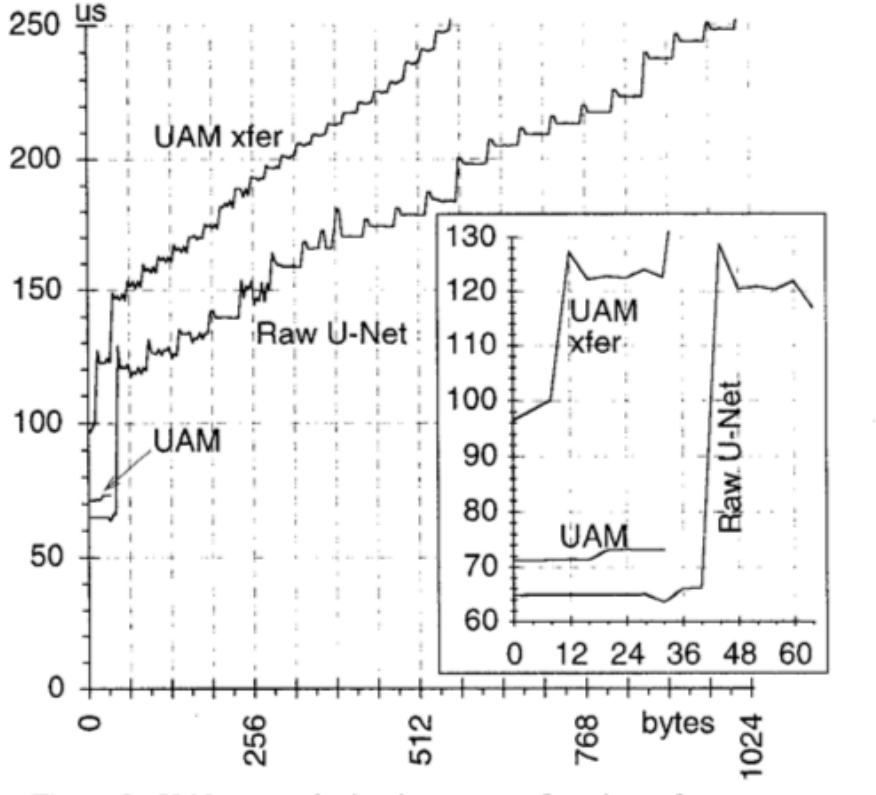


Figure 3: U-Net round-trip times as a function of message size. The *Raw U-Net* graph shows the round-trip times for a simple ping-pong benchmark using the U-Net interface directly. The inset graph highlights the performance on small messages. The *UAM* line measures the performance of U-Net Active Messages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

U-Net: Testing Results

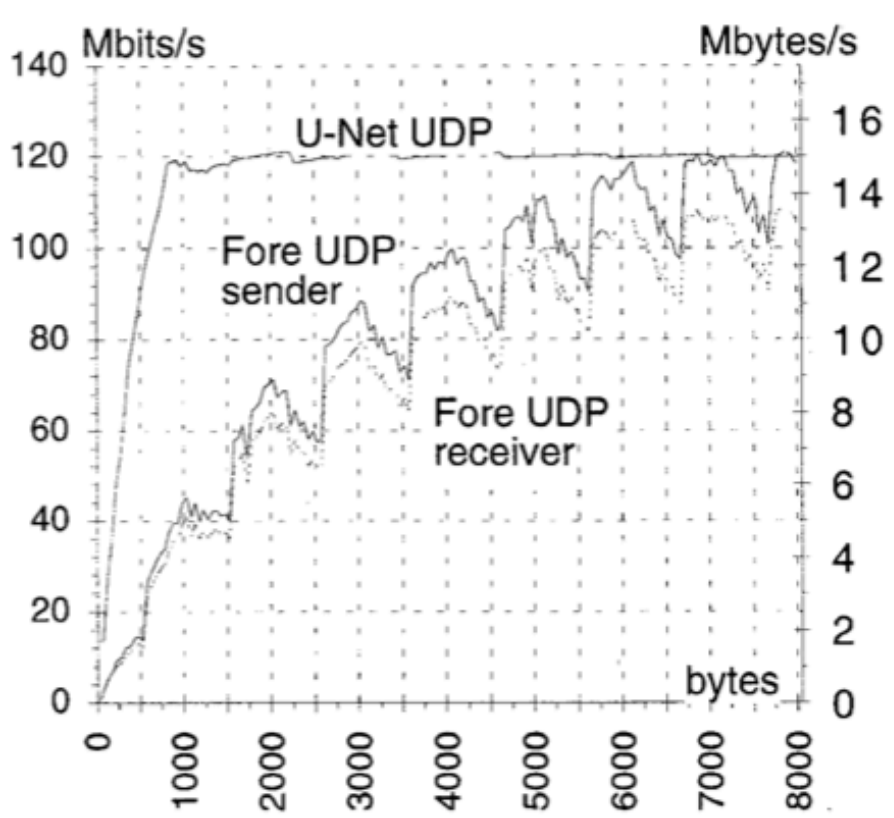


Figure 7: UDP bandwidth as a function of message size.

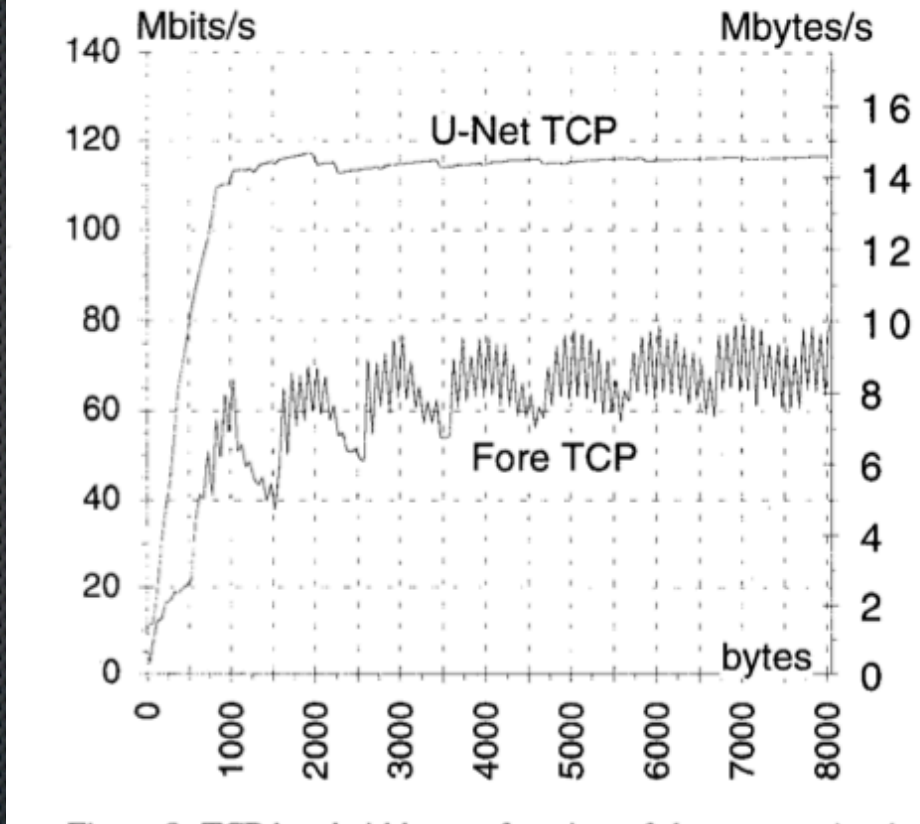


Figure 8: TCP bandwidth as a function of data generation by the application.

U-Net: Discussion

- Feasible technology
- Good performance on off the shelf hardware with existing standards
- “This encouraging result should, however, not obscure the fact that **significant additional system resources**, such as parallel process schedulers and parallel file systems, **still need to be developed** before the cluster of workstations can be viewed as a unified resource.”

FM/VIA

- Virtual Interface Architecture standardizes System Area Networks (Microsoft, Intel, Compaq)
- VIA however, has no simple programming interface
- Fast Messages provided layering for small messages

FM/VIA :: Testing Results

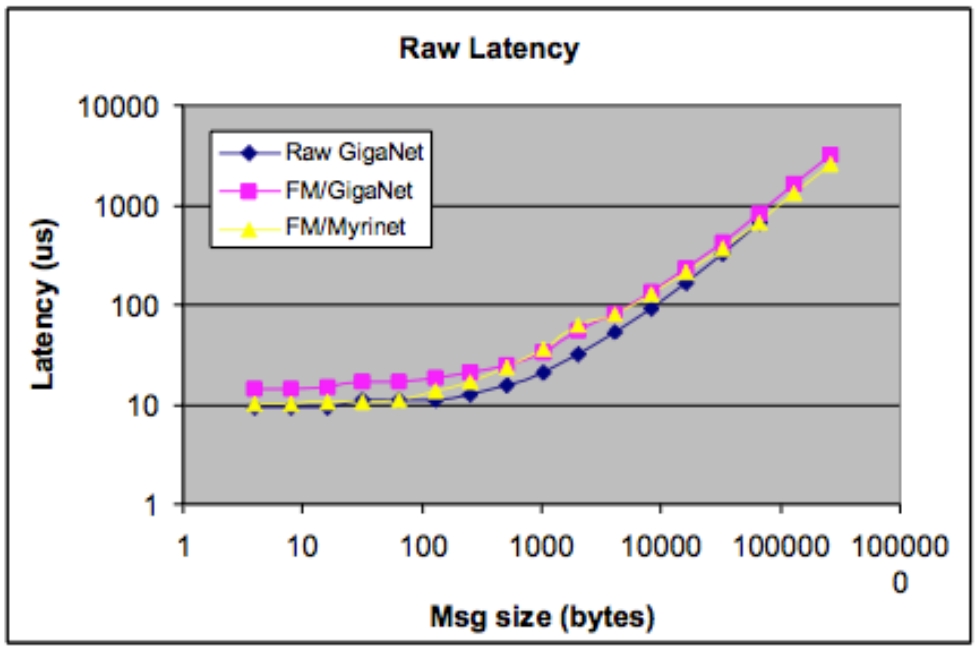


Figure 2 Raw Latency

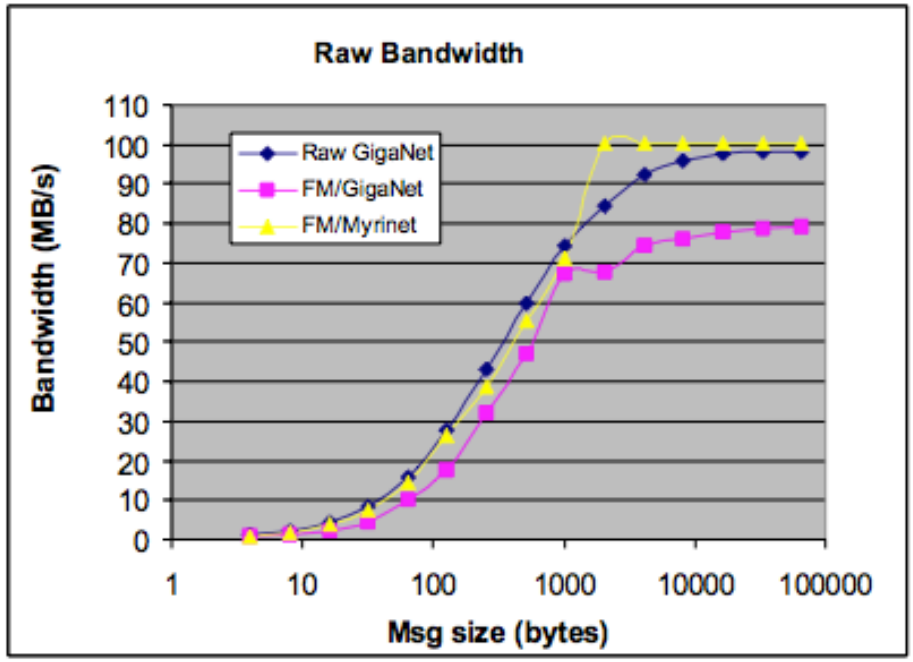


Figure 3 Raw Bandwidth

FM/VIA :: Testing

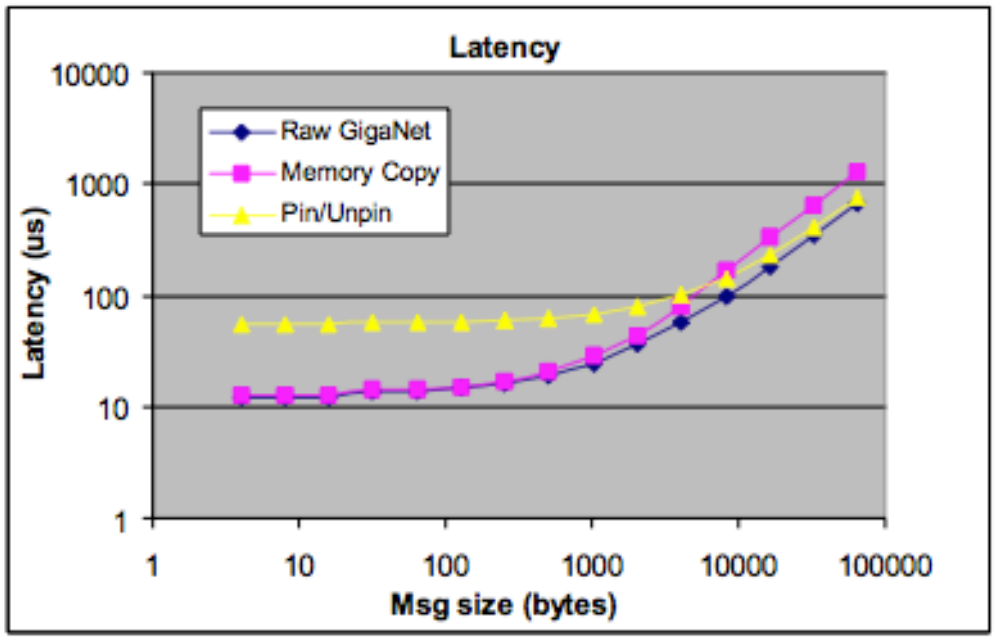


Figure 8

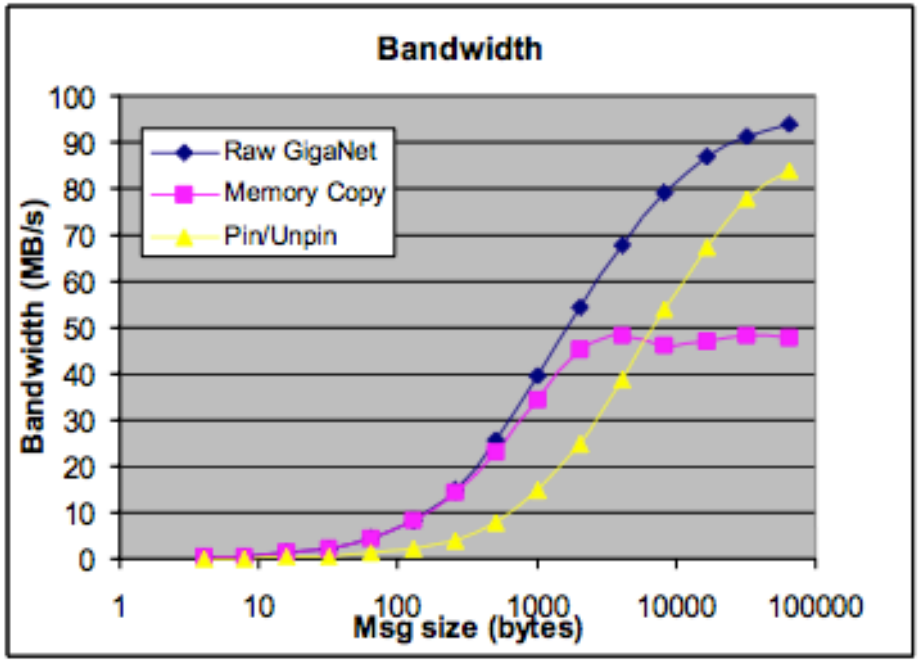


Figure 9

Effects of U-Net

- FM/VIA was not useful because it did not meet the requirement of fast end points
- FM/VIA goes on to influence InfiniBand, iWarp
  - Zero-copy