A COMPREHENSIVE TAXONOMY AND SYSTEM ARCHITECTURE FOR ATM HOST-NETWORK INTERFACES

Bo-kyun Na

Dept. of Electrical Engineering and Computer Science
Syracuse University

May 12, 2000

Current Network Systems

The current networks are:

- Service dependence: Telephony, computer neworks
- Inflexibility: In adapting changes or new service requirements
- Inefficiency: Resource sharing between networks (Packet switching)

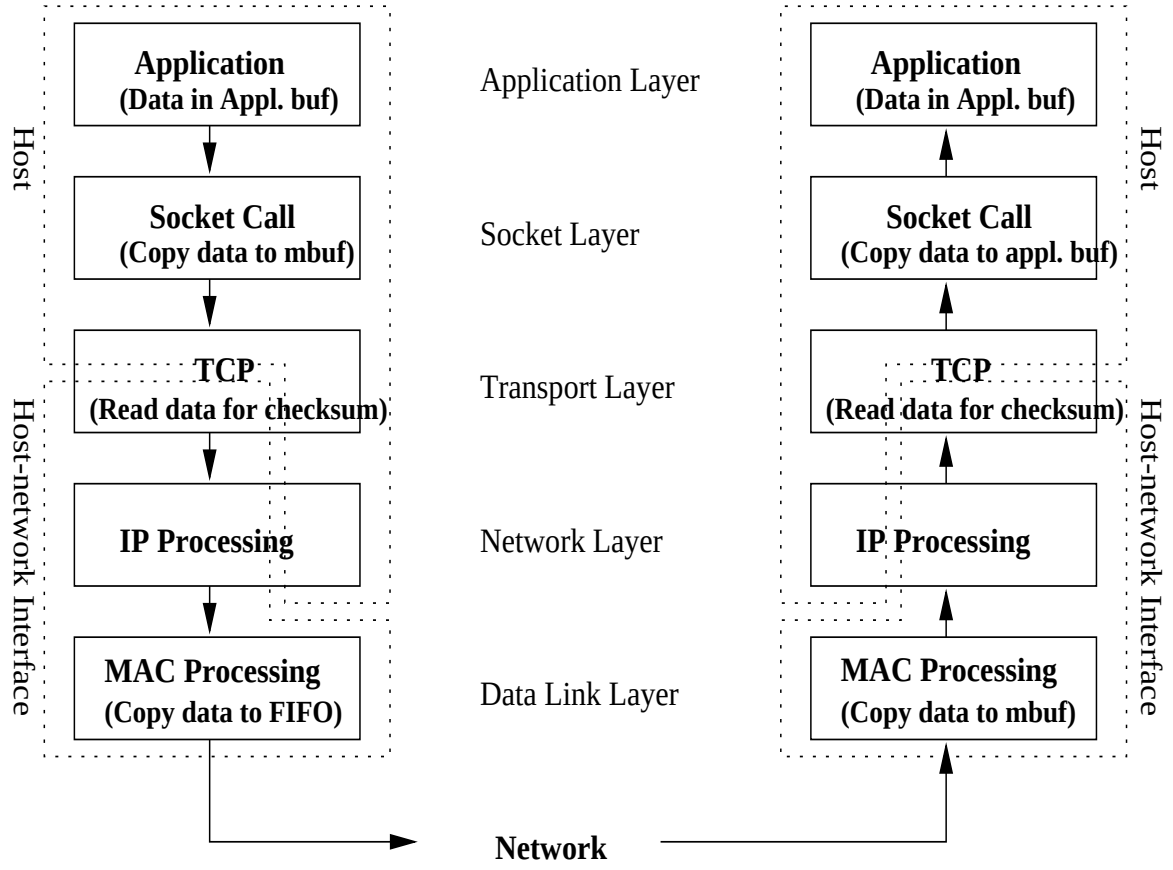
The single service-independent network is:

- Flexible and future-safe in bandwidth (Adaptive network)
- Efficient in the use of available resources (Statistical switching)
- Less expensive (Single nework <- manufacturing, operating, maintaining)

Review of Host-Network Interface Systems

Definition: A network I/O to connect host to network.

With TCP/IP suites,



Problems of Current HNI Systems

- Poor protocol implementations:
  - Complicated and slow transport protocol
- Poor interface between applications and HNI:
  - Unnecessary data copying
  - Low-speed system bus compared to CPU bus
- High overhead with OS functions and context switchings:
  - Process scheduling
  - Entities managements - timers, buffers, and connection states
  - Interprocess communication between application and protocol-implementing processes
  - Interrupts
  - Context switching

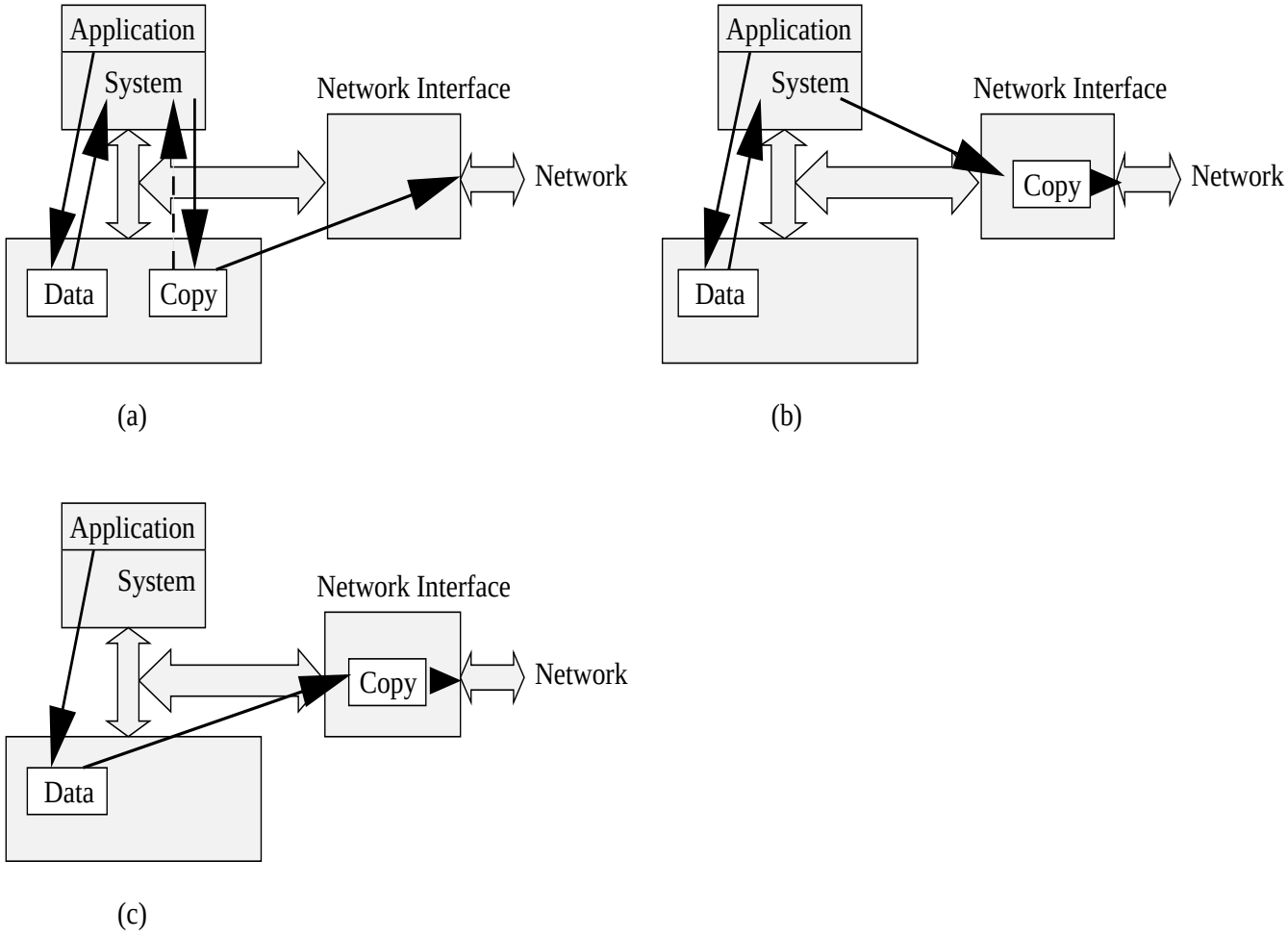
Conventional Classifications of HNIs

- By the number of data copying
- By the processing capability of HNI's
- By the memory access
- By the host access

The Number of Copies for Data Transfer

- Peter A. Steenkiste and *et al*.
- According to the system buffer, Host-network interfaces are classified as:
  1. Traditional 5-access data flow
  2. Outboard buffering
  3. DMA based
- Similarly, C. Dalton and *et al* analyzes TCP/IP protocol implementation in terms of minimizing data movement in memory (Afterburner HNI).

The Number of Copies for Data Transfer



Limitations of Current Taxonomies

- No comparative power
  - Only one category and non-hierarchical
- No predictive power
  - No inference of similarities or differences
- No comprehensive power
  - Authorization only the related architectures
  - LAN, WAN, MAN, tightly-coupled or loosely-coupled systems
- No systematic power
  - No combined aspects of the hardware and the software functional units

Research Objectives and Approaches

HNI taxonomy is motivated and directed to:

- Construct an architectural knowledge base as a foundation (Descriptive)
- Understand the position of architectures relative to one another (Comparative or hierarchical)
- Predict a new design as an optimized to the service requirements (Predictive)
- Understand the relations between architectural components and protocol processing functional units (Systematic)
- Accordingly, construct a comprehensive atlas of HNIs within the unified framework of the taxonomy (Comprehensive)

Skillicorn's Taxonomy on Computer Systems

- Three reasons for classifying architectures:
  - To understand what has already been achieved
  - To reveal possible configurations to suggest other possibilities
  - To allow useful performance models to be built and used
- More descriptive classification of the computer architectures
- Extends Flynn's taxonomy in the multiprocessor category
- A two-level hierarchy
  - Upper level classifies computer architectures based on the numbers of processors for data and for instructions and the interconnections between them: nIP - nIM, nDP - nDM, IPs - DPs, and DPs - DPs
  - Lower level to distinguish variants even more precisely and based on the state machine level of processors

System Model

Two-context taxonomy, as the approaches:

- Architectural context: The HNI architectures and their limitations
  - Main components to transfer data between host and network
    - Host CPU
    - Main memory
    - Cache memory
    - Protocol processor or controller
    - Network buffer
    - Switch
  - Connections and the interactions between these components
  - PMS notation to simplify the block diagram of HNIs
  - A system as an interconnected set of components or individual devices associated with a set of operations.

- Protocol context: The implementation of network protocol functions
  - Identifying all the software functions, required to transfer data (descriptive power)
  - Mapping the software functions into the components identified at the architectural classification (systematic power)
  - Identifying all possible techniques to develop a HNI (predictive power)
  - Protocol functional units:
    * Routing/switching
    * Data copying
    * Flow control
    * Error handling
    * Packetization/Depacketization

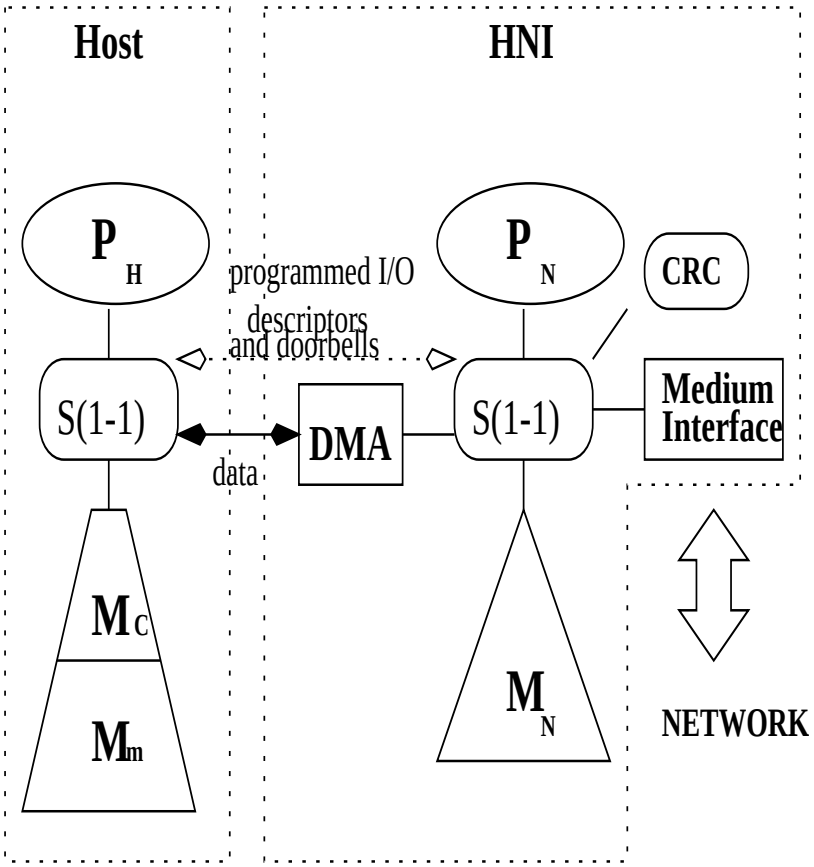
Performance Model

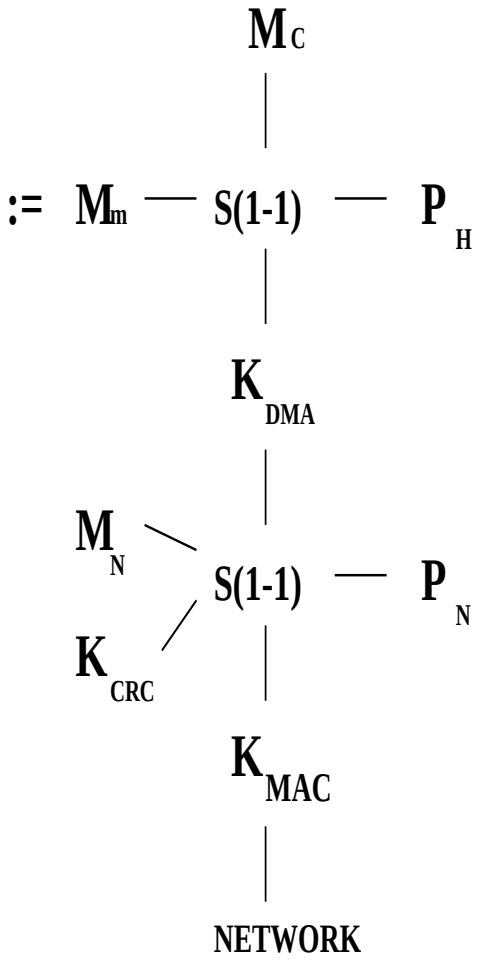
- Rearranging protocol functional units so that it takes less time (parallelization of protocol implementations)
- Reducing unnecessary data copying in main memory
  - No data copying to system communication buffer in kernel space
- Fast host interfacing (low-speed system bus v.s. high-speed CPU bus)
  - Increase the speed of system bus
  - Through other medium (host register, dual-ported memory)
- Hardware implementation of protocol functional units
  - On-board protocol processor
  - CRC checking or Flow control
- HNI implements protocol functional units
  - Avoiding system calls by OS and context switching

Illustrative Example: Jaguar

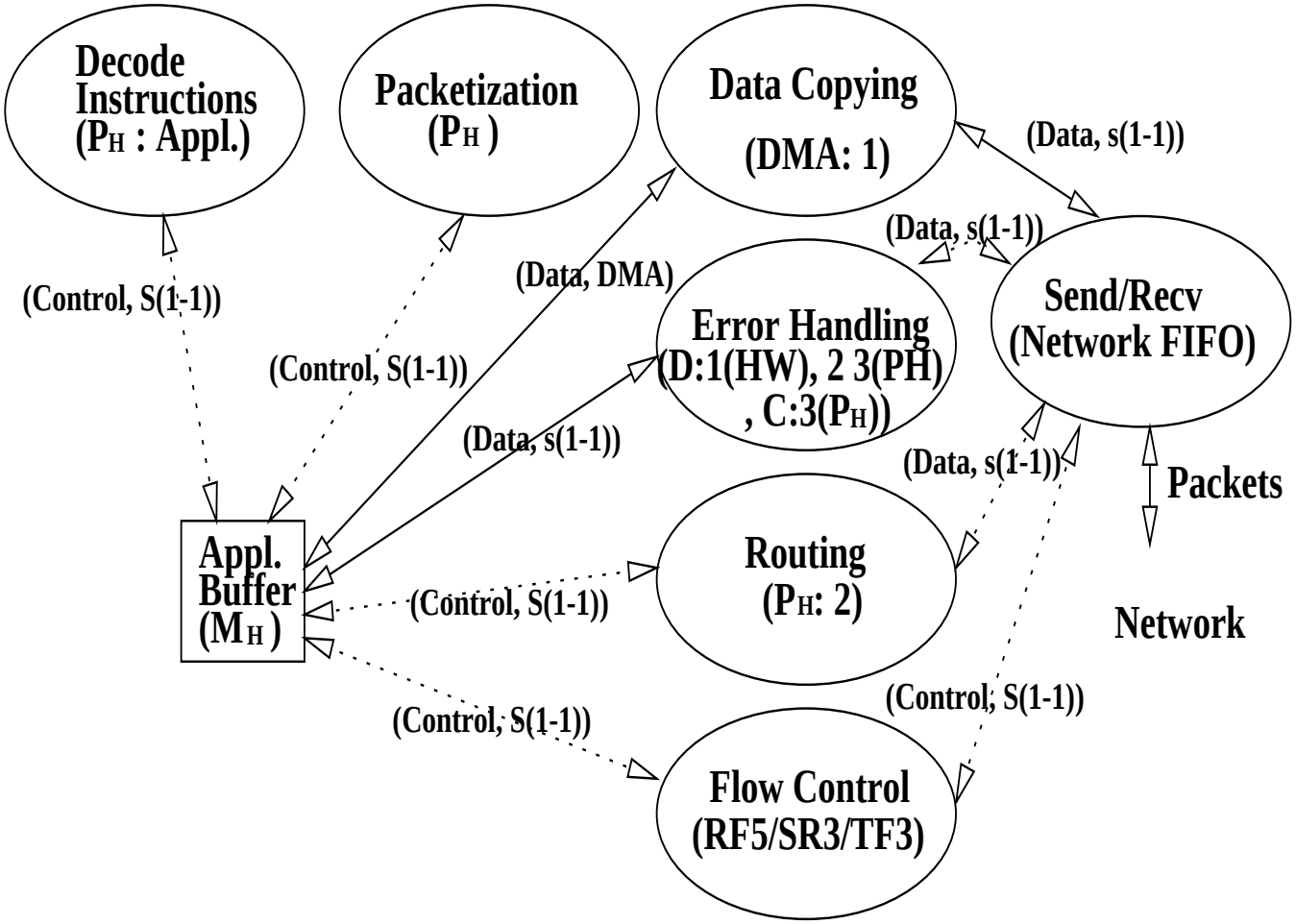
- Matt Welsh and David Culler in University of California at Berkeley
- Providing Java applications with efficient access to system resources
- Retaining the protection of the Java environment
- Allowing Java applications to directly manipulate memory outside of the Java heap such as communication buffer
- JaguarVIA, as a Java interface to the Berkeley VIA communication layer
- Implemented with Intel PentiumII, Linux OS, and Myrinet network interface board
- Shown performance improvement only by optimizing API

Architectural Context of Jaguar





Protocol-implementation Context of Jaguar



Syntactic Notation of Conventional HNIs

This table shows our syntax notation for host-network interface designs that have been reported in the literature.

Name	Architectural Level					Protocol Level				
	$P_H - P_N$	$P_H - M_N$	$M_C - P_N$	$M_C - M_N$	$M_H - M_N$	Packetization	Data copying	Flow control RFC/SR/TFC	Error handling D/C	Routing
WD 2840		PIO			DMA	1	3	1/1/1	1,2,3,4/3	3
Medusa	PIO	B_{mc}	PIO		B_{mc}	1	4	1/1/1	1,2,3/3	3
VuNet					DMA	3	3	1/1/1	1,2,3/3	4
Yes V2	reg	DMA			DMA	4	3	3/2/2	1,2,3,4/3	4
PHNI	PIO	PIO			PIO	4	2	1/1/1	1,2,3,4/3	4
MINI		PIO		B_{cc}	PIO	4	3	5,1/3,1/3,1	1,2,3,4/3	4
Myrinet	**PIO**	**PIO**			**DMA**	**1**	**2**	**5/3/3**	**1,2,3/3**	**2**
Nectar CAB	PIO	DMA			DMA	4	2	1/1/1	1,2,3/3	2
VMP NAB	reg	PIO			B_{mc}	2	2	4/2/2	1,2,3/3	3
SHRIMP				B_{cc}	DMA	2	3	1/1/1	1,2,3/3	3
ORBIT	reg				DMA	4	1	2/2/2	1,2,3,4/3	4
Jaguar	**PIO**	**PIO**			**DMA**	**1**	**1**	**5/3/3**	**1,2,3/3**	**2**

Design of Intelligent HNI

- Highest performance
- Reducing the heavy loads of the host
  - on protocol processing
  - on system peripheral bus
- Support host to take more-processing required jobs

Design Requirements

- In architectural context,
  - Protocol processing in HNI
  - A better connection component than a system peripheral bus
  - CRC checking by hardware component while data being transferred
- In protocol-implementation context,
  - Implementation in a pipelined or parallelized way
  - Routing - statistical switching
  - Data copying - direct transfer between M_H to network FIFO
  - Error handling - high speed CRC check for cell header and payload
  - Flow control - on-board protocol processor counts cells
  - Packetization - Fixed sized cell by P_N - message as a local data in host

Taxonomy-based Design Approach

- In architectural context,
  - $P_H - M_N$: Burst transfer by cache controller (B_{cc}) for status and control information, start address and length of data, and connection establishment parameters.
  - $M_C - M_N$: B_{cc} for data transfer.
- In protocol-implementation context,
  - The protocol processor packetizes in the fixed size.
  - Virtual circuit switching (ATM).
  - User space memory - network FIFO data copying scheme.
  - CRC-8 and CRC-32 by hardware, sequence number, MID, length indicator of data. FEC + ARQ for error correction.
  - Rate-based flow control.

Diagram for architectural context of intelligent HNI

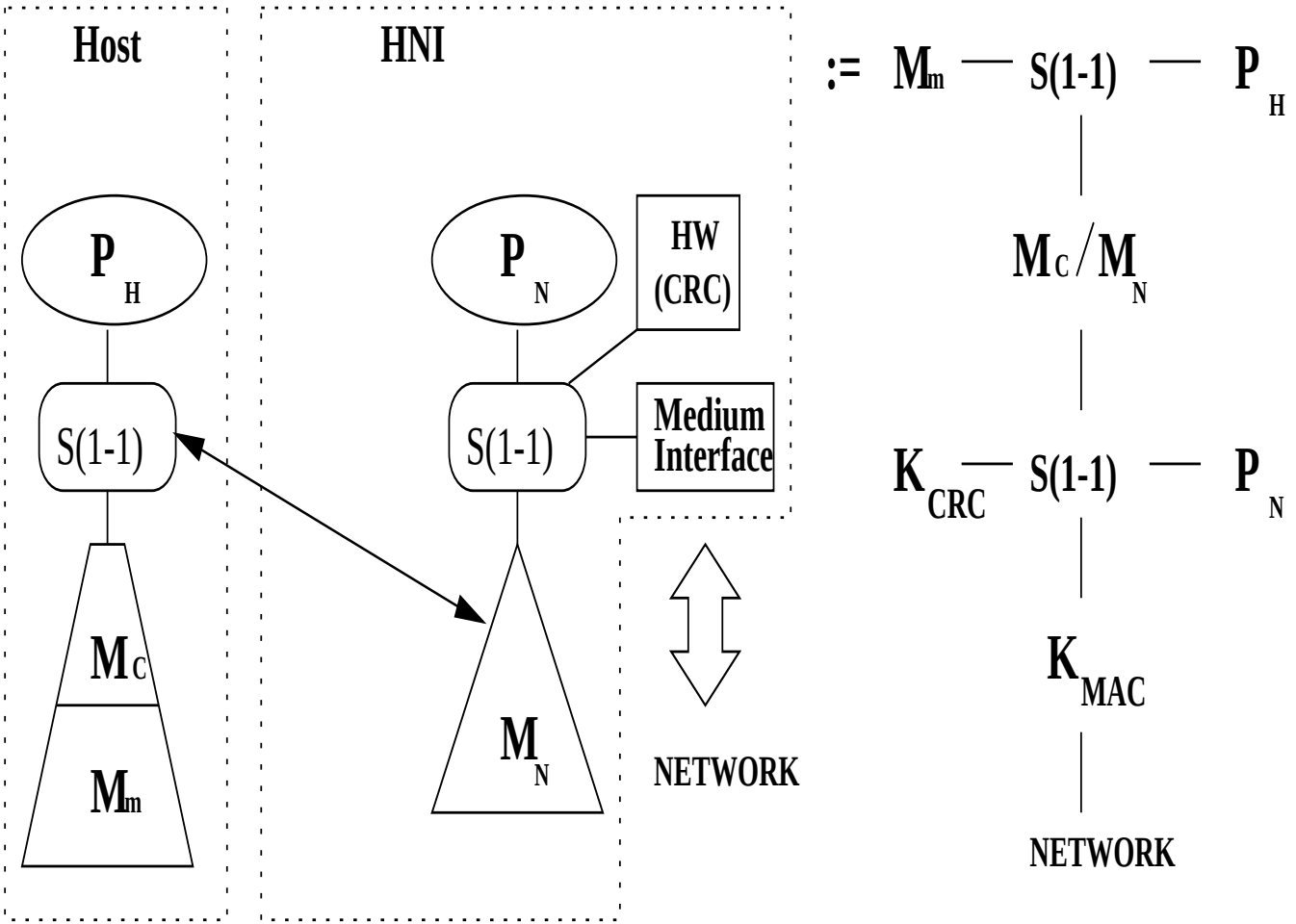
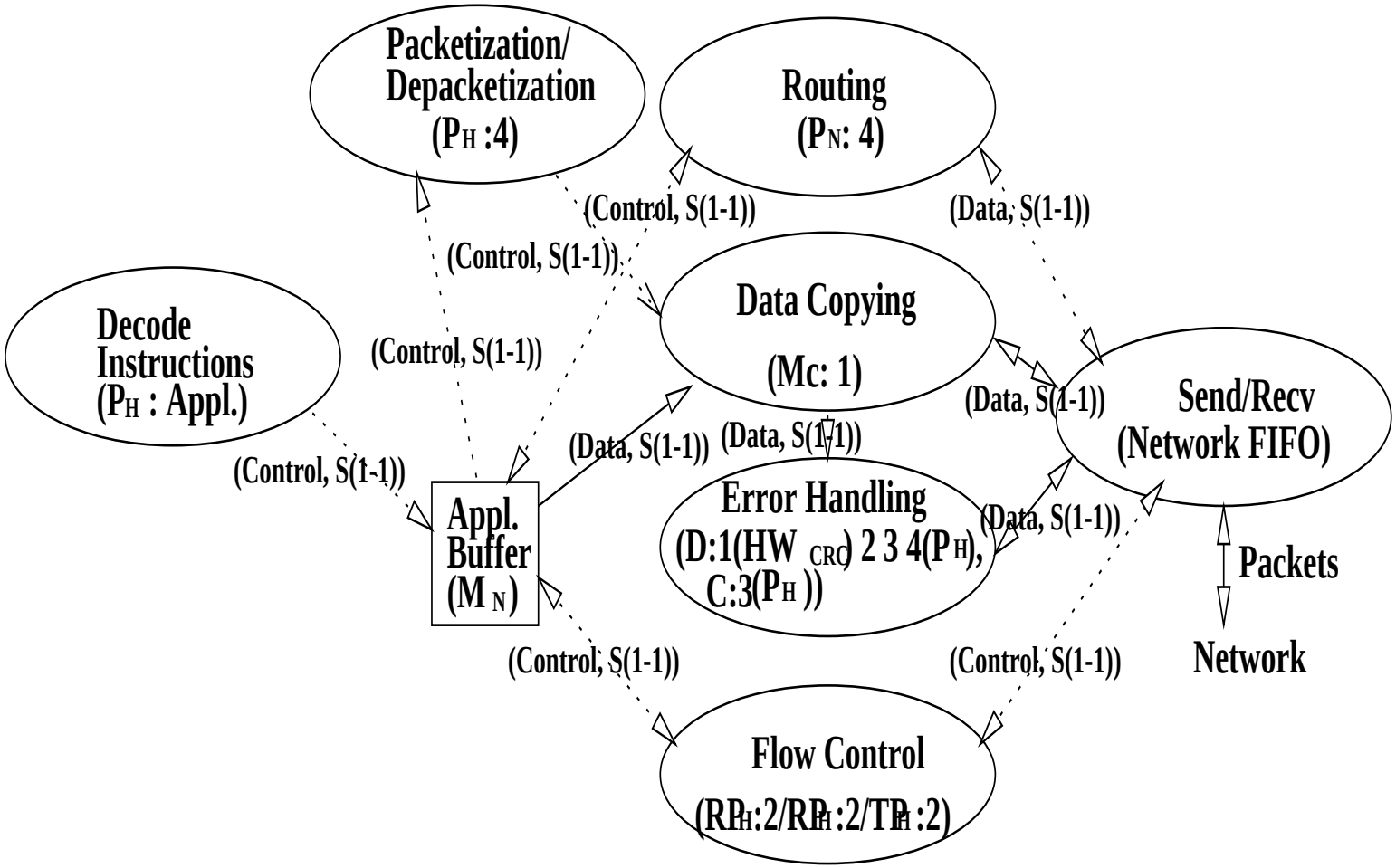
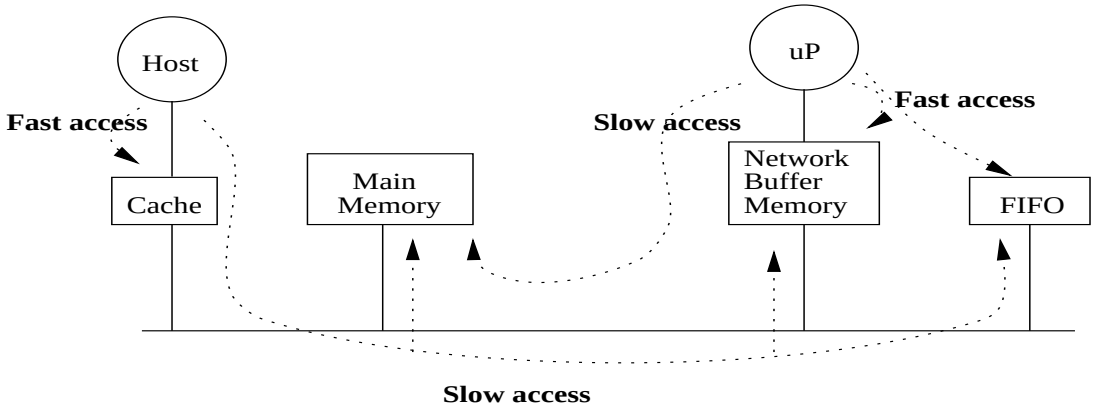


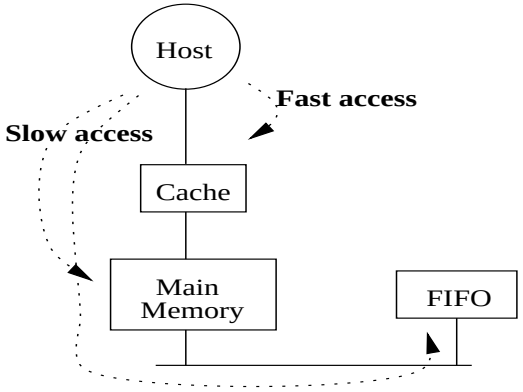
Diagram for protocol context of intelligent HNI



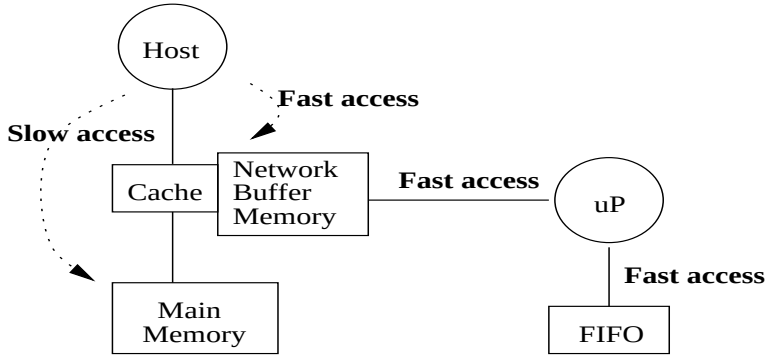
Formal Performance Metrics: 1



(a)



(b)



(c)

Formal Performance Metrics: 2

- $T_{\mu p}$: Clock cycle time of CPU data bus in nanoseconds. For Alpha 21164, it is $33MHz$ or $30.3nsec$ for 128 bits.
- $T_{\mu c}$: Clock cycle time of adaptor control processor. MC88110 has 25 nsec data bus clock cycle time for 64 bits.
- $L^f_{\mu p}$: The number of clock cycles per load from cache to host CPU, or from network buffer to host CPU.
- $L^f_{\mu c}$: The number of clock cycles per load from network buffer to network-adaptor control processor. It takes 3-1-1-1 data transfer clock cycles.
- $L^s_{\mu p}$: The number of clock cycles per load from main memory to host CPU. For typical EDO (Expanded Data Output) dynamic RAMs, 7-2-2-2 data transfer clock cycles in a pipelined-burst mode.
- $L^s_{\mu c}$: The number of clock cycles per load through bus. It is supposed

to have the same value with $L^s_{\mu p}$.

- $S^f_{\mu p}$: The number of clock cycles per store to the network buffer, or to the cache from the host CPU.
- $S^f_{\mu c}$: The number of clock cycles per store to one cell FIFO in network adaptor.
- $S^s_{\mu c}$: The number of clock cycles per store to main memory from host CPU or to cell FIFO from main memory in conventional architecture.

Transmitting Performance of Intelligent HNI

- For 48 Bytes data:
- 3 slow loads (M_H - 128 bits/load) in 7-2-2 μP cycle
- 3 fast stores (128 bits/store) in 5 clock cycles (3-1-1) to M_N
- 6 fast loads (64 bits) in 8 (3-1-1-1-1-1) clock cycles in pipelined burst
- 7 fast stores (64 bits) in 9 (8 payloads and 1 header) clock cycles
- 5 arithmetic and 1 branch instructions

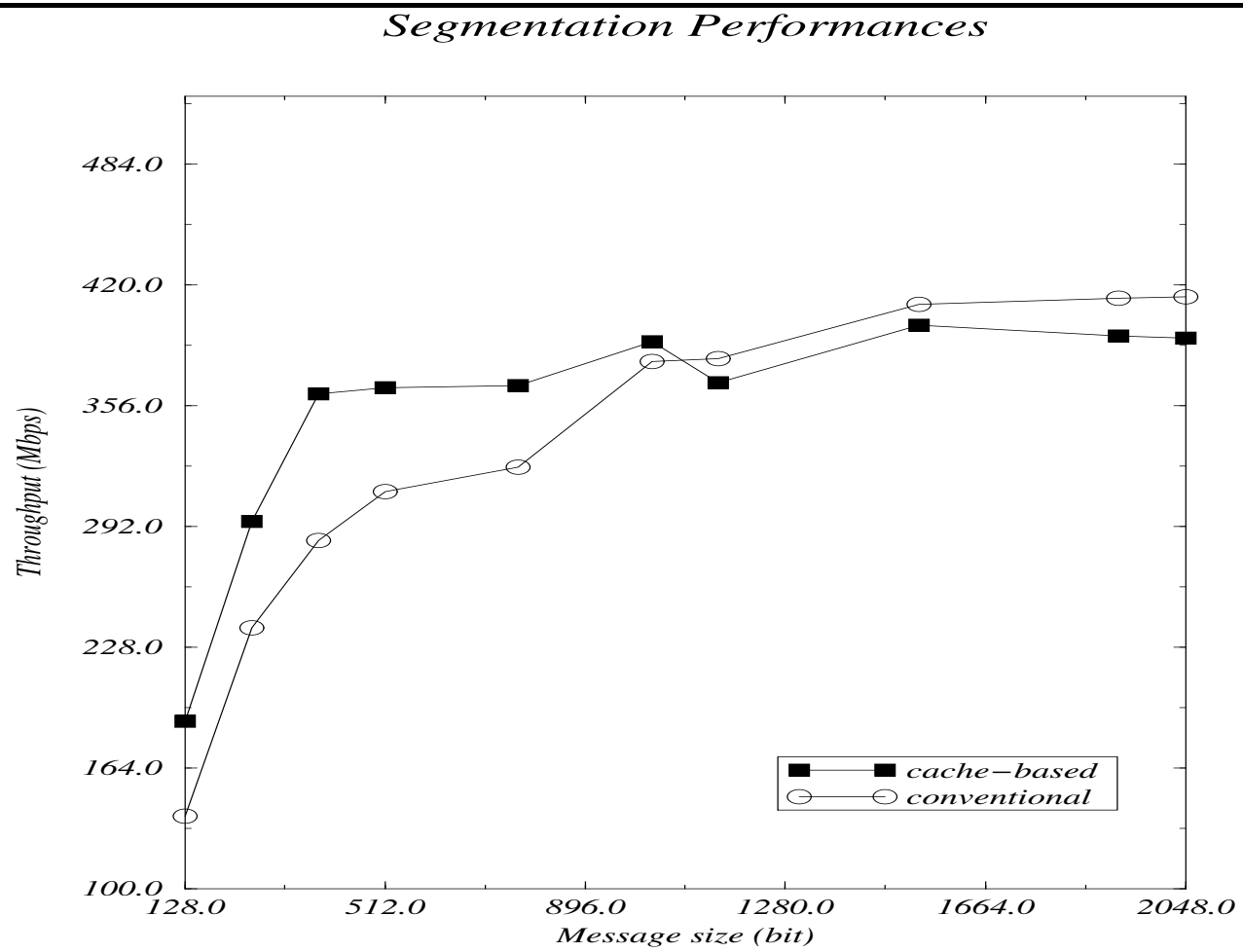
$$
\begin{aligned}
Throughput &= \frac{48\ byte/cell \times 8\ bit/byte}{(3L^s_{\mu p} + 3S^f_{\mu P})T_{\mu P} + (6L^f_{\mu C} + 7S^f_{\mu C} + 6\ computation)T_{\mu C}} \\
&= \frac{384}{(11+5)(30.3 \times 10^{-9}) + (8+9+6)(25 \times 10^{-9})} \\
&= 362.33 Mbps
\end{aligned}
$$

Transmitting Performance of Conventional OSIRIS HNI

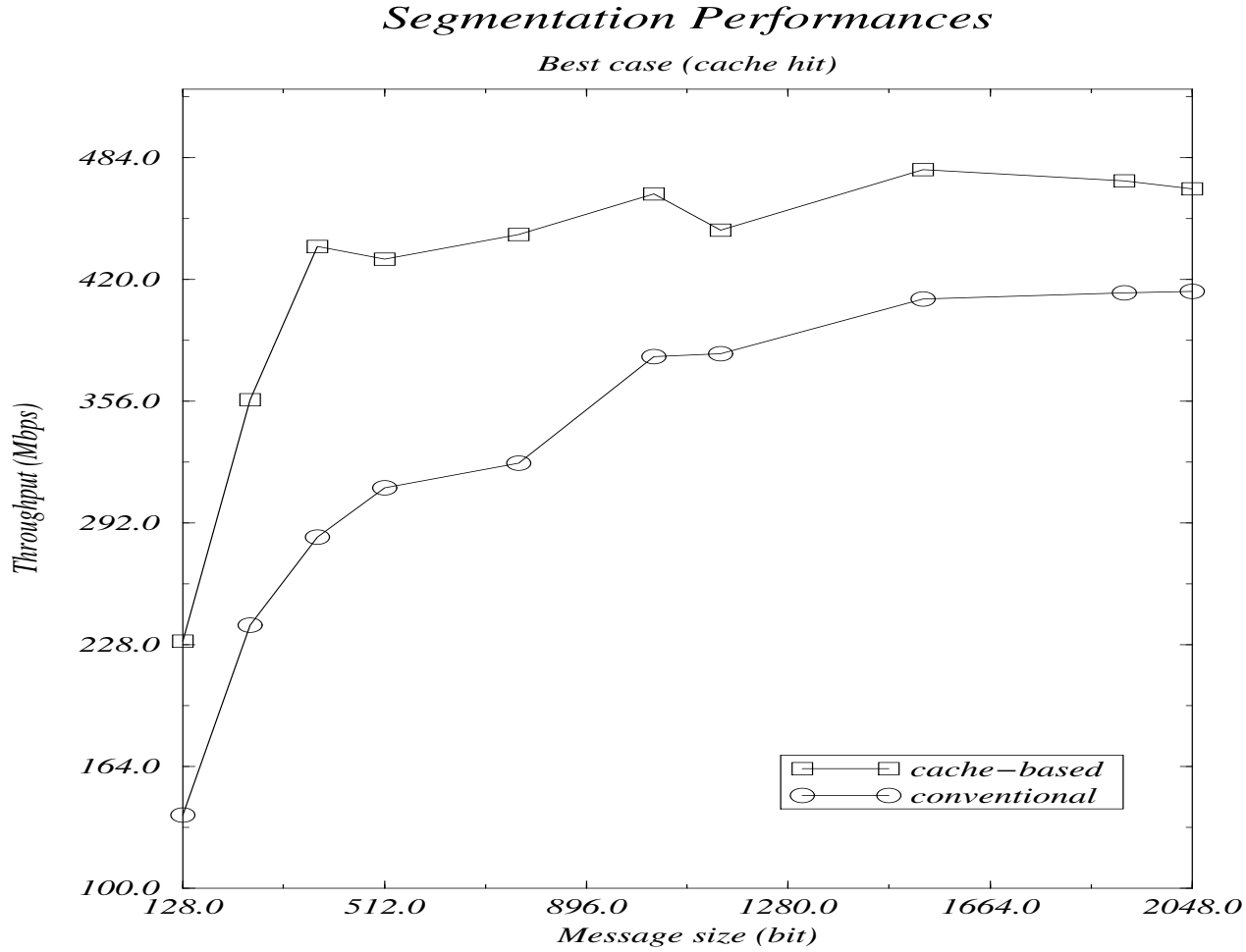
- P_N loads control information first, then loads and stores directly cell payload into cell FIFO
- 1 slow load and store for control information
- 6 slow loads for data transfer in 7-2-2-2-7-2 clock cycles with 64-bit wide
- 7 fast stores - 6 for data and 1 for header
- 5 arithmetic and 1 branch instructions

$$
\begin{aligned}
Throughput &= \frac{48\ byte/cell \times 8\ bit/byte}{(L^s_{\mu p} + S^s_{\mu p})T_{\mu p} + (6L^s_{\mu c} + 7S^f_{\mu C} + 6computation)T_{\mu C}} \\
&= \frac{384}{(7+7)(30.3 \times 10^{-9}) + (22+9+6)(25 \times 10^{-9})} \\
&= 284.61 Mbps
\end{aligned}
$$

Segmentation Performance of intelligent HNI: worst case



Segmentation Performance of intelligent HNI: best case



Receiving Performance of intelligent HNI

- In intelligent HNI for 48 Bytes:
- 6 fast loads (3-1-1-1-1-1)
- 6 fast stores (3-1-1-1-1-1)
- 8 arithmetic and 4 branch instructions

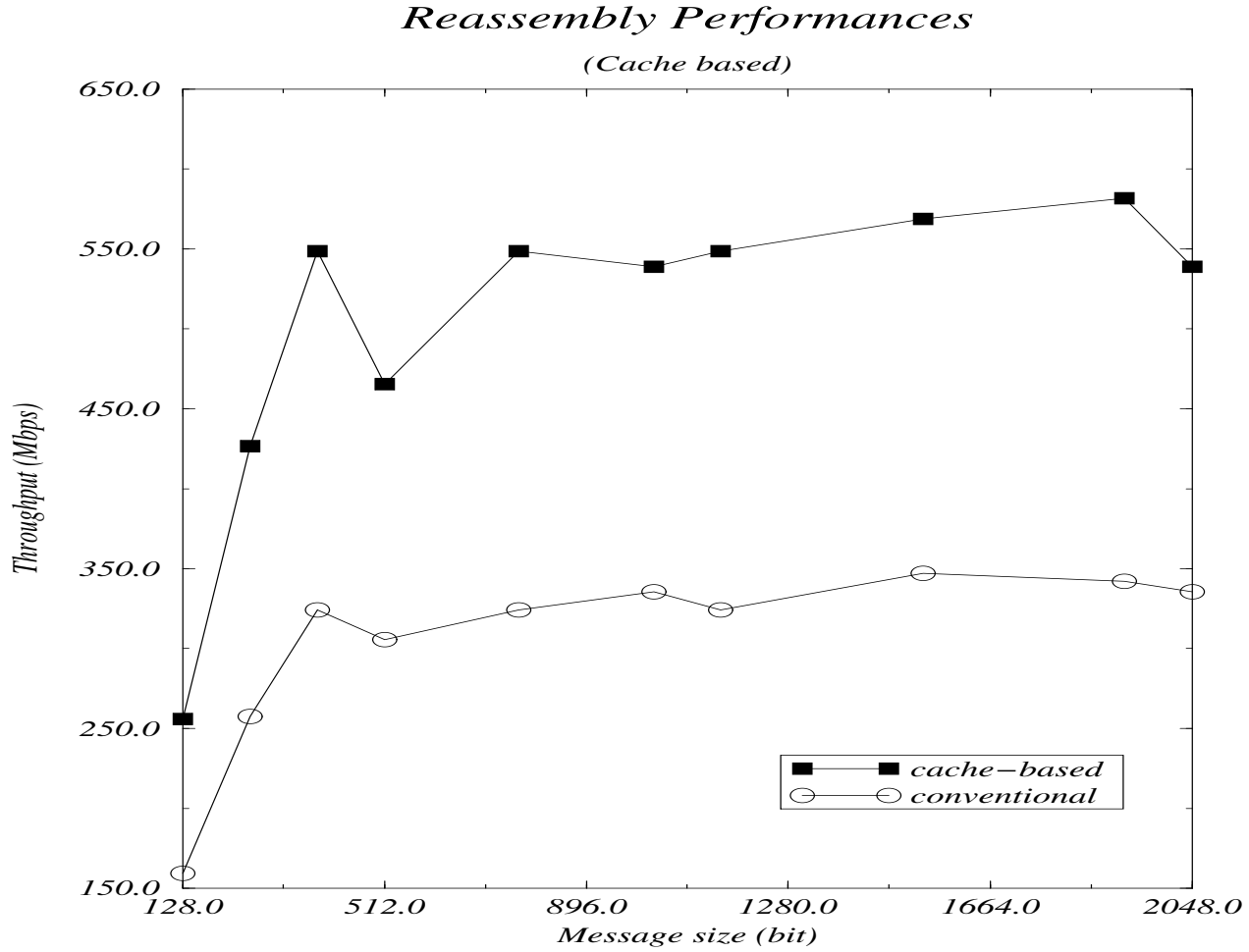
$$
\begin{aligned}
Throughput &= \frac{48 Bytes}{(6L^f_{\mu c} + 6S^f_{\mu c} + 12\ computation)T_{uC}} \\
&= \frac{384}{(8+8+12)25 \times 10^{-9}} \\
&= 548.57\ Mbps
\end{aligned}
$$

Receiving Performance of OSIRIS HNI

- In OSIRIS for 48 Bytes:
- 6 fast loads and stores to M_N
- 8 arithmetic and 4 branch instructions
- 6 slow loads of DMA data transfers (64 bit/load)
- 3 fast stores in main memory (128 bits/store)
- 6 fast store to M_N can be ignored by the dual-ported memory feature

$$
\begin{aligned}
Throughput &= \frac{48 Bytes}{3S^f_{\mu p} \times T_{\mu p} + (6L^s_{\mu c} + 6S^f_{\mu c} + 12\ comp)T_{uC}} \\
&= \frac{384}{5 \times 30.3 \times 10^{-9} + (22 + 8 + 12)25 \times 10^{-9}} \\
&= 319.60\ Mbps
\end{aligned}
$$

Reassembly Performance of intelligent HNI



Answer to Research Questions

- Any problems on HNIs?
  - Complicated and slow transport protocol implementations
  - Host interfacing between host and HNI board,
  - High overhead caused by OS functions and context switchings
  - => A comprehensive taxonomy
    - For better analyzing and understanding of problems on HNIs
    - For designing of improved HNI systems
- Any current taxonomy with comprehensive features?
  - Only the argument about superiority of the proposed design
  - Only one category $- >$ a narrow breadth of comparison
  - Lack of predictive, comprehensive powers
  - => A comprehensive taxonomy for HNIs with two context layers

- How does our proposed taxonomy help to design HNI?
  - By predictive features, comparison of the design with other systems
  - Optimized components in architectural and protocol-implementation contexts
  - Capability to evaluate performance of the design

- Therefore, the problems of host-network interfaces can be solved by our taxonomy.

Future Works

- Extention of protocol-implementation context with API
- Bad API implementations:
  - Even with low-latency HNI and high-computing-powered host
  - Overheads from OS
  - Whole network environment to slow down
- One example - JaguarVIA
  - a Java API to improve performance of communication and I/O methods in Java programming.

Skillicorn's Taxonomy

Using his taxonomic scheme, Skillicorn established a single category of 28 classes:

class	nIP	nDP	IP-DP	IP-IM	DP-DM	DP-DP	Name
3	0	n	-	-	$n-n$	$n \times n$	loosely coupled dataflow
4	0	n	-	-	$n \times n$	-	tightly coupled dataflow
6	1	n	$1-1$	$1-1$	$1-1$	-	von Neumann uniprocessor
8	1	n	$1-n$	$1-1$	$n-n$	$n \times n$	type 1 array processor
9	1	n	$1-n$	$1-1$	$n \times n$	-	type 2 array processor
13	n	n	$n-n$	$n-n$	$n-n$	-	separate von Neumann uniprocessors
14	n	n	$n-n$	$n-n$	$n-n$	$n \times n$	loosely coupled von Neumann
15	n	n	$n-n$	$n-n$	$n-n$	-	tightly coupled von Neumann

The Processing Capability of HNI's

- E. Cooper and *et al*,
- Due to the difference in performance and implementation complexity,
  - Protocol processor on board
  - Memory accessing way (user and kernel spaces in main memory)
  - DMA capability
- Simple HNIs
  - Host as protocol processor
  - Memory accessing on both of user and kernel spaces
- Intelligent HNIs
  - On-board protocol processor
  - Memory accessing only on user space
  - DMA capability

Memory Access

According to the address-mapping spaces and data accessing mechanisms, David Banks and *et al* classifies HNIs as mapping:

- In memory space
- In I/O space
  - Programmed I/O
  - Direct Memory Access (DMA)
  - Burst mode transfer by memory controller

Host Access

D. Henry and C. Joerg proposes four categories:

- OS-level DMA based
- User-level memory mapped
- User-level register mapped
- Hardwired

Architectural Context

- The characteristics are from the logical structures and connections of hosts, host-network interfaces, and a network.
- Principal components are:
  - Host CPU
  - Main memory
  - Cache memory
  - Protocol processor or controller
  - Network buffer
  - Switch
- By PMS (processor-memory-switch) notations:
  - The relations between processor, memory, and I/O devices.
  - A system as an interconnected set of components or individual devices, associated with a set of operations.

continue...

- Host CPU unit is to execute instructions of the user application which may require a communication to, if required, the protocol processor related to the communication, and to transform data usually in ways that correspond to basic arithmetic operations.
- Main memory unit is an intelligent storage device that passes data to and from the host CPU.
- Cache unit consists of a small fast memory that acts as a buffer between the main memory and the host CPU.
- Network buffer unit is a staging and speed-matching area for data in transit between the host and the network. It consists of a network buffer memory and network FIFOs. Network buffer memory is the storage for transport-layer data (so called message) and information related to the control and management parameters, while network FIFOs are two set of registers for sending and receiving. It also

provides the protocol processor with contention-free memory access to the packet data.

- Protocol processor unit manages packet processing and various bookkeeping functions associated the protocol.
- Switch unit provides connectivity between other functional units in one way of programmed I/O, DMA (Direct Memory Access), burst transfer by memory controller, or register accessing.

Protocol-Implementation Context

- Packetization/Depacketization
- Data copying
- Flow control
- Error handling
- Routing/switching

Packetization/Depacketization

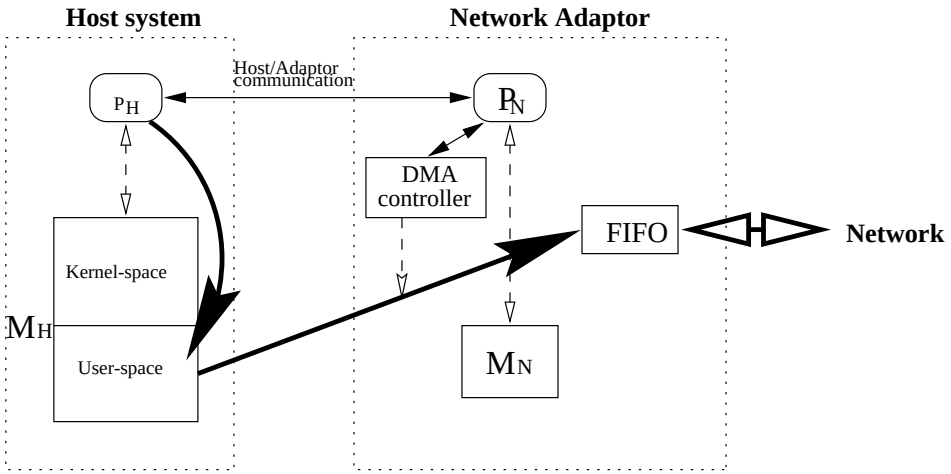
- Data is bundled and transmitted as packets of bit streams with some overhead bits to avoid network errors
- A message larger than the packet data size
- OS or a protocol processing unit fragments the message into a series of packet data
- Reassembles the packets into the message in the receiving process.
- Classified by who processes and by packet length as following:
  1. A variable-sized data unit that is packetized by the host.
  2. A variable-sized data unit that is packetized by the HNI.
  3. A fixed-sized data unit that is packetized by the host.
  4. A fixed-sized data unit that is packetized by the HNI.

Data Copying

- Memory space accessing - a user space and a kernel space
- Data Path - Programmed I/O, DMA, burst transfer by memory controller
- Classification:
  1. 0-copy from user space on main memory to network.
  2. 1-copy from user space of main memory to network buffer memory, then to network.
  3. 1-copy from user space of main memory to kernel space of main memory.
  4. 2-copy from user space to kernel space of main memory then to network buffer memory.

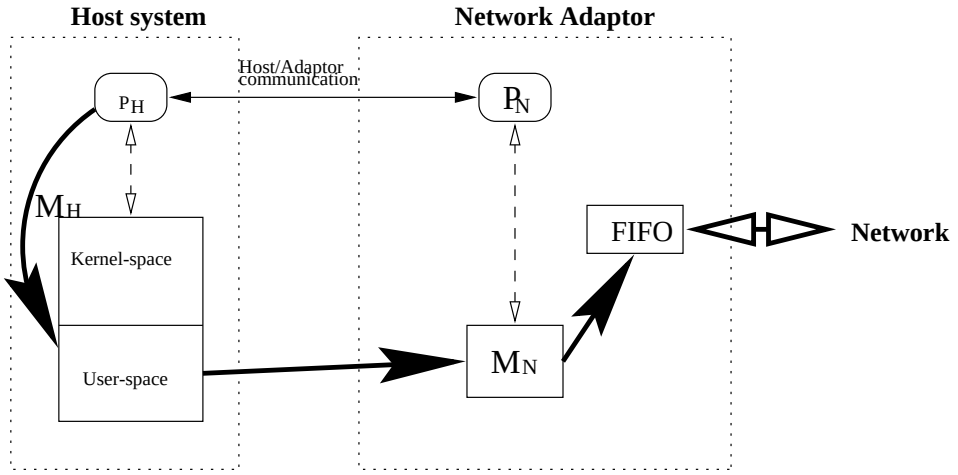
0-copy from user space on main memory to network

- The host processor informs data address and length to the network protocol processor.
- The network protocol processor initiates DMA controller to transfer data from a user space of main memory to network.



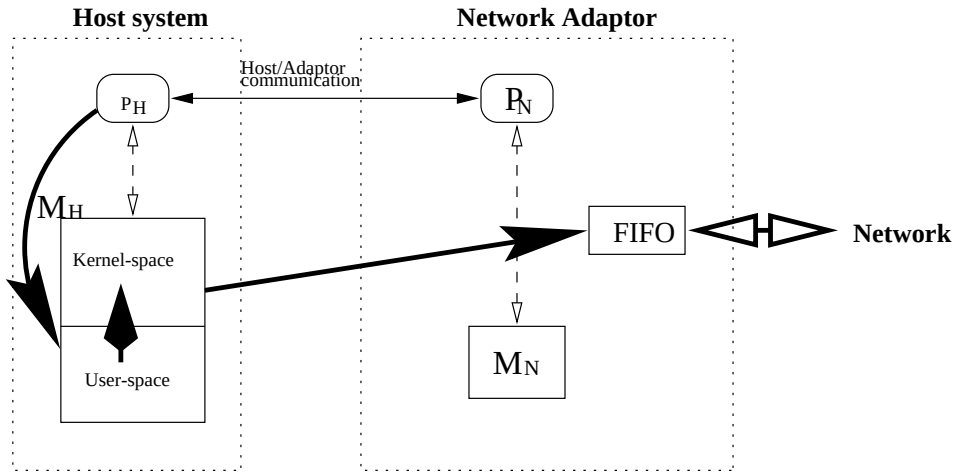
1-copy from user space of main memory to network buffer memory, then to network

After data is copied to network buffer memory, HNI provides some other protocol processing such as CRC and composing of header fields.



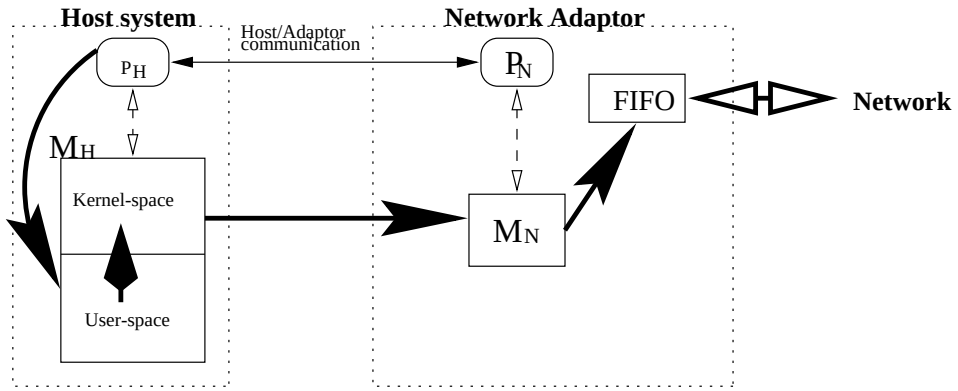
1-copy from user space of main memory to kernel space of main memory

- To process protocol, data is copied to kernel space.
- Good security and protection of data between users.



2-copy from user space to kernel space of main memory then to network buffer memory

Transport protocol processing on kernel space except CRC checking which is supported on host-network interface.



Flow Control

- The flow of offered-data load just enough to achieve a throughput that is very close to that of the resources' capacity
- Receiver flow control schemes (RFC):
  1. A receiving window by acknowledgements and sequence numbers
  2. Software-based for counting of the number of received packets
  3. Hardware-based for counting of the number of received packets
  4. Hardware-based measurement of the inter-packet arrival time
  5. Accepting/discarding packets based on buffer availability
- State-reporting schemes (SR):
  1. Window-based acknowledgements
  2. The current inter-packet arrival time or burst size of packets
  3. Start/stop control signal

- Transmitter flow control (TFC):
  1. Sending packets based on current window size
  2. Sending packets based on the current acceptable transmission rate
  3. Sending packets based on start/stop control signal

Error Handling

- Error detection, and correction when the network is not reliable.
- Error-detection functions (D):
  1. Cyclic Redundancy Check (CRC)
     - Type: CRC-8, CRC-10, CRC-12, CRC-16, CRC-32, CRC-CCITT.
     - Domain: A control part, A data part, or A control + data part.
  2. Length indicator of data.
  3. Sequence number.
  4. Multiplexing identifier (MID).
- Error-correction functions (C):
  1. Forward Error Correction (FEC).
  2. Backward Error Correction/Automatic Repeat Request (ARQ).
  3. FEC + ARQ.

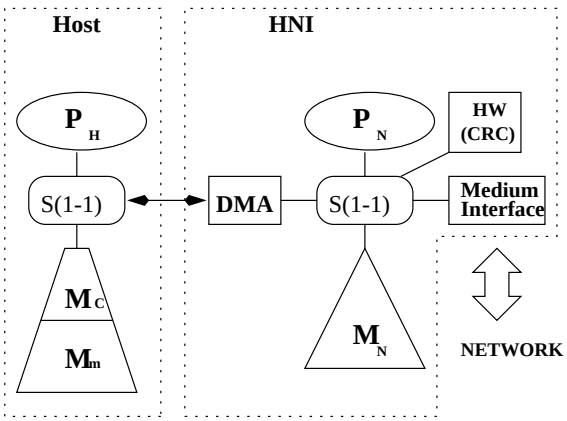
Routing/Switching

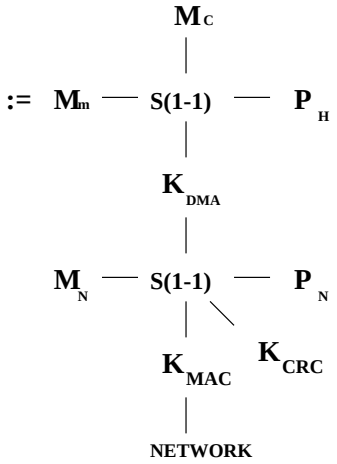
- When an application process sets up a connection to another
- Address generating (or address resolution) function
- Source address, destination address(es) and intermediate-node addresses if needed.
- At least, port numbers, host numbers, network numbers, and frame numbers in TCP/IP suites
- Classification:
  1. Circuit switching
  2. Source routing/packet switching
  3. Internode routing/packet switching
  4. Virtual circuit switching (statistical switching)

Illustrative Example: AURORA

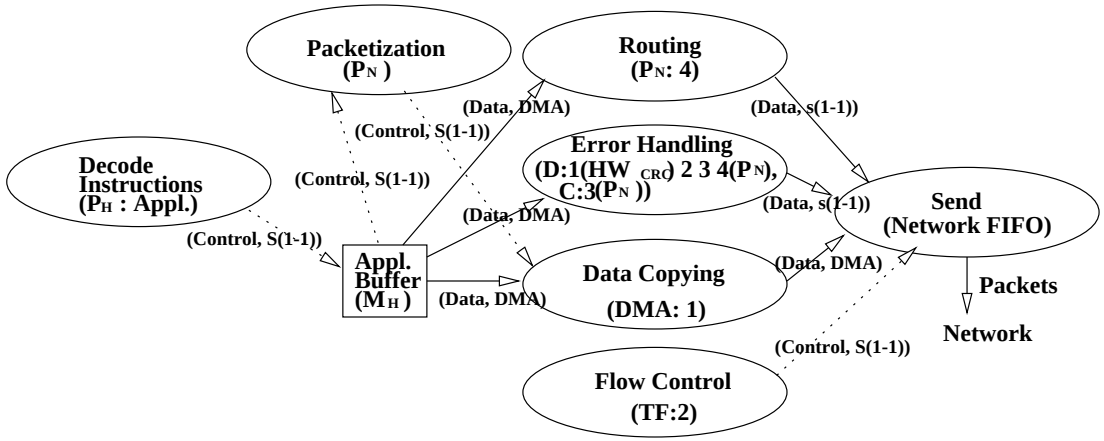
- AURORA - An experimental wide area network testbed
- Operation at near giga-bit per second bandwidths
- HNI converts data between the network format (ATM cells) and a format useful to the host (transport level message - TPDU)
- OSIRIS for TURBOchannel bus on DECstation 5000
- ORBIT for MICROchannel bus on the IBM RS/6000

Continue...

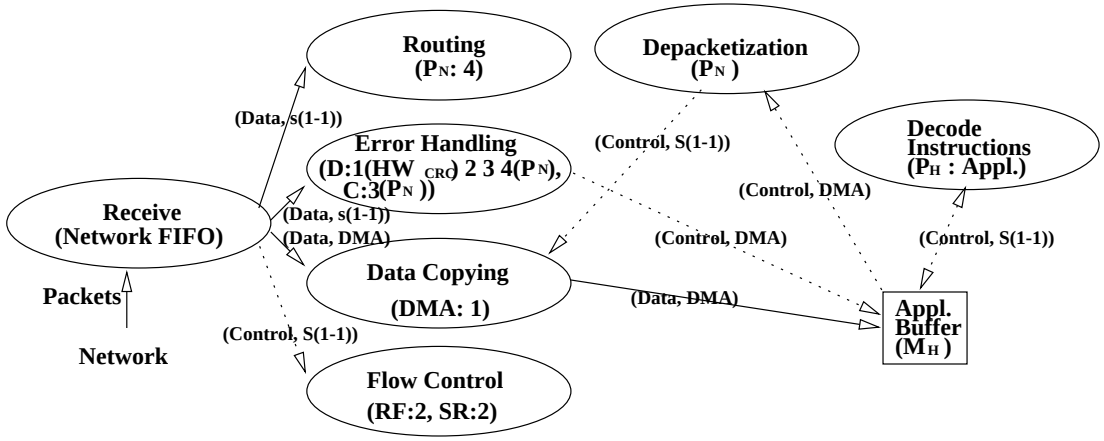

Host
HNI
P H
S(1-1)
M C
M m
DMA
P N
S(1-1)
M N
HW (CRC)
Medium Interface
NETWORK


M C
:= M m
S(1-1)
P H
K DMA
M N
S(1-1)
P N
K MAC
K CRC
NETWORK

Continue...



(a) Transmission



(b) Reception

Simple HNI Design

- ATM network with AAL-5
- A cost-effective system: minimal hardware support
- Easy adjustments to the changes of environment
- low speed network and multiple protocol support environment

Design Requirements

In architectural context:

- Conection component - Comparative bandwidth to that of the system bus
- Fast processing of the cell header error checking

In protocol context:

- Routing - only the source's and destination's addresses
- Data copying - main memory to network FIFO
- Error handling - in ATM network, CRC checking, sequence numbering, MID, data length indicator and FEC + ARQ
- Flow control - simple implementation.
- Packetization - fixed sized payload

Taxonomy-based Design Approach

- In the architectural context,
  - Connection between main memory and network FIFO by direct memory access: DMA for $M_H - FIFO$.
- In protocol implementation context,
  - Virtual circuit switching.
  - Zero data copying between user space of main memory and network FIFO.
  - CRC-8 by hardware for header error check, CRC-10 by software for data error check, sequence number, MID, and length indicator of data.
  - Window-based and rate-based flow control

Diagram for Archtiectural context of simple HNI

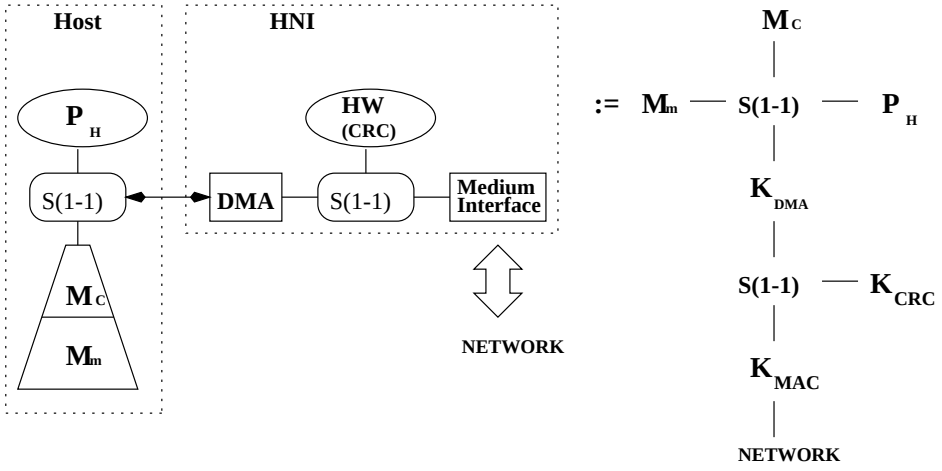
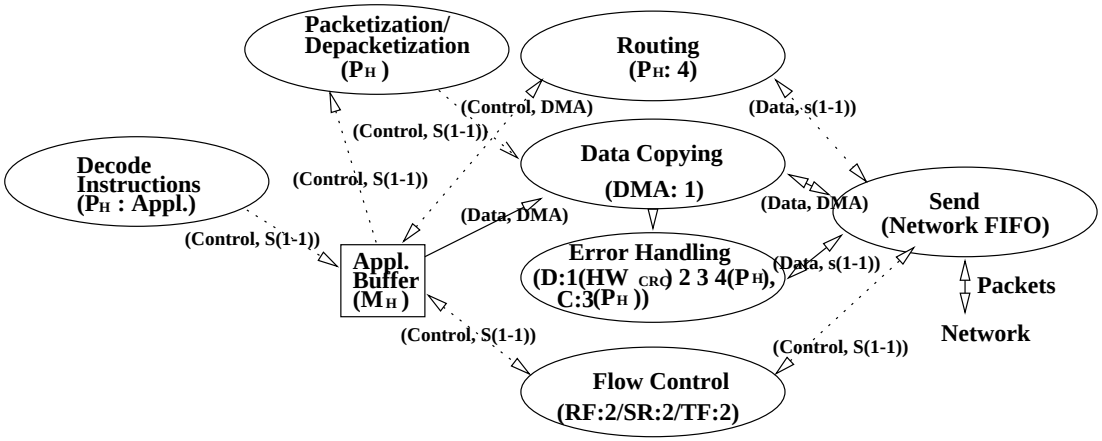


Diagram for protocol context of simple HNI



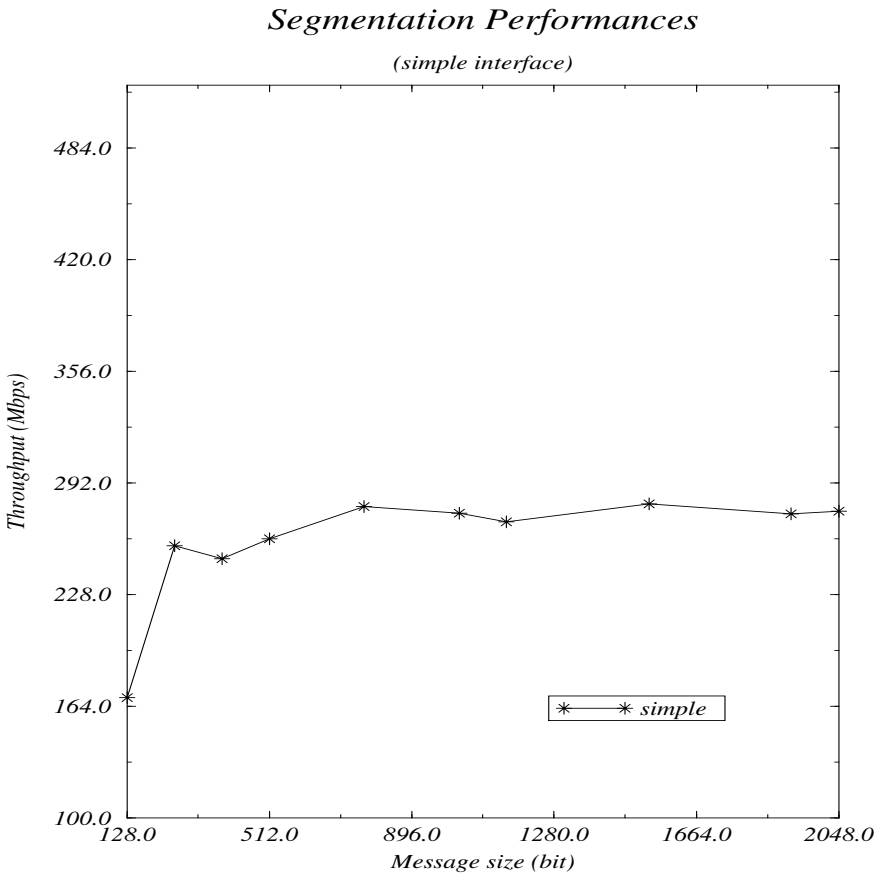
Transmitting Performance of simple HNI

To send 1 payload data (384 bits),
6 slow loads and 6 slow stores through the 64-bit PCI system bus.

For the header information,
1 slow store (since less than 1 fast load for the header information and CRC field), 5 arithmatic and 1 branch instructions.

$$
\begin{aligned}
Throughput &= \frac{48\ byte/cell \times 8\ bit/byte}{(6L^s_{\mu p} + 7S^s_{\mu p} + 6computation)T_{\mu p}} \\
&= \frac{384}{(22 + 22 + 6)(30.3 \times 10^{-9})} \\
&= 248.50 Mbps
\end{aligned}
$$

Segmentation Performance of simple HNI

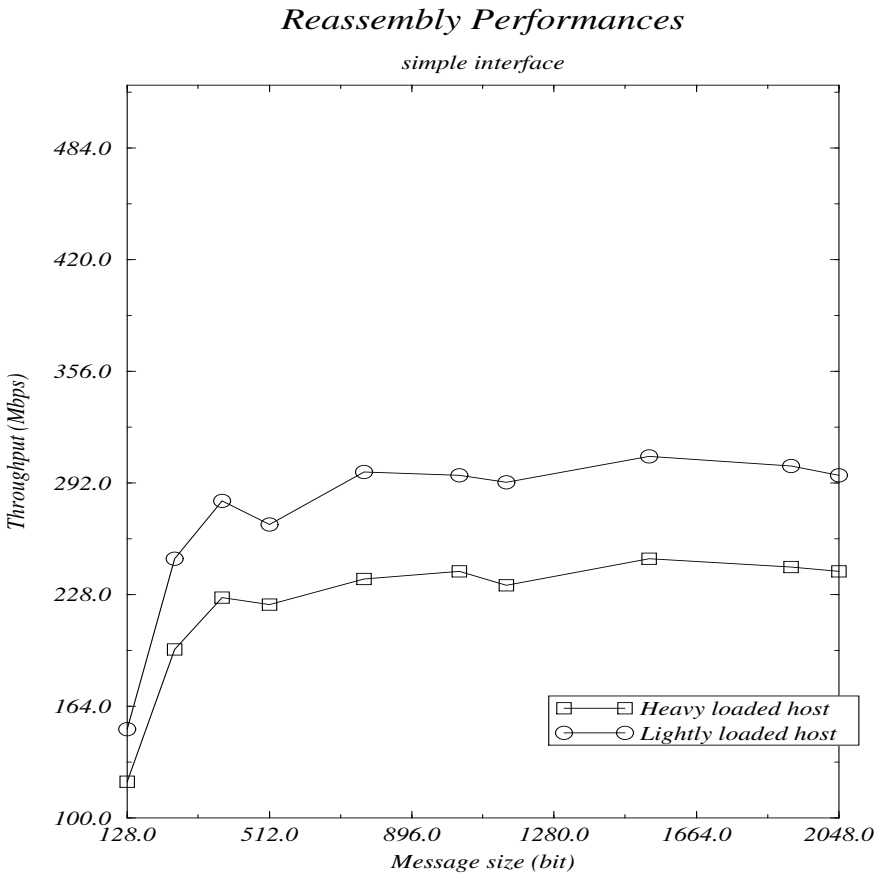


Receiving Performance of simple HNI

Reassembly by a lightly loaded host for an ATM cell: 6 loads (7-2-2-2-7-2) by 64 bit DMA B/W; 3 (7-2-2) stores by 128 bit memory B/W.

$$\begin{aligned} Throughput &= \frac{48 Bytes}{(6L^s_{\mu p} + 3S^s_{\mu p} + 12\ computation)T_{up}} \\ &= \frac{384}{(22 + 11 + 12)30.3 \times 10^{-9}} \\ &= 281.63\ Mbps \end{aligned}$$

In heavy loaded host: additional 3 slow loads from main memory.

$$\begin{aligned} Throughput &= \frac{48 Bytes}{(6L^s_{\mu p} + 3L^s_{\mu p} + 3S^s_{\mu p} + 12\ computation)T_{up}} \\ &= \frac{384}{(22 + 11 + 11 + 12)30.3 \times 10^{-9}} \\ &= 226.31\ Mbps \end{aligned}$$

Reassembly Performance of simple HNI