Summary

Conclusions

- Outstanding performance
  - 26 $\mu$s round trip — at 26% CPU utilization
  - 29 MB/s — at 2–9% CPU utilization, 96% of bus bandwidth

- Techniques
  - Direct application access to network hardware
  - Sender-managed reception buffers
  - Automatic message reassembly and DMA

- Protection
  - Protection keys support untrusted applications
  - Safe multi-process access
Summary

Lessons learned - III

- High-performance networking on stock hardware demands attention to lots of details
Summary
Lessons learned - II

- Host versus interface function split is critical
  - Where to store shared interface-control data
  - DMA versus DIO
Summary

Lessons learned - I

- Out-of-order packet delivery is manageable
  - Self-placing packets
  - Efficient packet counting

⇒ Can use faster interconnects
Summary

Related work

- Cranium: McKenzie et al. (1994)
- Active Messages: von Eicken et al. (1992)
- Others: see paper
Performance

Loopback backwidth
says your EPS file is not valid, as it calls setpagedevice
Effect of resource contention on bandwidth

Bandwidth (Mb/s)

Message size (bytes)

4K packet size

256 byte packet size

separate host loopback

Effect of resource contention on bandwidth.
Performance

Bandwidth

Bandwidth for various packet sizes

1/2 Max. BW @ ≈1K bytes
Performance

Large packets

1-way latency with 4K packets

No metadata
Performance

Latency (multiple small packets)

1-way latency with ≤256 byte packets

- 60 bytes of metadata
- 4 bytes of metadata
- no metadata

Latency (µs)

Message size (bytes)

17 µs
Performance

Latency components (1-packet messages)

Average 10,000 trials
< 1% variation
Performance
Measurement conditions

- HP 9000 Series J200 (100 MHz) workstations
  - Cache-coherent I/O
  - DMA: 106 MB/s in, 32 MB/s out
  - HP-UX Version 10.00
- 40 MHz 32-bit graphics I/O bus
- 80 MB/s Myrinet, LANai 4.0

![Diagram showing round trip time calculation](image)
Design
HP-UX Device Driver API

- Interface device *open*(2)
- Administrative (Super User) *ioctl*(2) commands
  - Flags: get status, set control
  - Load firmware
- Hamlyn (application) *ioctl*(2) commands
  - Open slot, bind and wire down buffers
  - Open terminus, bind and wire down buffers
  - Change slot protection key
  - Sleep until message arrives
- Interface device *close*(2): deallocate process’ slots and termini, unbind and unwire buffers
Design
Library protocol API (record-stream example)

- Provides
  - In-order delivery
  - Buffer management
  - High-level flow control

- Receiving application
  - Gets data-buffer pointer
  - Releases buffer after use
Design

Transmission termini and message areas

Hamlyn interface

- Memory mapped work queues
- Termini
- Outgoing packets
- Message offset and length
- DMA request
- Message area
- Message count
Design
Packet counting

if ((Accumulator += Delta) == 0) notify();

Delta
-3

Incoming packets

Card SRAM
Slot structure

Metadata index from packet header

Host DRAM
Metadata area

Accumulator
Design

Message-arrival notification

Hamlyn interface

Slots

Message’s last packet
Notification control block

Circular notification queue
# Design

## Packet format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination host ID</td>
<td></td>
</tr>
<tr>
<td>Slot number</td>
<td></td>
</tr>
<tr>
<td>Metadata index</td>
<td></td>
</tr>
<tr>
<td>Protection key</td>
<td>(64 bits)</td>
</tr>
<tr>
<td>Data offset</td>
<td></td>
</tr>
<tr>
<td>Data length</td>
<td></td>
</tr>
<tr>
<td>Delta (counter)</td>
<td></td>
</tr>
<tr>
<td>Metadata length</td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td>(optional metadata)</td>
</tr>
</tbody>
</table>

(data follows here …)

32 bits wide
Design

Reception: packet arrival

Hamlyn interface

- Incoming packet
- Slot index in packet
- Offset in packet

Metadata area
- DMA into host RAM

Message area
- Packet payload goes here
Design

Reception: slots and message areas

- Hamlyn interface
- Message areas
- DMA
- Interface control registers
- Slots
- Protection key
- Message area base, bounds
Design

Interface card layout (EISA size)
Design

Interface card layout

5”

3”
Design

Interface card organization

Host I/O bus

Host DMA

LANai controller

Packet staging area

Work queues

Slot table

Link DMA

Link DMA

interconnect
Design

Hamlyn system structure

Application program

Interface library: higher-level protocols

Interface library: low-level procedures

HP-UX OS

Device driver

Workstation hardware

Interface card

Myrinet switch
Introduction

Tactics

- Zero-copy protocol
- Message segmentation/reassembly in interface
- Bypass OS during normal transfers
- Optional features are “off the critical path”
- Myrinet prototype (80 MB/s, packet-switched)
Introduction

Strategy

- Sender-managed reception buffers
- Direct, application access to interface hardware
- Separate data transfer and arrival notification
Introduction

Assumption: interconnect is not a LAN

- Packet damage/loss exceedingly rare
- Physically secure
- Flow controlled

⇒ Network is as reliable as a backplane (no errors)

- Out-of-order packet delivery allowed
Introduction

**Goals**

- Interconnect scalable, commercial multi-computer
  - For OS, database managers, and “middleware”
  - Protects mutually suspicious applications
  - Uses “off the shelf” components

- Low latency ($\leq 20 \mu s$), high bandwidth ($\geq 30$ MB/s)

- Network hardware-to-software interface is key
Introduction

Outline of talk

- Introduction
- Design
- Performance
- Summary
An implementation of the Hamlyn sender-managed interface architecture

Greg Buzzard
David Jacobson
Milon Mackey
Scott Marovich
John Wilkes

Hewlett-Packard Laboratories