Summary

Conclusions

- Outstanding performance
  - 26 µs 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

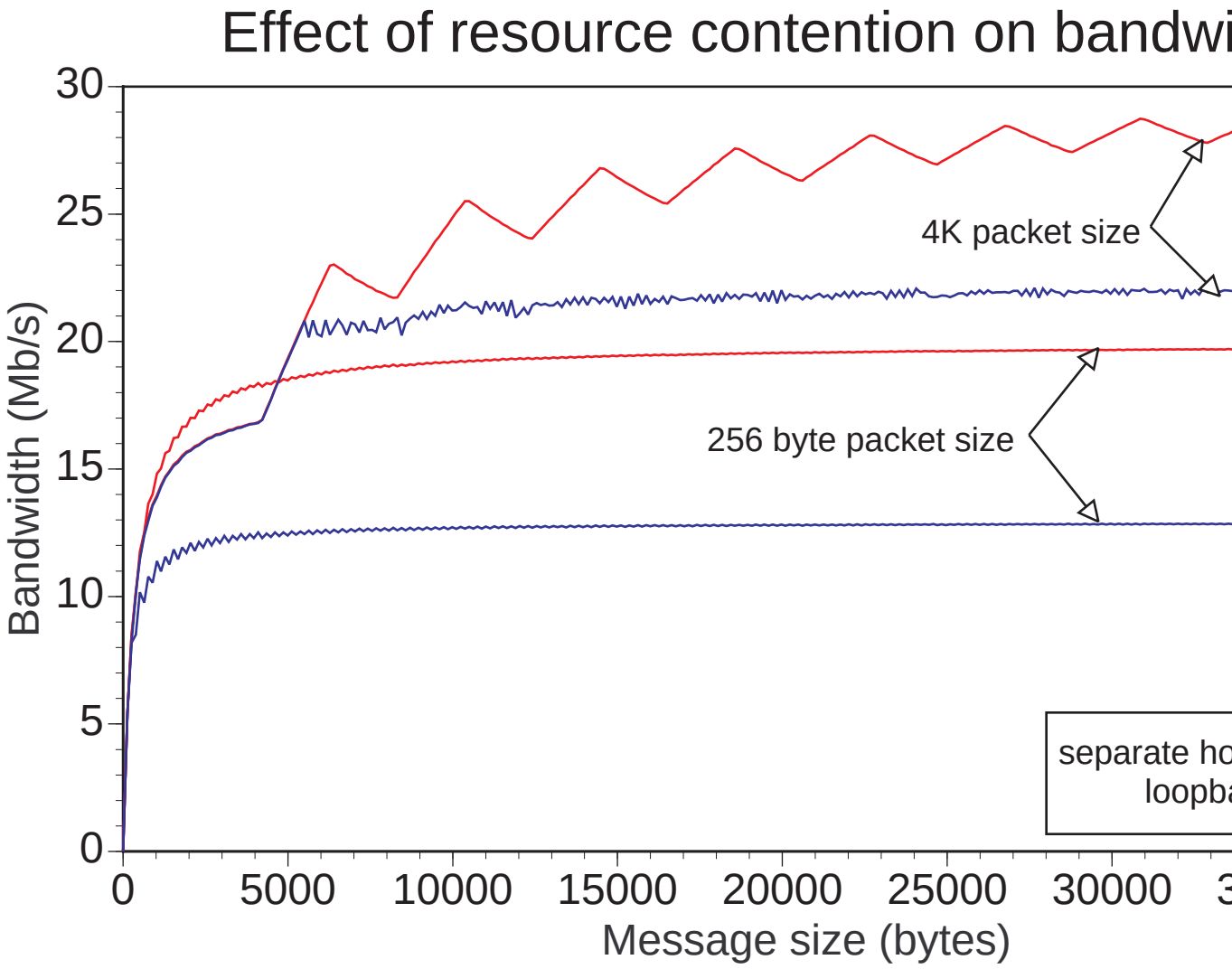
- Cranium: McKenzie *et al*. (1994)
- Active Messages: von Eicken *et al*. (1992)
- U-Net: von Eicken *et al*. (1995)

- Others: see paper

Performance

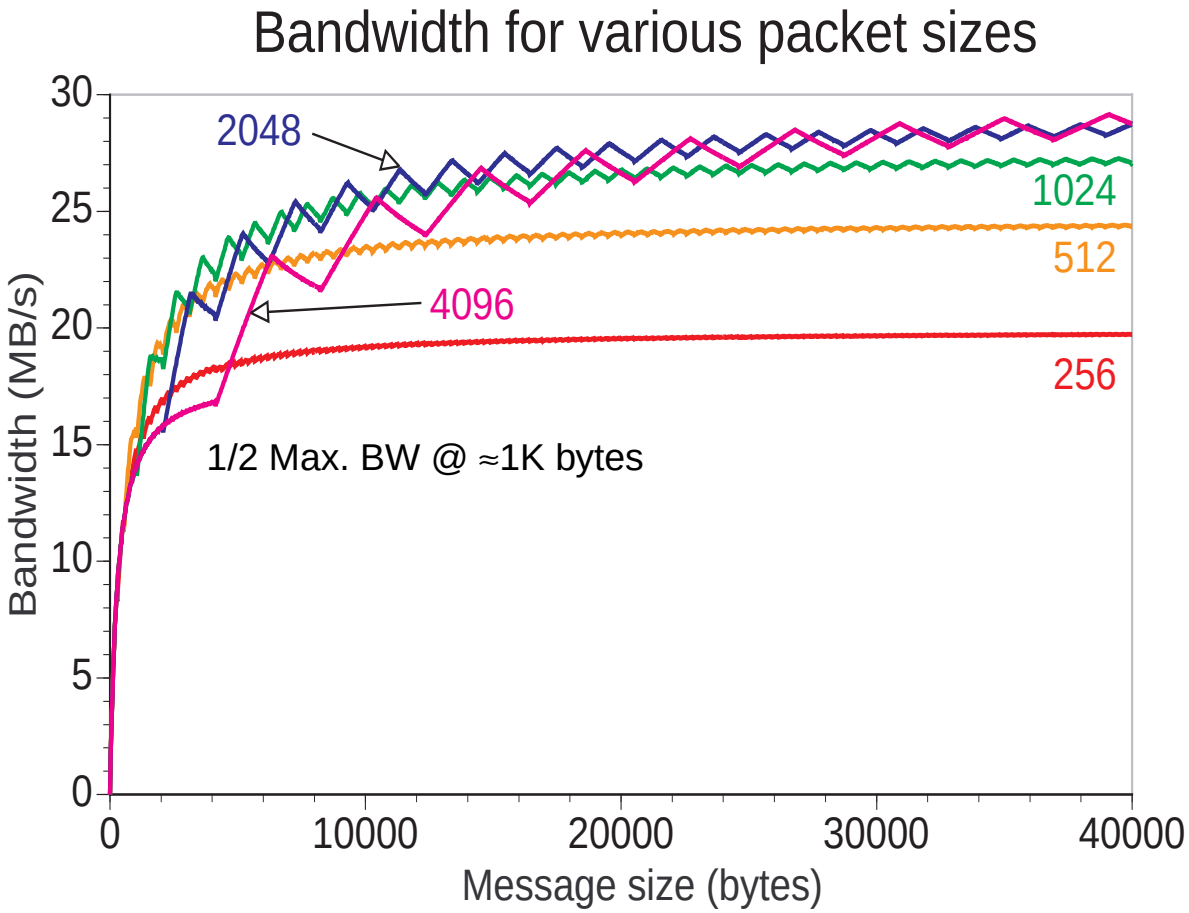
Loopback backwidth

Adobe’s PostScript Language Reference Manual, 2nd Edition, section H.2.4 says your EPS file is not valid, as it calls setpagedevice


Effect of resource contention on bandwi
Bandwidth (Mb/s)
Message size (bytes)
0
5
10
15
20
25
30
0
5000
10000
15000
20000
25000
30000
4K packet size
256 byte packet size
separate ho
loopba

Performance

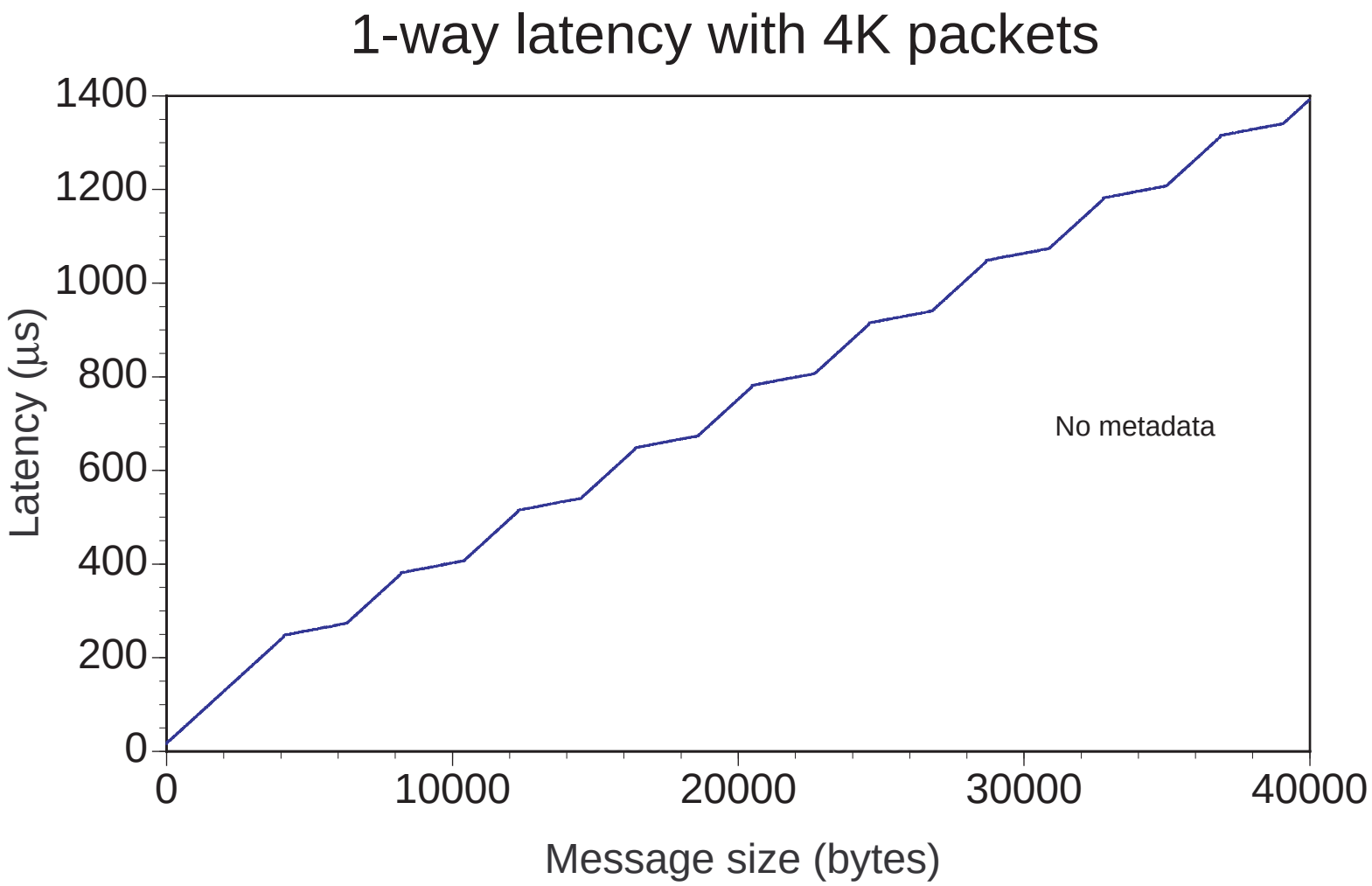
Bandwidth



HEWLETT PACKARD

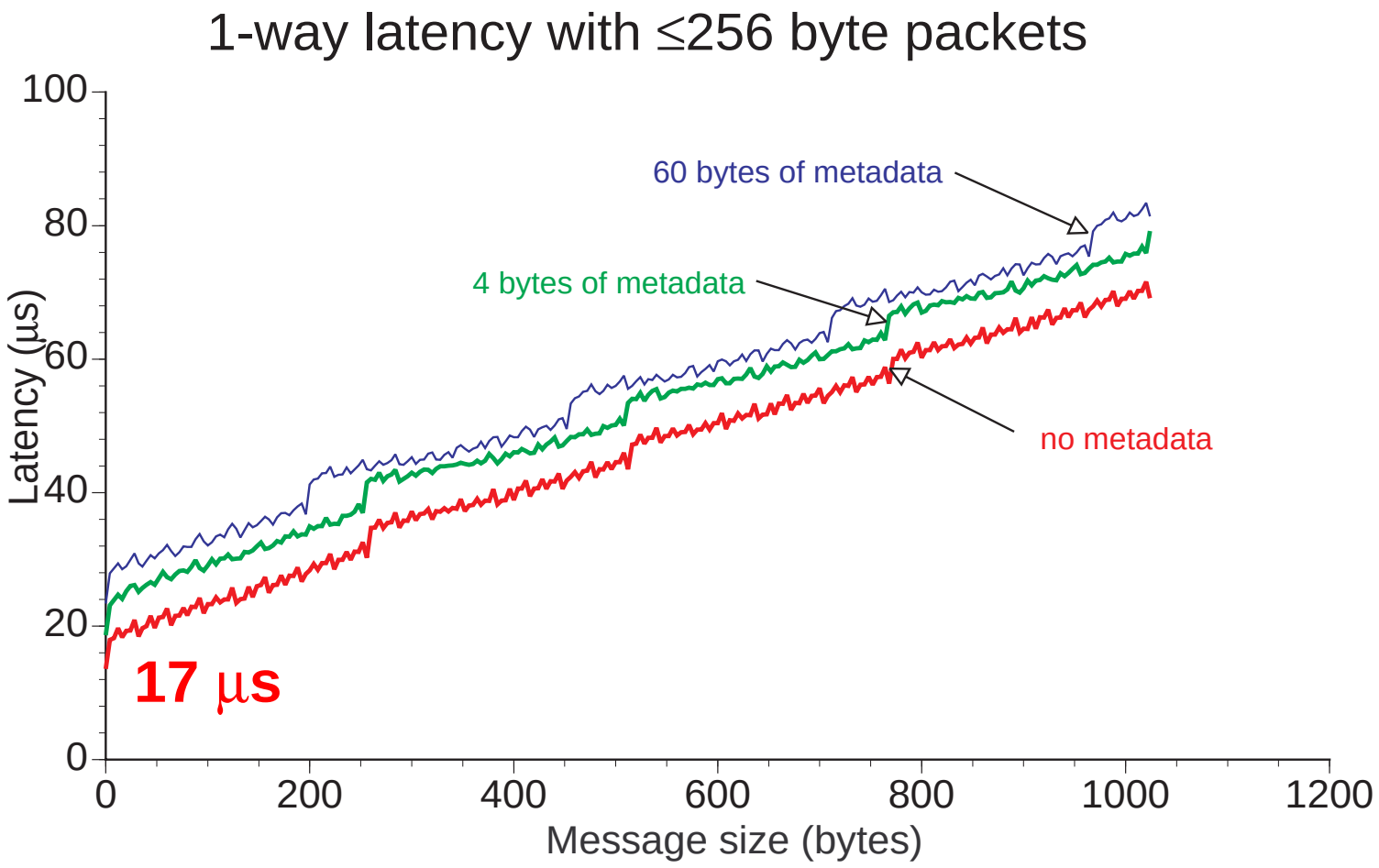
Performance

Large packets



Performance

Latency (multiple small packets)

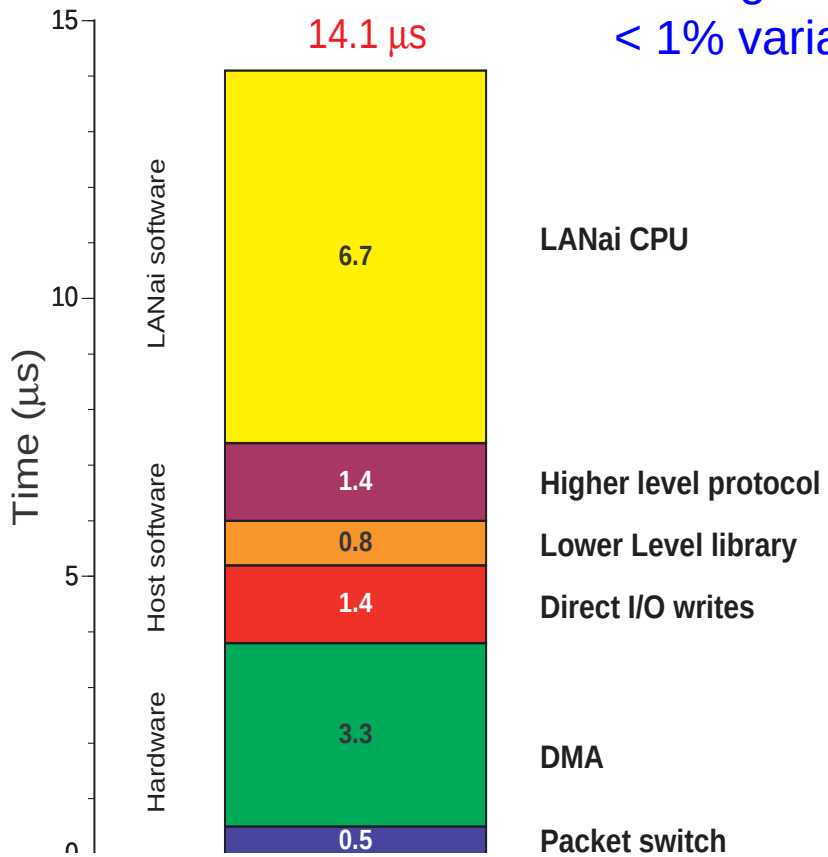


HEWLETT PACKARD

Performance

Latency components (1-packet messages)

Average 10,000 trials
< 1% variation

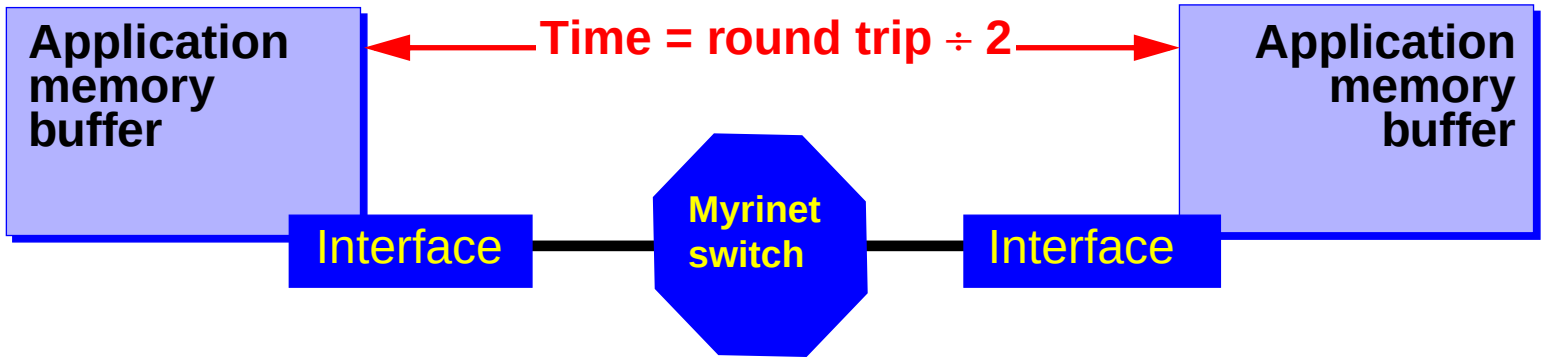


HEWLETT PACKARD

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



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

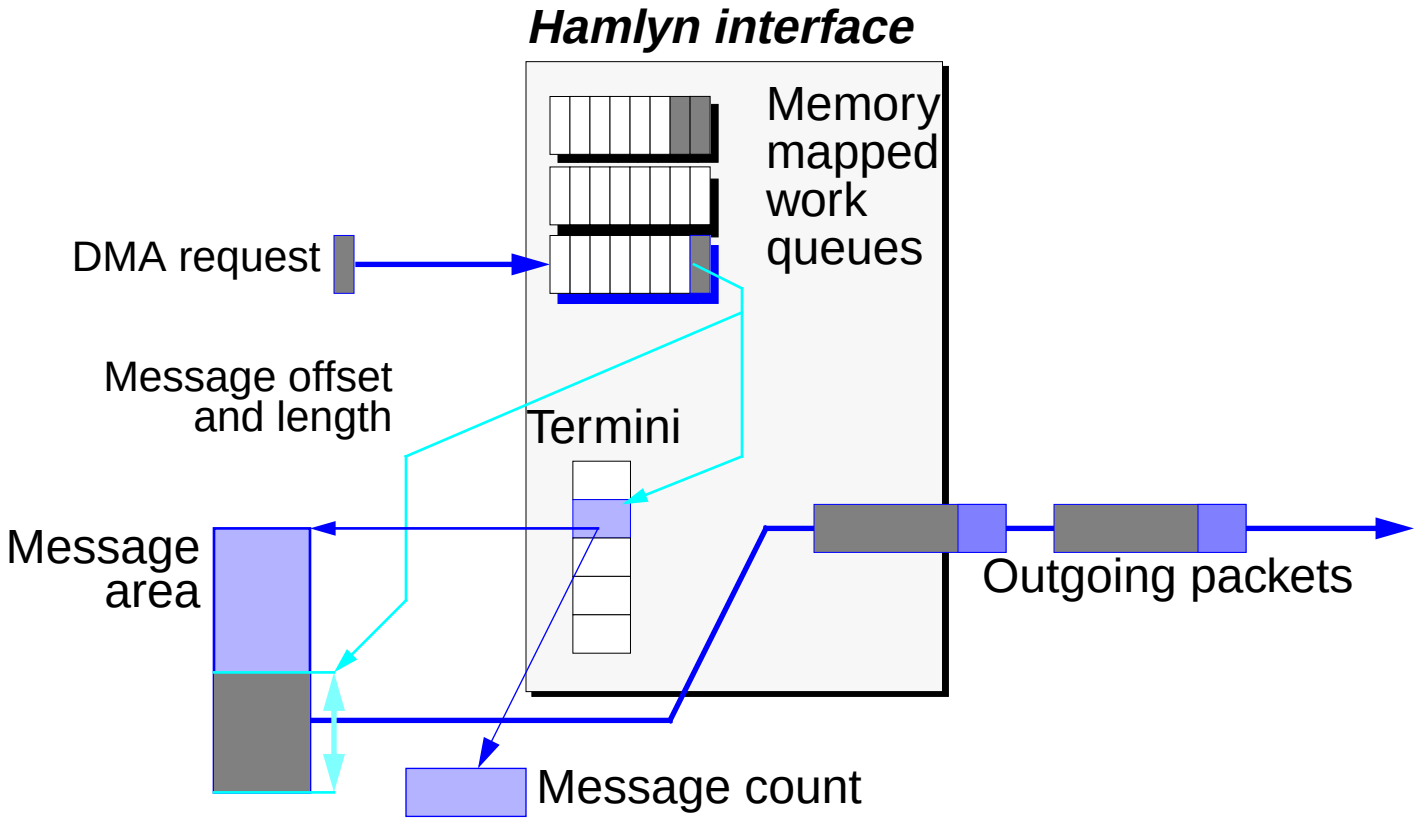
Design

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

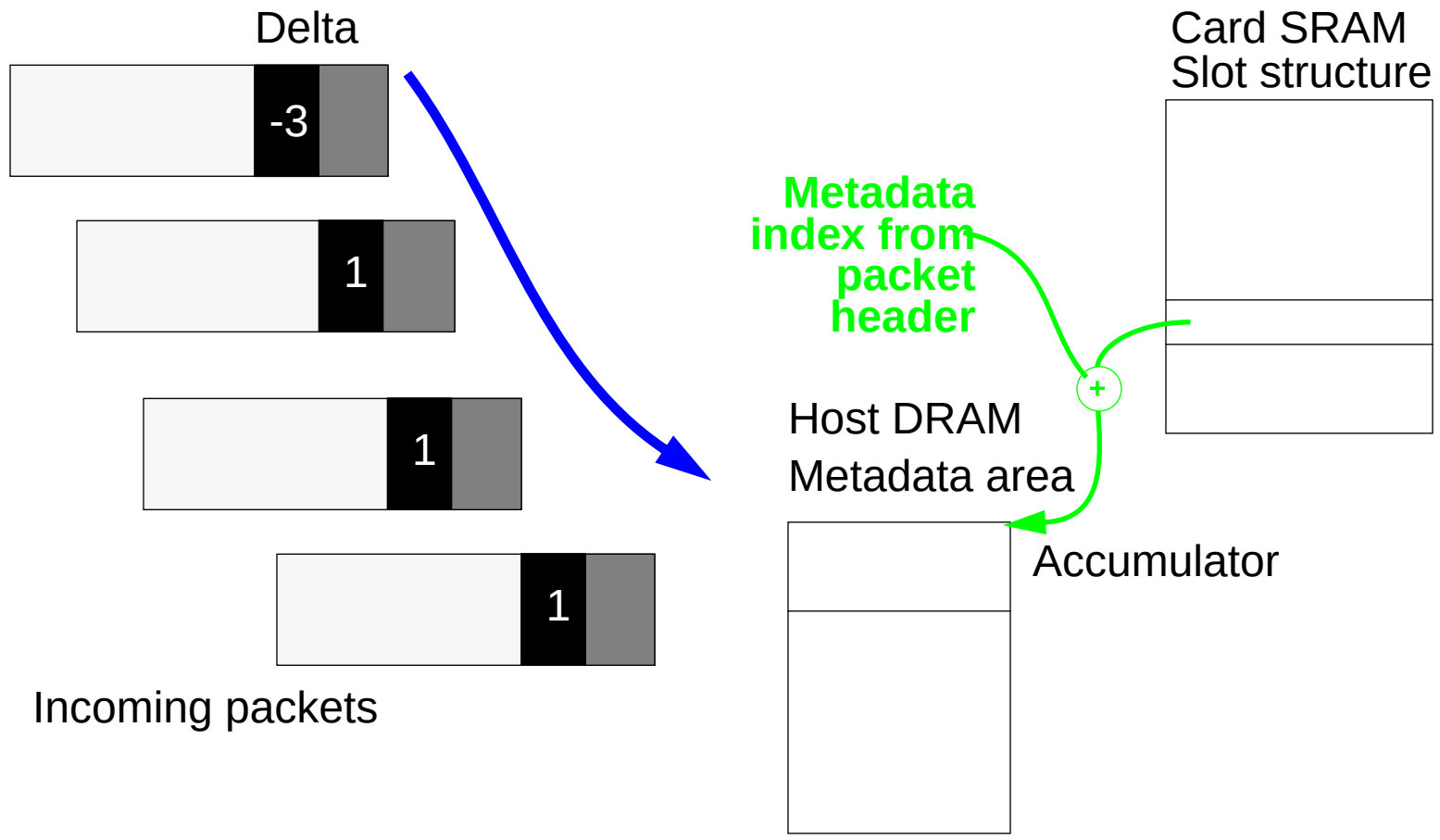
Design

Transmission termini and message areas



Design

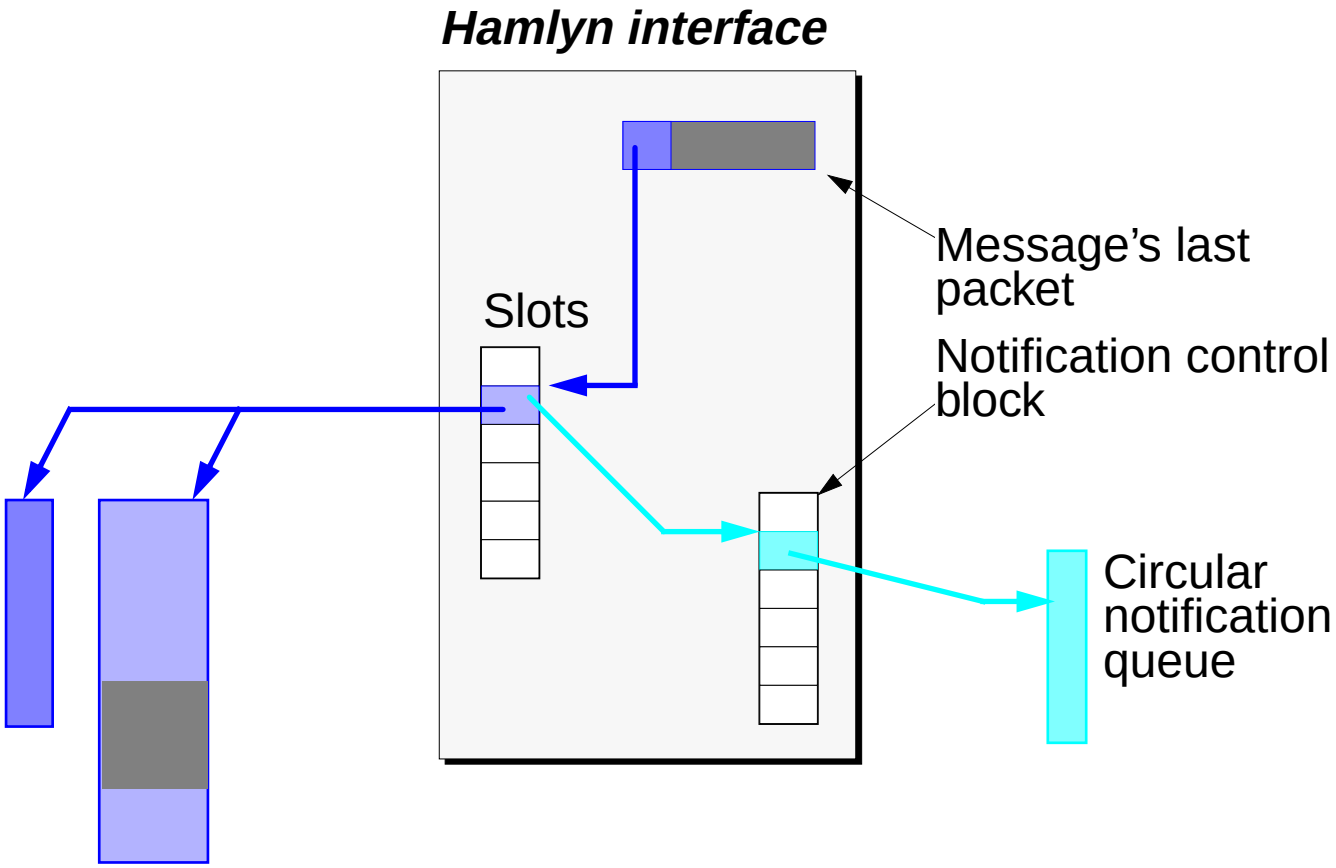
Packet counting



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

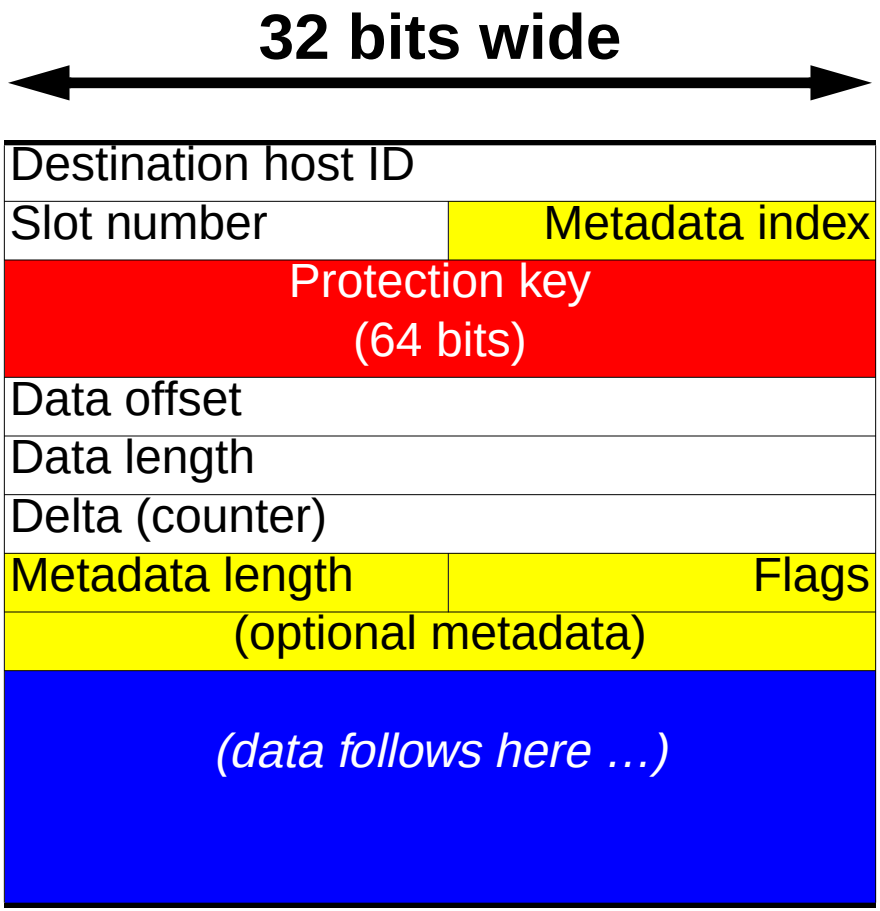
Design

Message-arrival notification



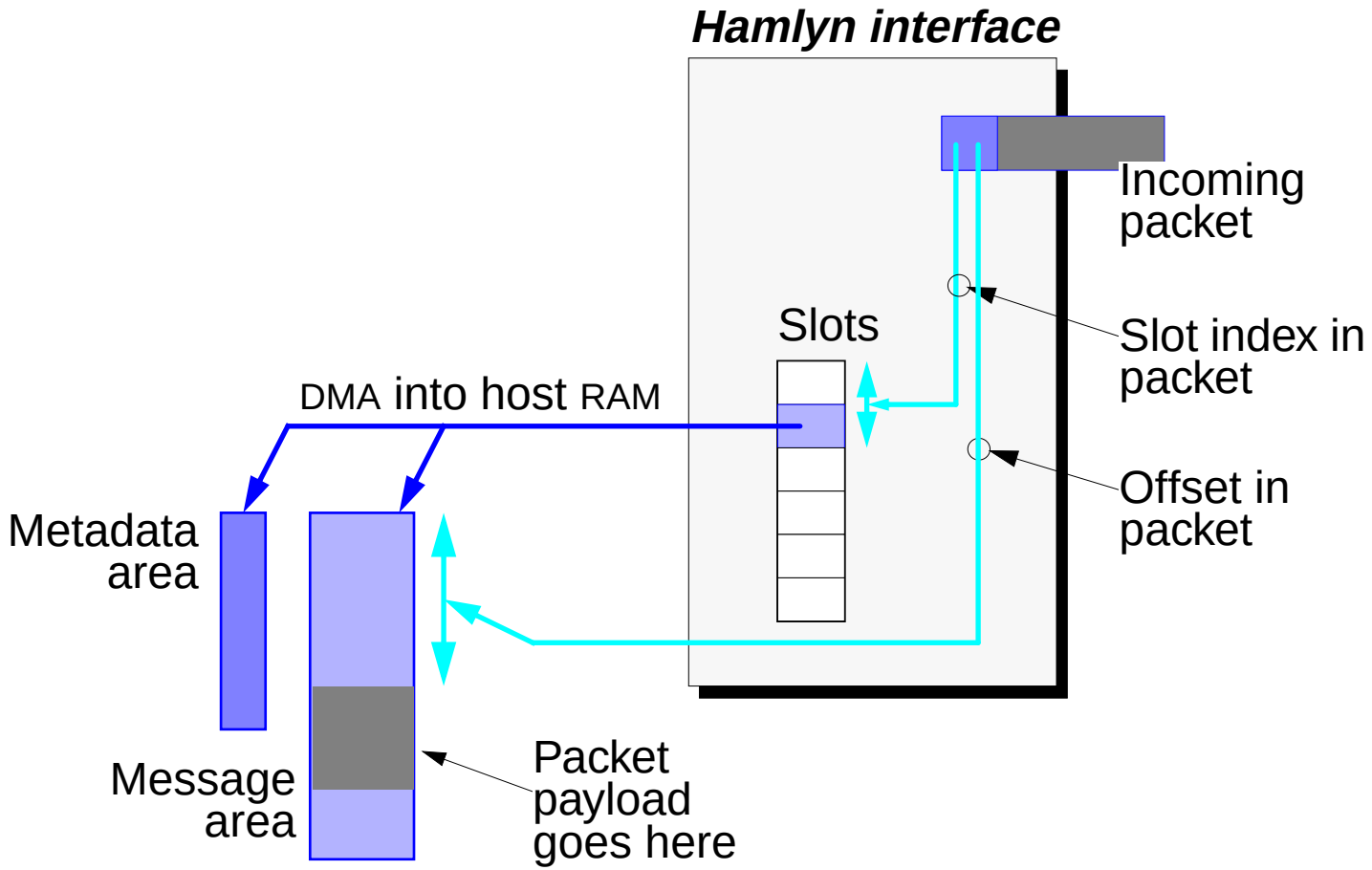
Design

Packet format



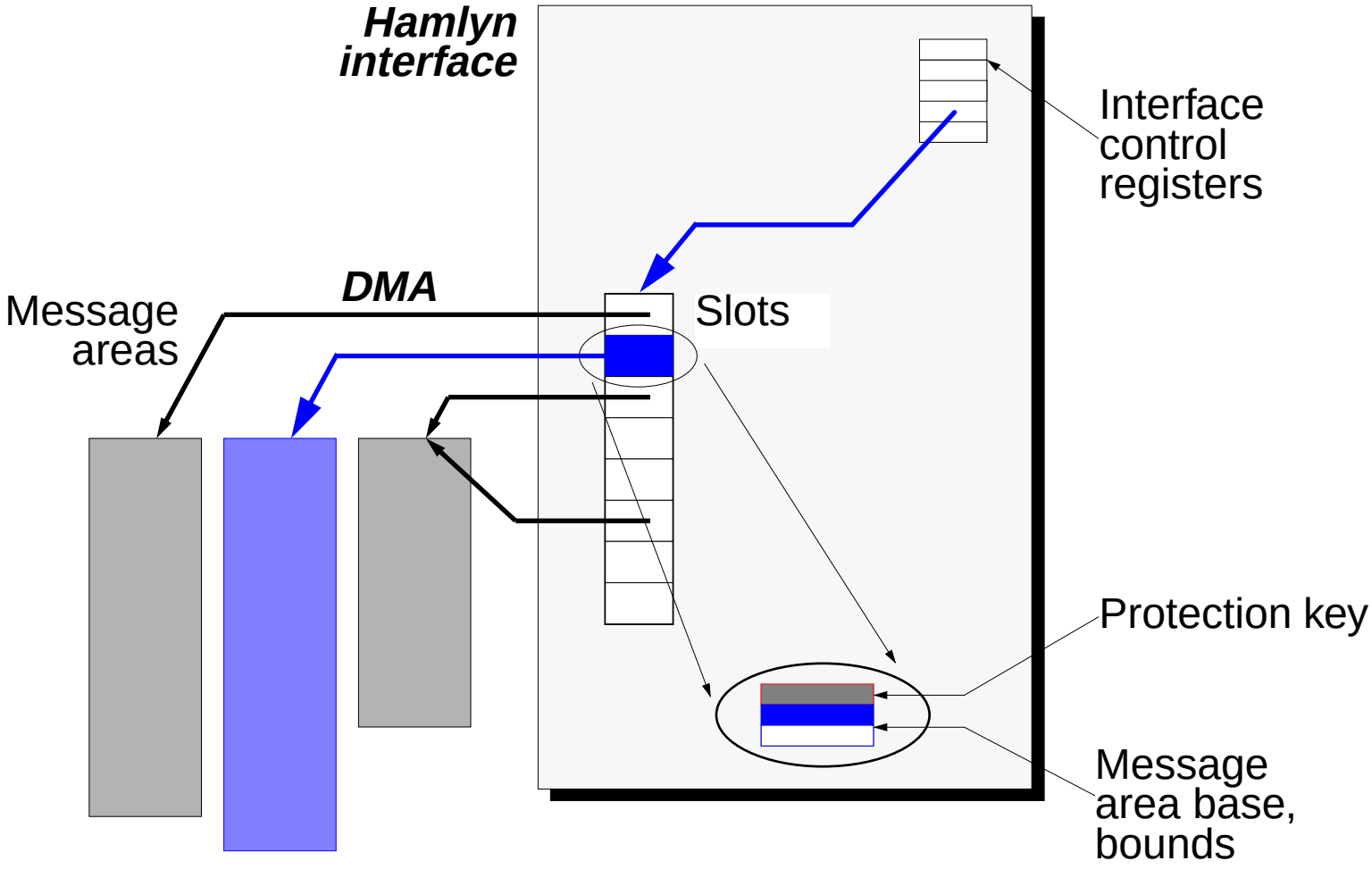
Design

Reception: packet arrival



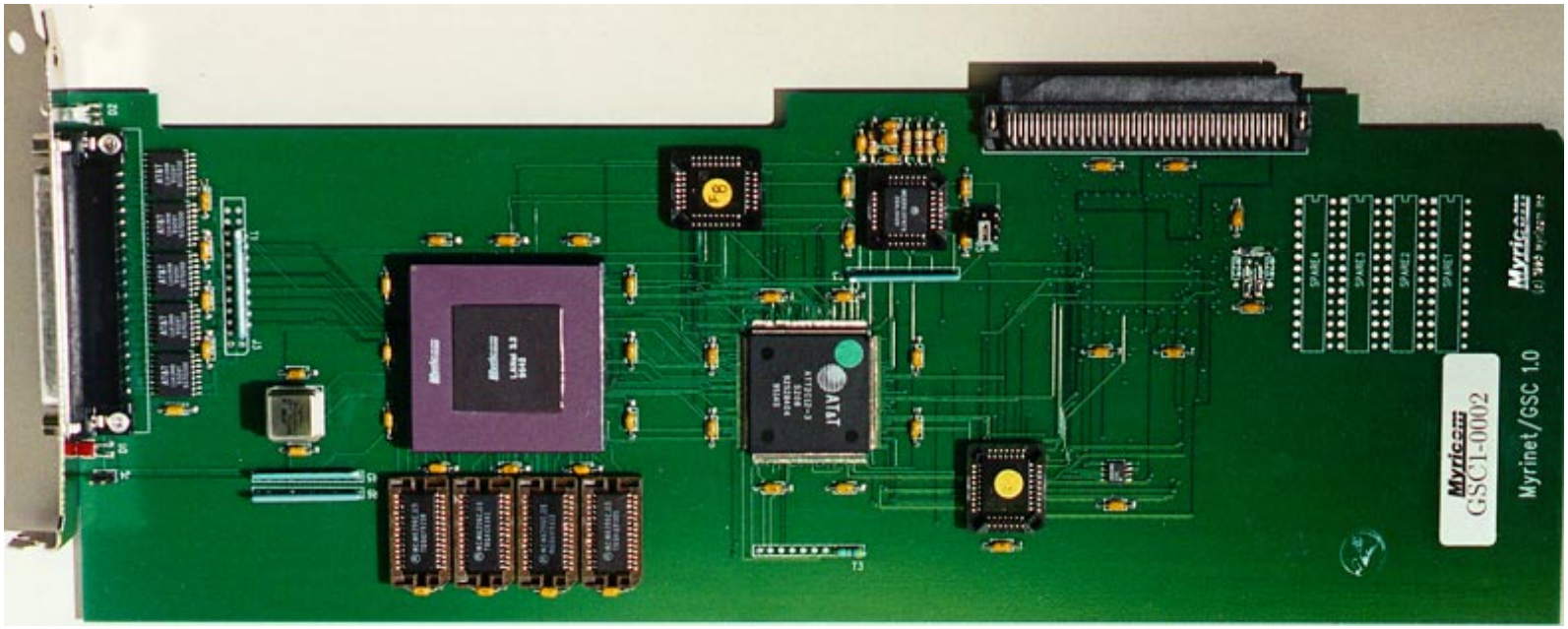
Design

Reception: slots and message areas



Design

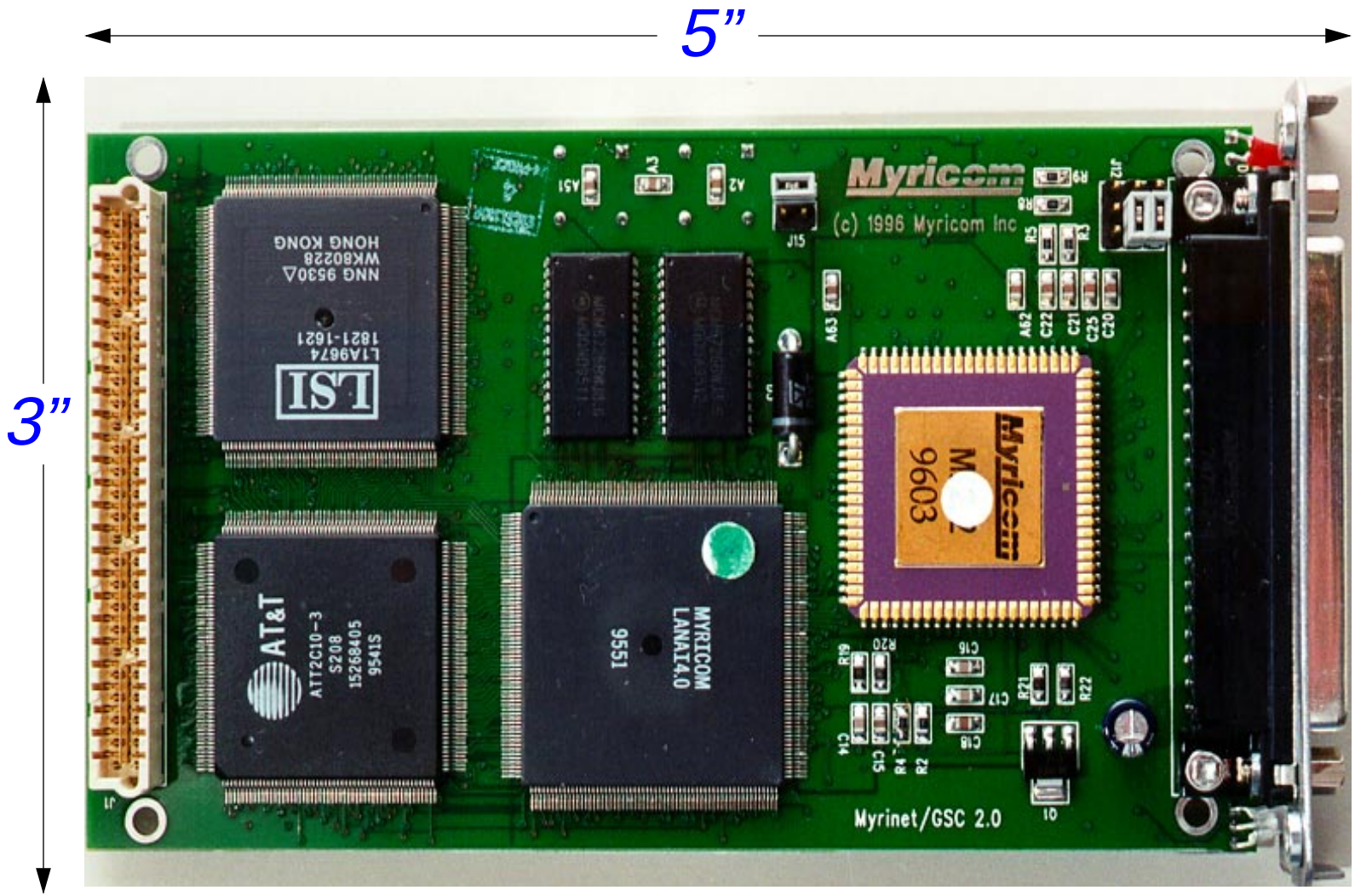
Interface card layout (EISA size)



HEWLETT PACKARD

Design

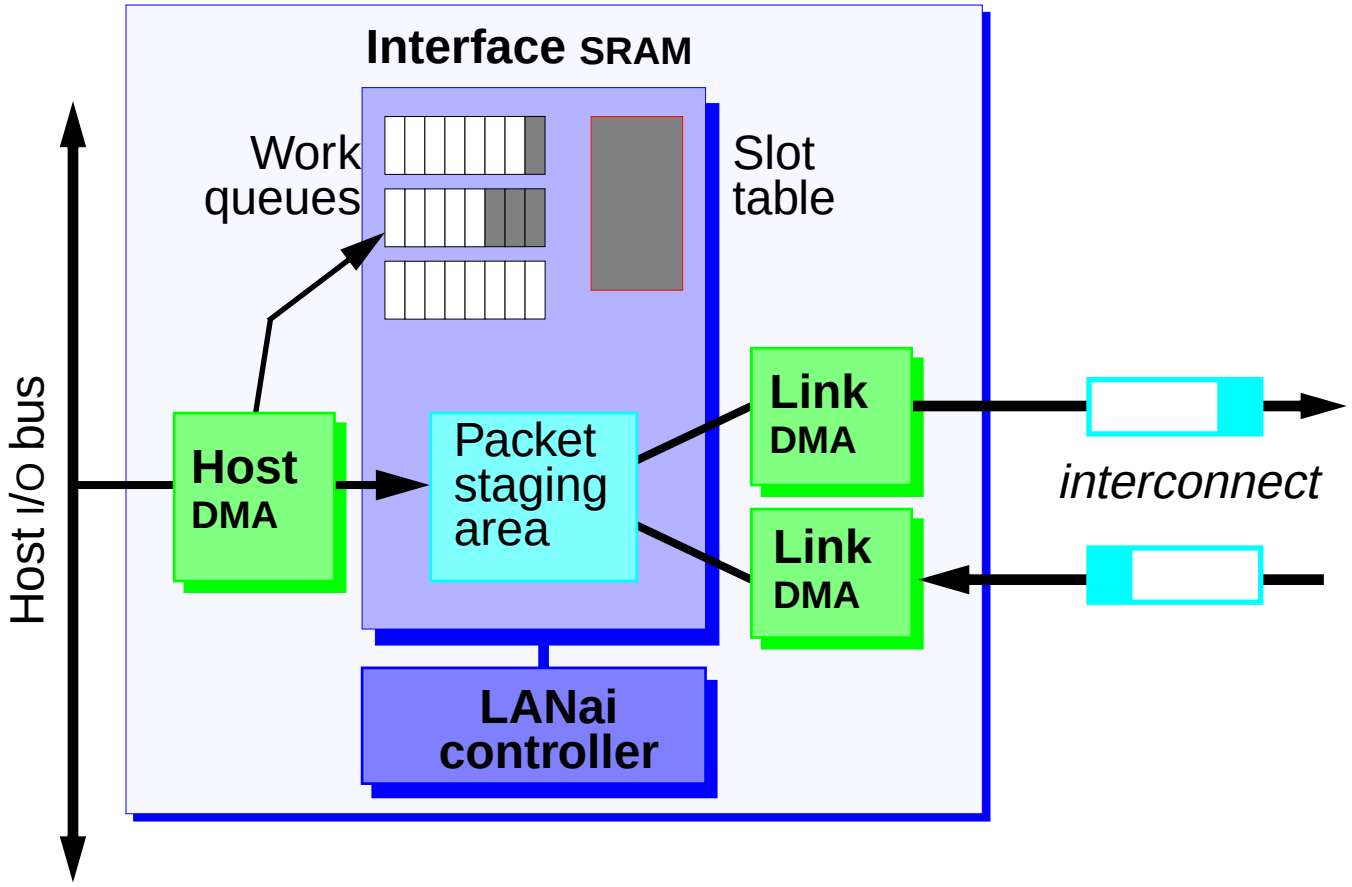
Interface card layout



HEWLETT PACKARD

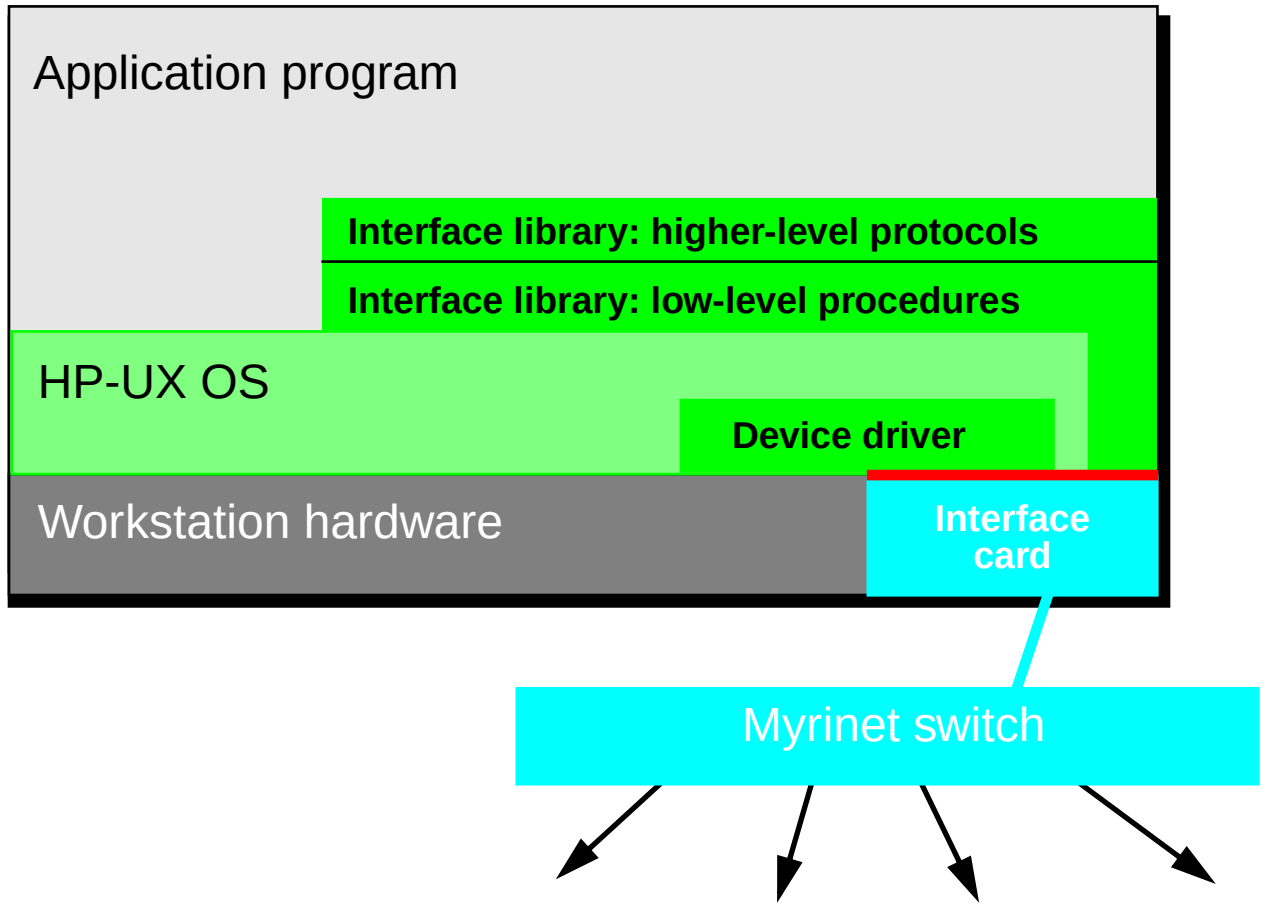
Design

Interface card organization



Design

Hamlyn system structure



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 (≤ 20 μs), high bandwidth (≥ 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

HEWLETT PACKARD