University of Crete
School of Sciences & Engineering
Computer Science Department

Master Thesis

by

Michael Papamichael

Network Interface Architecture and Prototyping for Chip and Cluster Multiprocessors

Supervisor

Prof. Manolis G.H. Katevenis

Heraklion, Crete, July, 2007

Presentation Outline

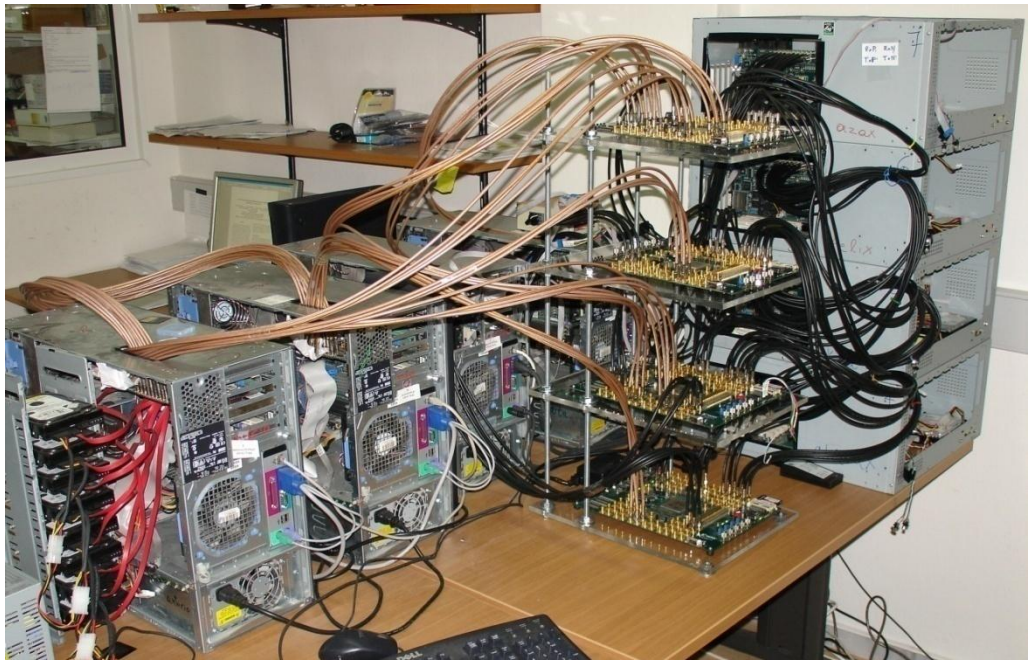
- **NI Queue Manager**
  - Introduction
  - Key Concepts
  - Architecture & Implementation
- **NI Design for CMPs**
  - NI Design Goals
  - NI Design Issues
  - Proposed NI Design
- **Conclusions & Future Work**

Presentation Outline

- **NI Queue Manager**
  - **Introduction**
  - Key Concepts
  - Architecture & Implementation
- **NI Design for CMPs**
  - NI Design Goals
  - NI Design Issues
  - Proposed NI Design
- **Conclusions & Future Work**

Introduction

- **FPGA-based Prototyping Platform**
  - PCI-X RDMA-capable NIC in cluster environment
  - Buffered crossbar switch
- **Goals**
  - Confirm buffered crossbar behavior
  - Interprocessor communication research

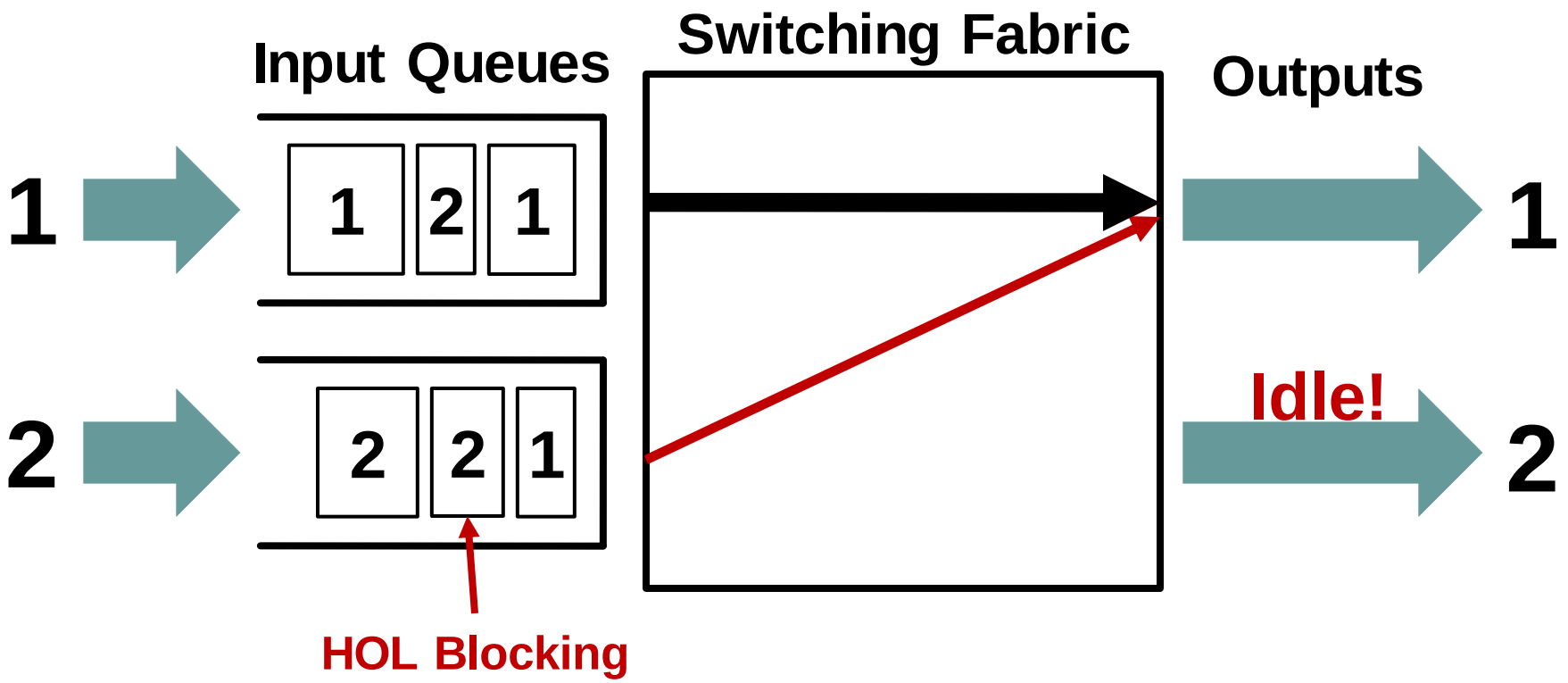


Presentation Outline

- **NI Queue Manager**
  - Introduction
  - **Key Concepts**
  - Architecture & Implementation
- **NI Design for CMPs**
  - NI Design Goals
  - NI Design Issues
  - Proposed NI Design
- **Conclusions & Future Work**

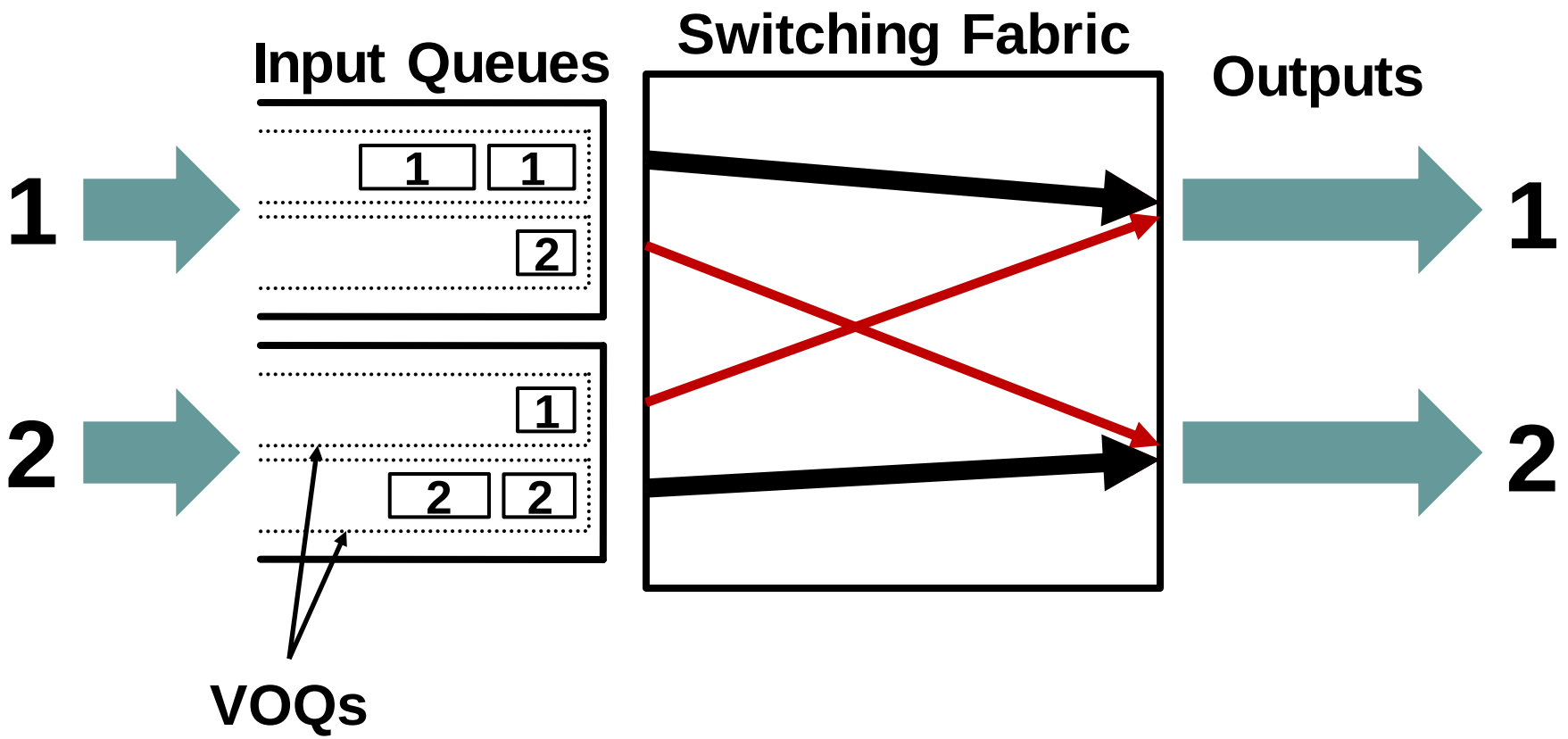
Head-Of-Line (HOL) Blocking

- HOL Blocking reduces switch throughput



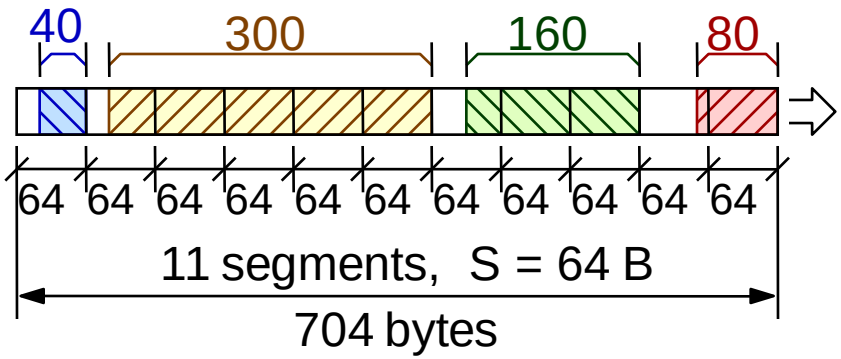
Virtual Output Queues (VOQs)

- VOQs eliminate HOL Blocking

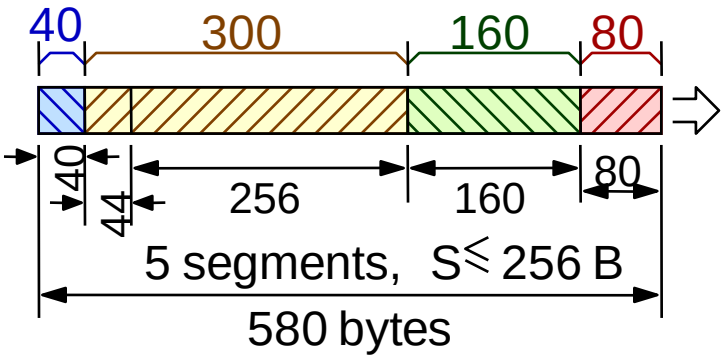


Traffic Segmentation Schemes

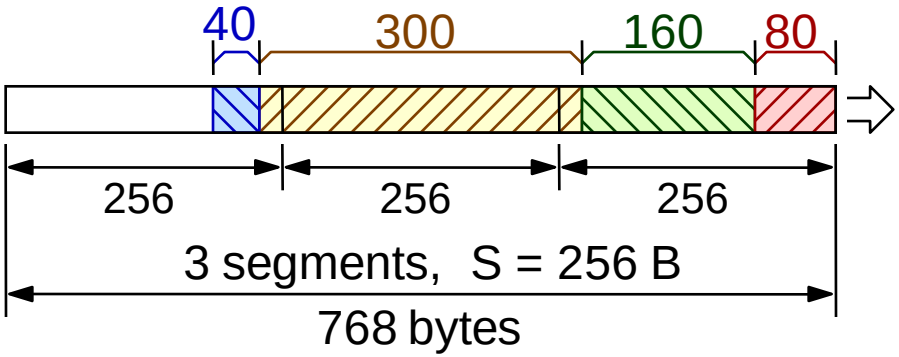
- **Traffic segmented to optimize switching**
- **Variable-Size MultiPacket (VSMP) Segmentation well suited to buffered crossbar**



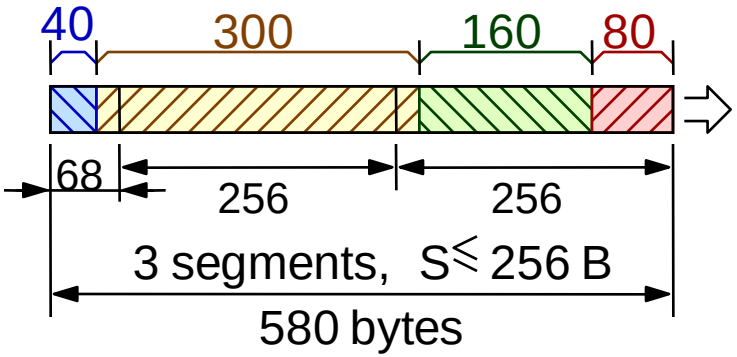
Fixed-size Unipacket segments



Variable-size Unipacket seg.



Fixed-size Multipacket segments



Variable-size Multipacket seg.

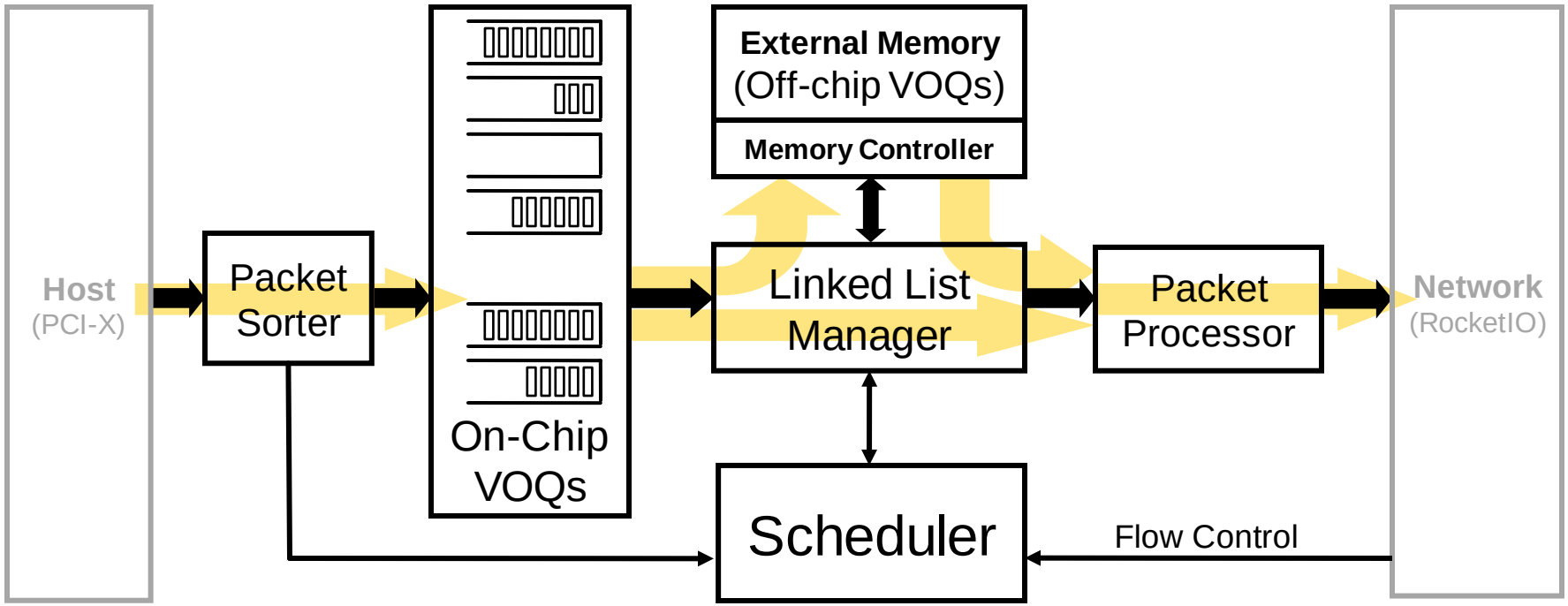
Presentation Outline

- **NI Queue Manager**
  - Introduction
  - Key Concepts
  - **Architecture & Implementation**
- **NI Design for CMPs**
  - NI Design Goals
  - NI Design Issues
  - Proposed NI Design
- **Conclusions & Future Work**

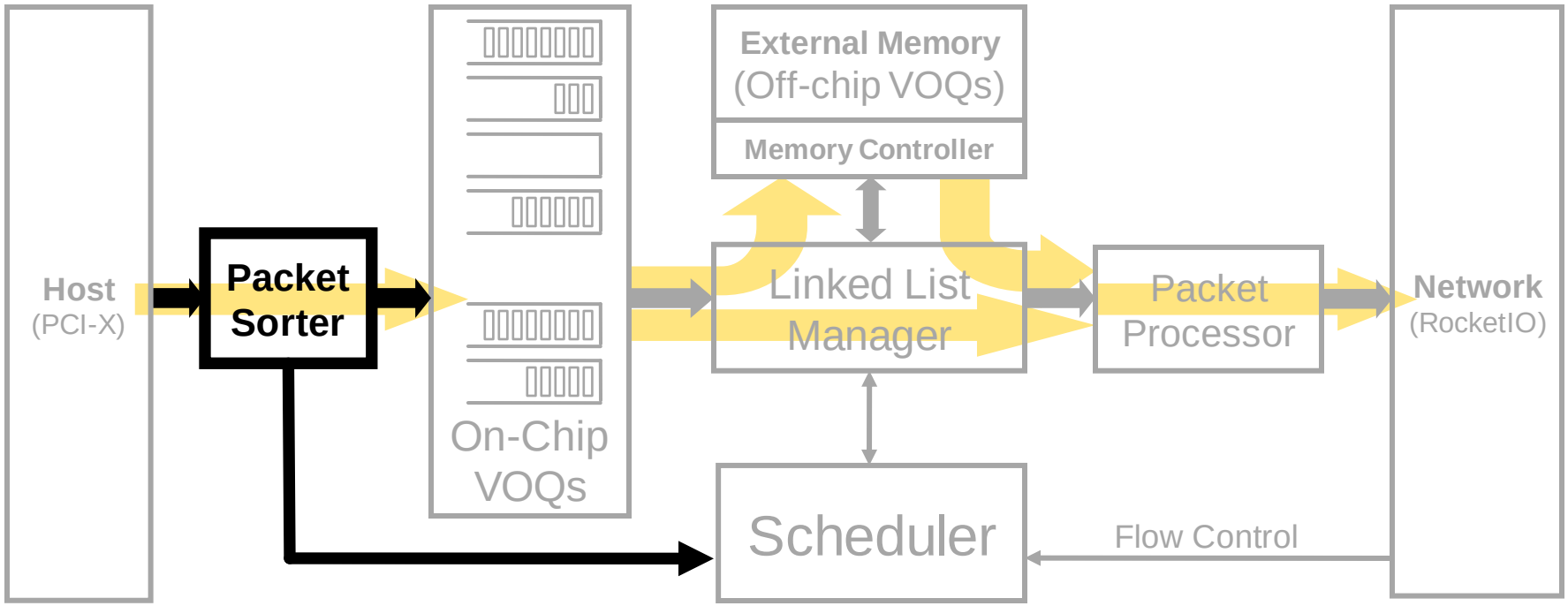
Overview

- **Virtual Output Queues (VOQs)**
- **VOQ migration to external memory**
  - Hardware-managed linked lists
- **VSMP Segmentation**
- **Scheduling**
- **Flow Control**
- **3 major versions implemented**

Architecture



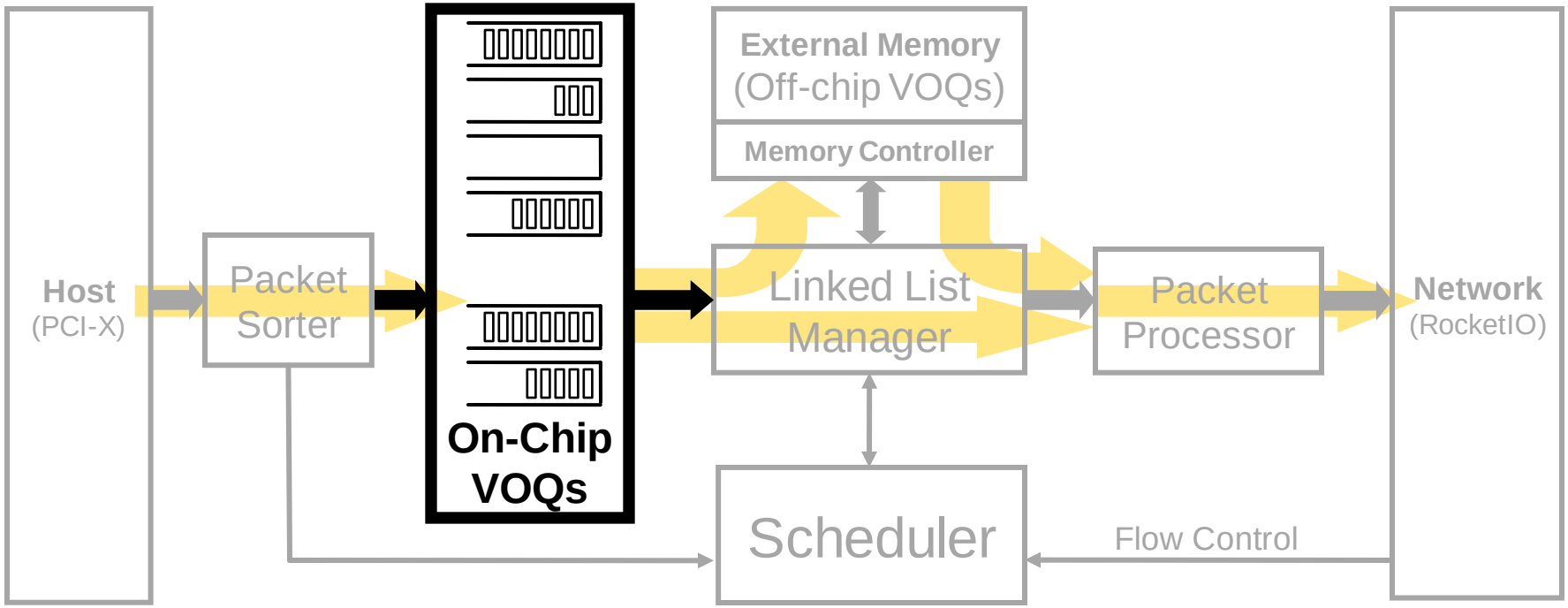
Packet Sorter



Packet Sorter

- **Sorts packets according to:**
  - destination
  - other criteria (e.g. priority)
- **Notifies Scheduler about incoming traffic**
- **Light-weight packet processing**
  - e.g. enforce maximum packet size
- **2 versions implemented**
  - with packet segmentation
  - without packet segmentation

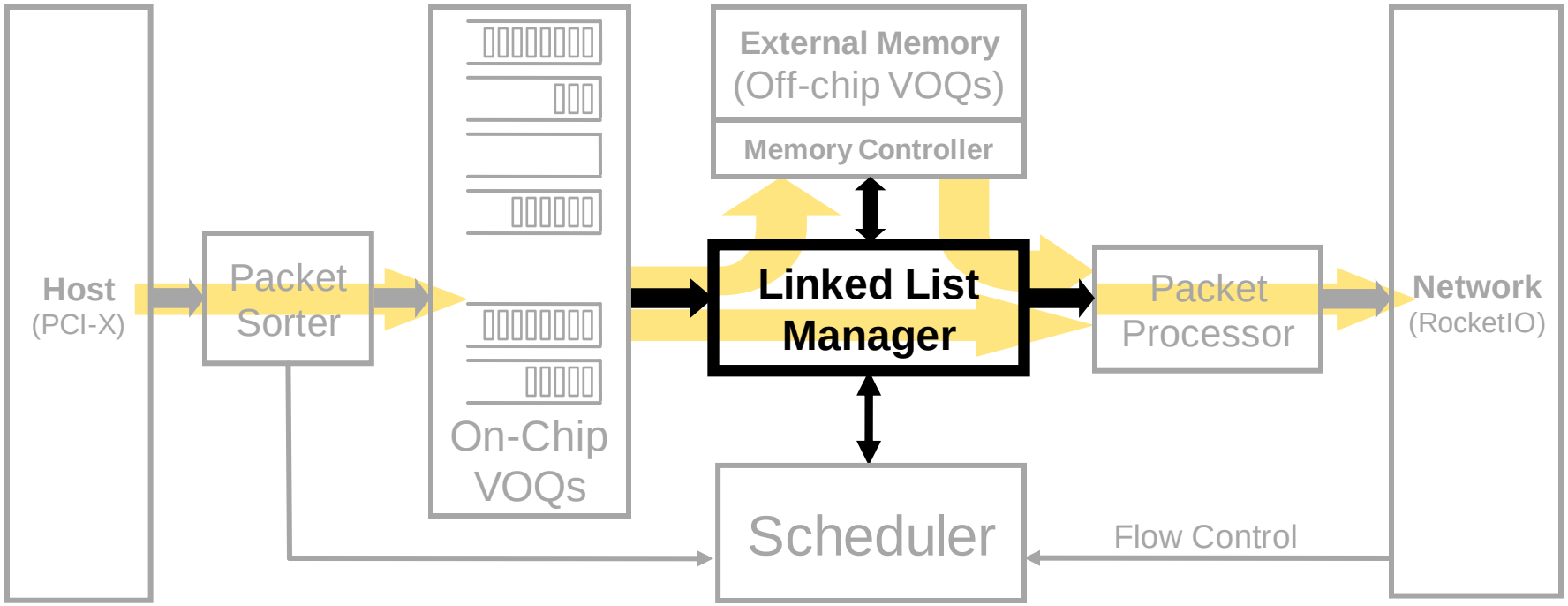
On-Chip VOQs


Host
(PCI-X)
Packet
Sorter
On-Chip
VOQs
External Memory
(Off-chip VOQs)
Memory Controller
Linked List
Manager
Packet
Processor
Network
(RocketIO)
Scheduler
Flow Control

On-Chip VOQs

- **Accumulates traffic in VOQs**
- **VOQs implemented as:**
  - Circular buffers in single statically partitioned on-chip memory
  - Xilinx FIFOs
- **2 versions implemented**
  - VOQs in BRAM
  - VOQs in Xilinx FIFOs

Linked List Manager

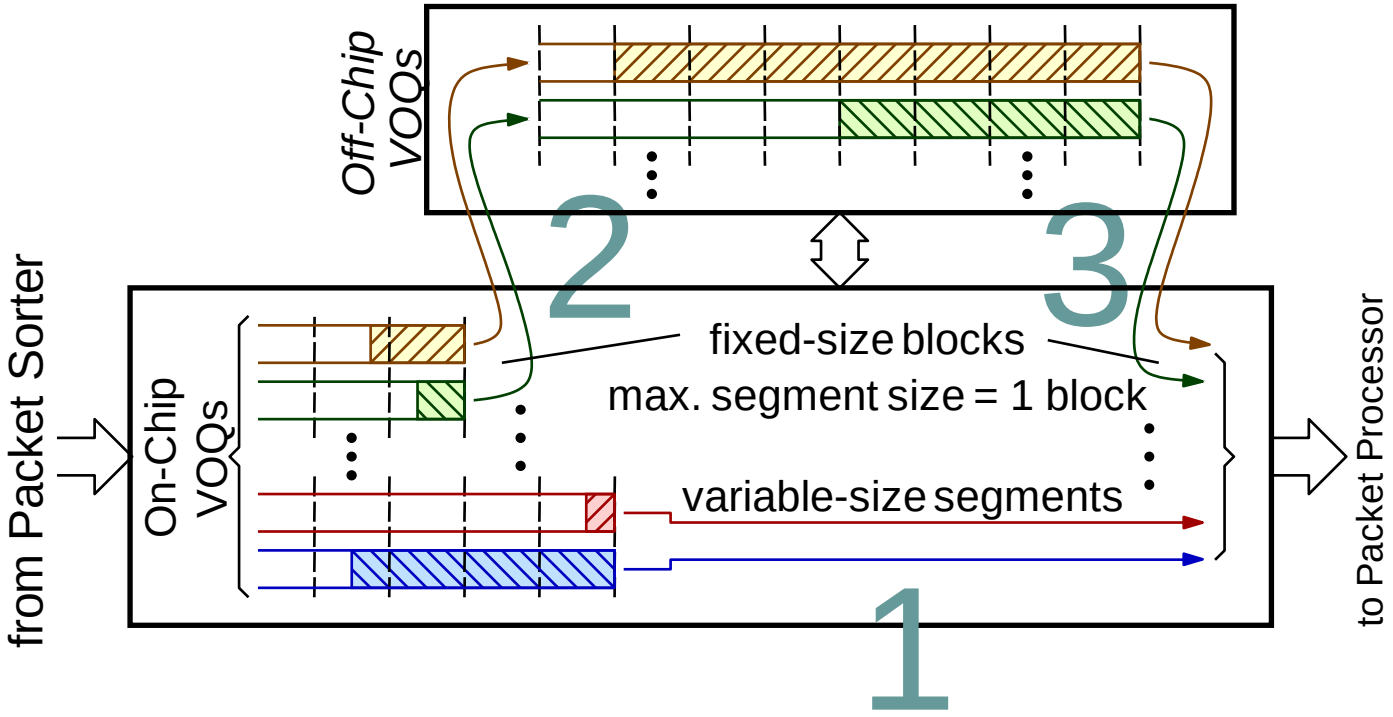


Linked List Manager

- **Performs Segment Transfers**
  - variable-size segments
  - fixed-size segments
- **Manages Linked Lists**
  - Head, Tail pointers in on-chip memory
  - Next Block pointers in DRAM (along data)
- **Optimization Techniques**
  - Free Block Preallocation
  - Free-List Bypass
- **FSM-based Implementation**

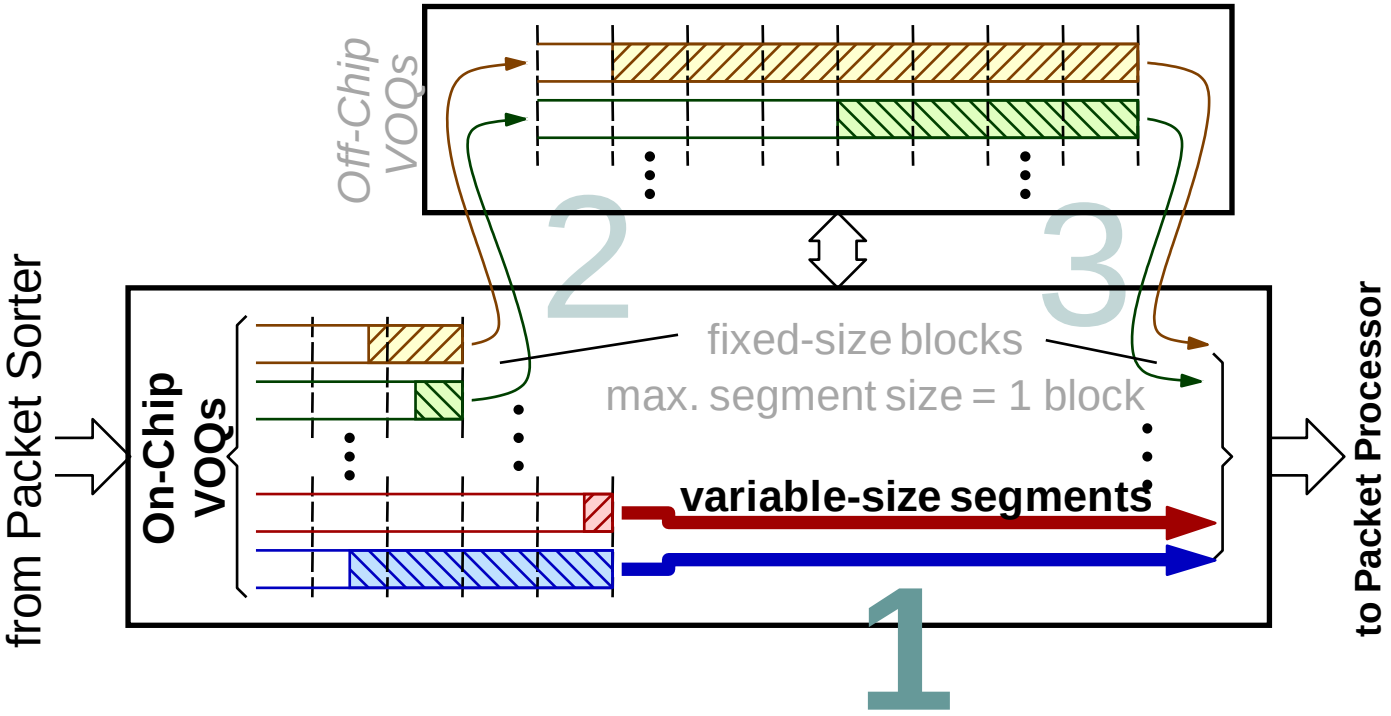
Segment Transfers

1. From On-Chip VOQs to Packet Processor
2. From On-Chip to Off-Chip VOQs
3. From Off-Chip VOQs to Packet Processor



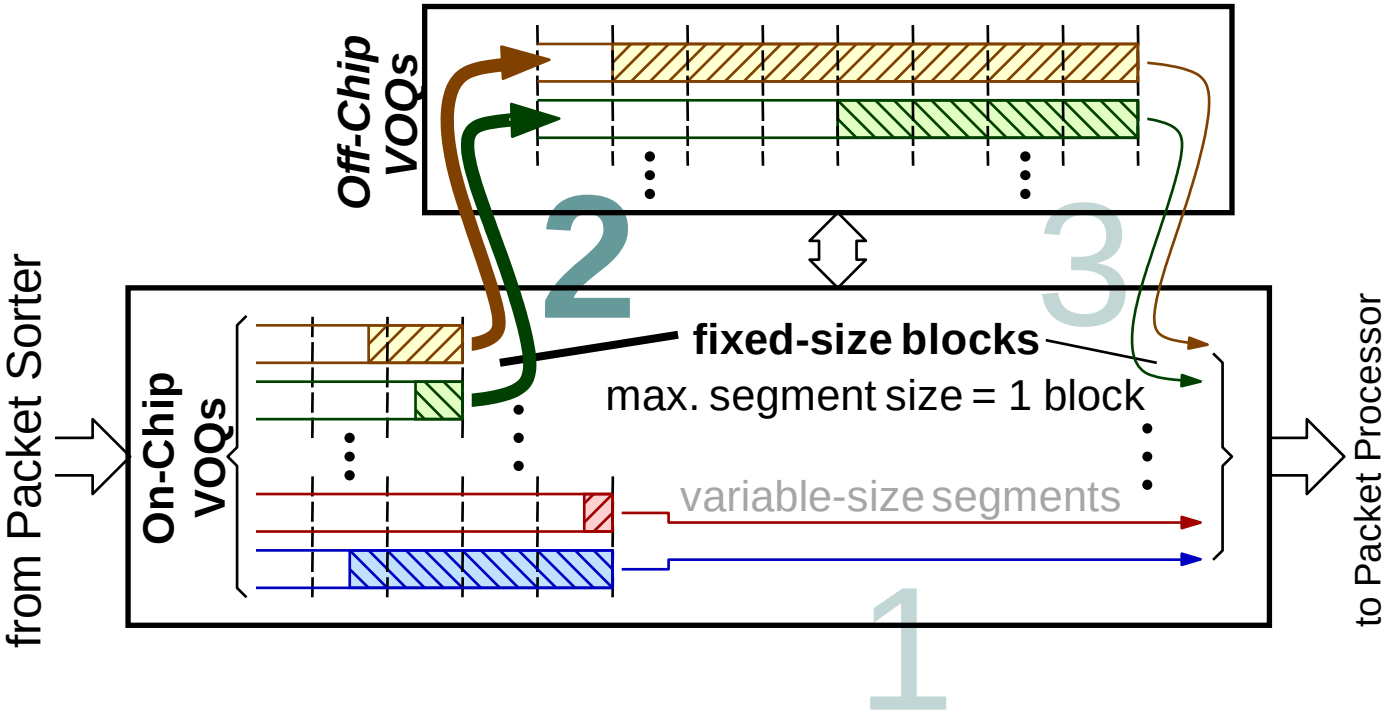
Segment Transfers

1. **From On-Chip VOQs to Packet Processor**
2. From On-Chip to Off-Chip VOQs
3. From Off-Chip VOQs to Packet Processor



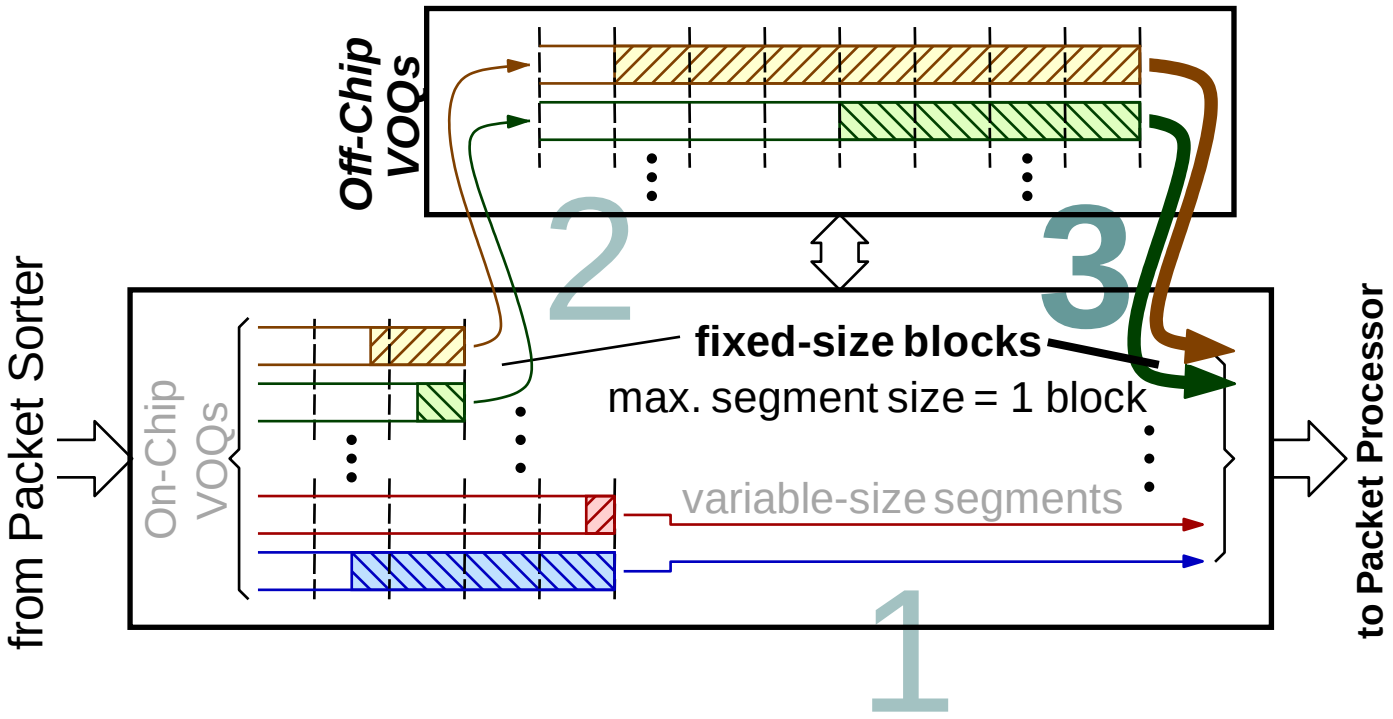
Segment Transfers

1. From On-Chip VOQs to Packet Processor
2. **From On-Chip to Off-Chip VOQs**
3. From Off-Chip VOQs to Packet Processor



Segment Transfers

1. From On-Chip VOQs to Packet Processor
2. From On-Chip to Off-Chip VOQs
3. **From Off-Chip VOQs to Packet Processor**



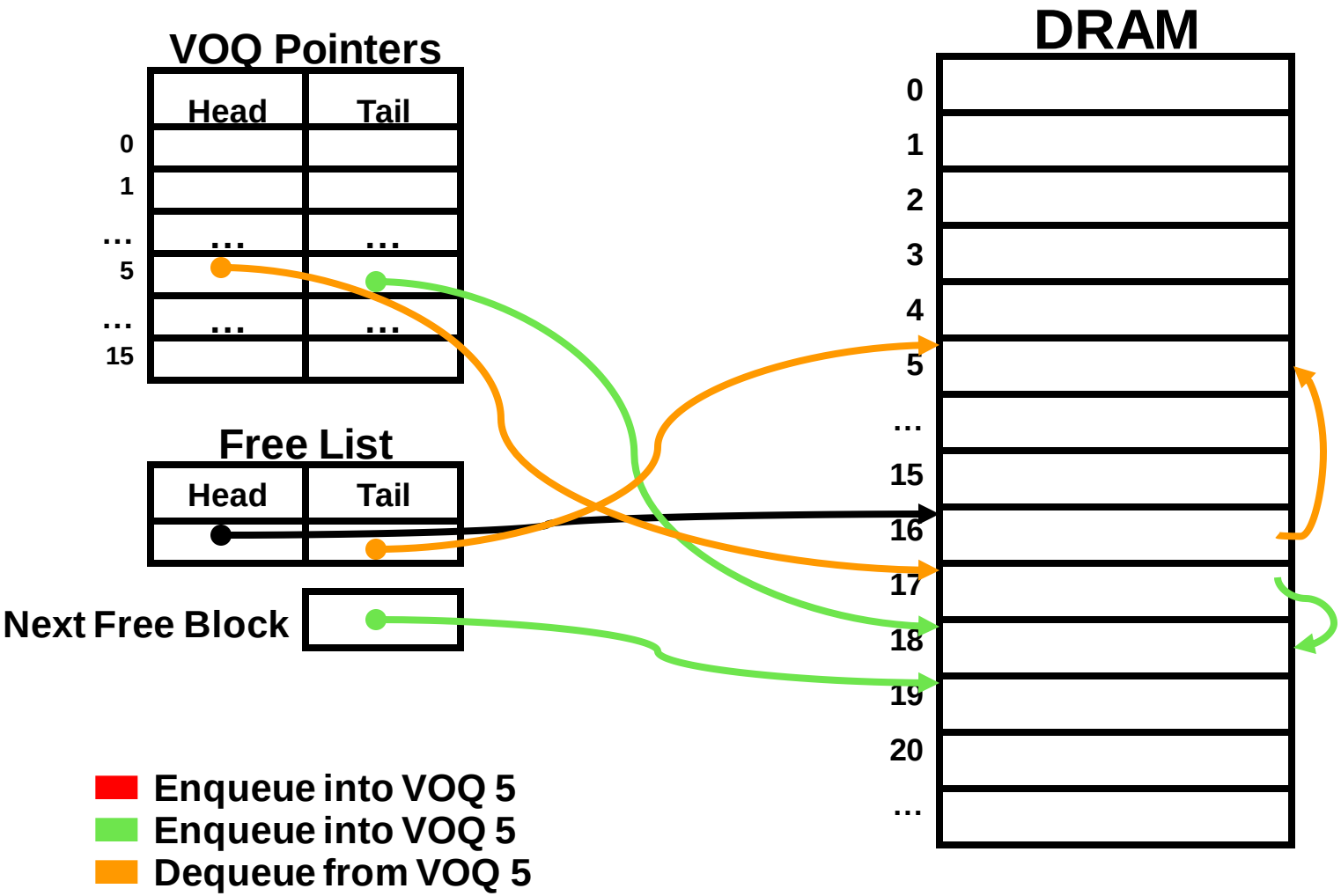
Linked List Management

- **Large VOQs migrate to DRAM**
  - Traffic stored in linked-lists of fixed-size blocks
  - Dynamic allocation of external memory
- **Block size needs to be:**
  - Large to benefit from DRAM burst length
  - Small to minimize size of On-Chip VOQs
- **2 Basic Operations**
  - Enqueue
  - Dequeue
- **Free blocks stored in Free-Block List**

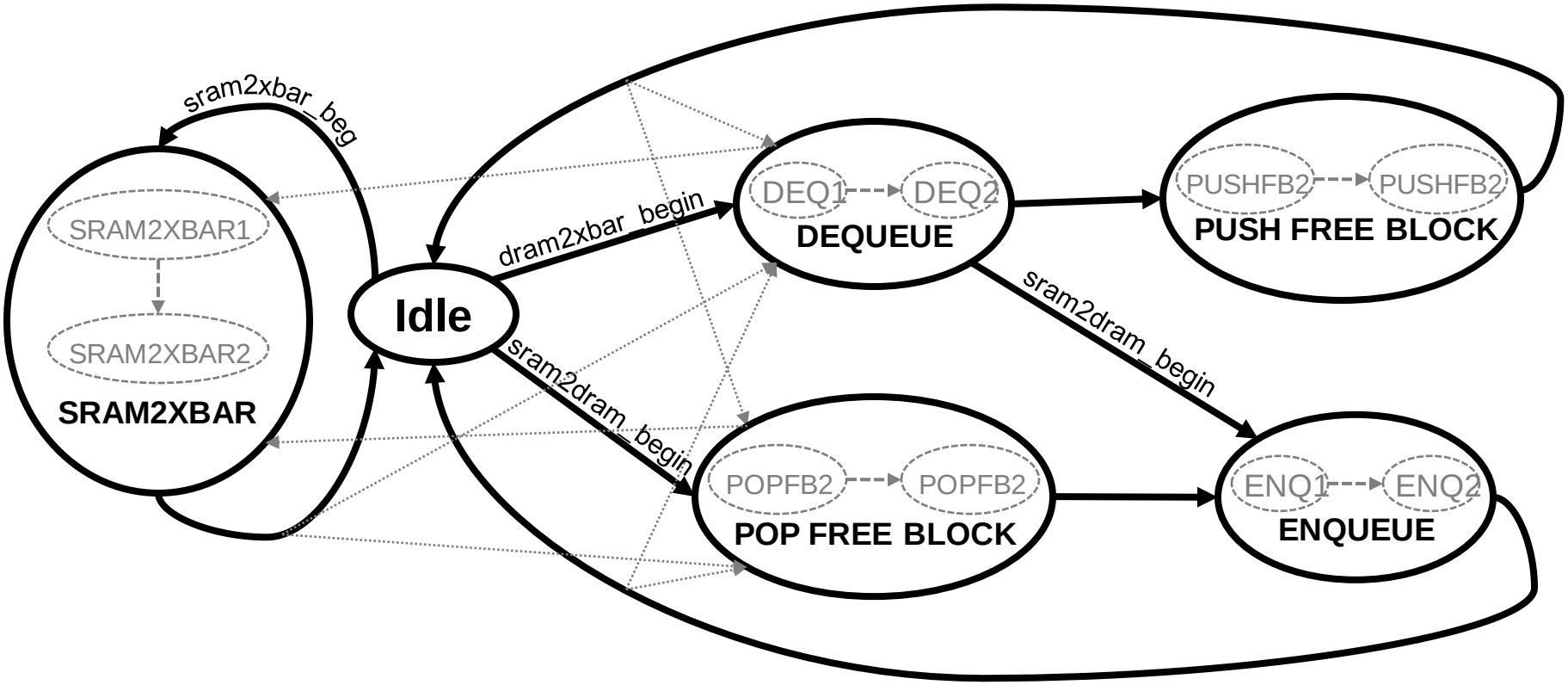
Basic Linked List Operations

- **Enqueue**
  - Get free block from Free-Block list
  - Write data in the new block
  - Update Next-Block pointer of the last block
  - Update VOQ Tail pointer
- **Dequeue**
  - Read data from the first block
  - Read Next-Block from first block
  - Update VOQ Head pointer
  - Put free block in Free-Block list

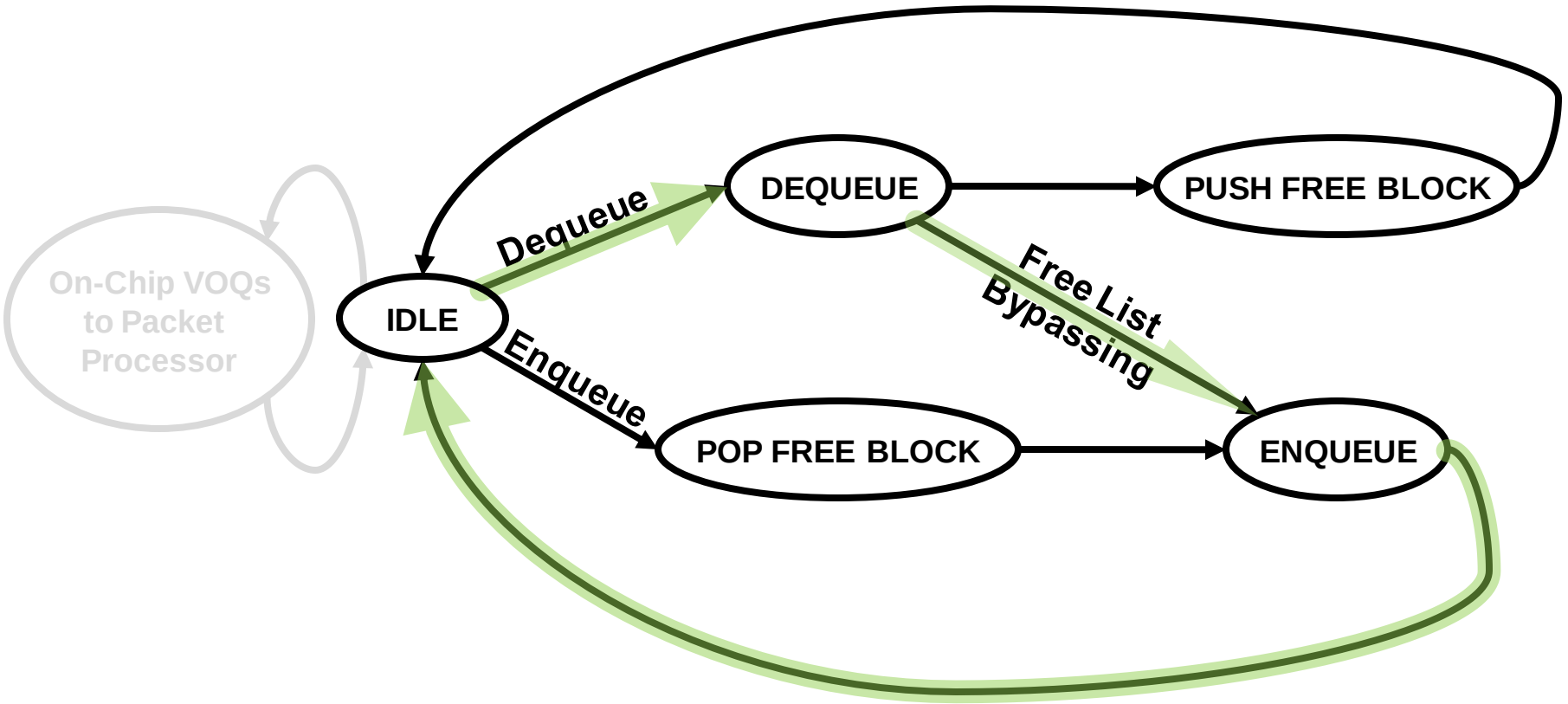
Enqueue/Dequeue Example



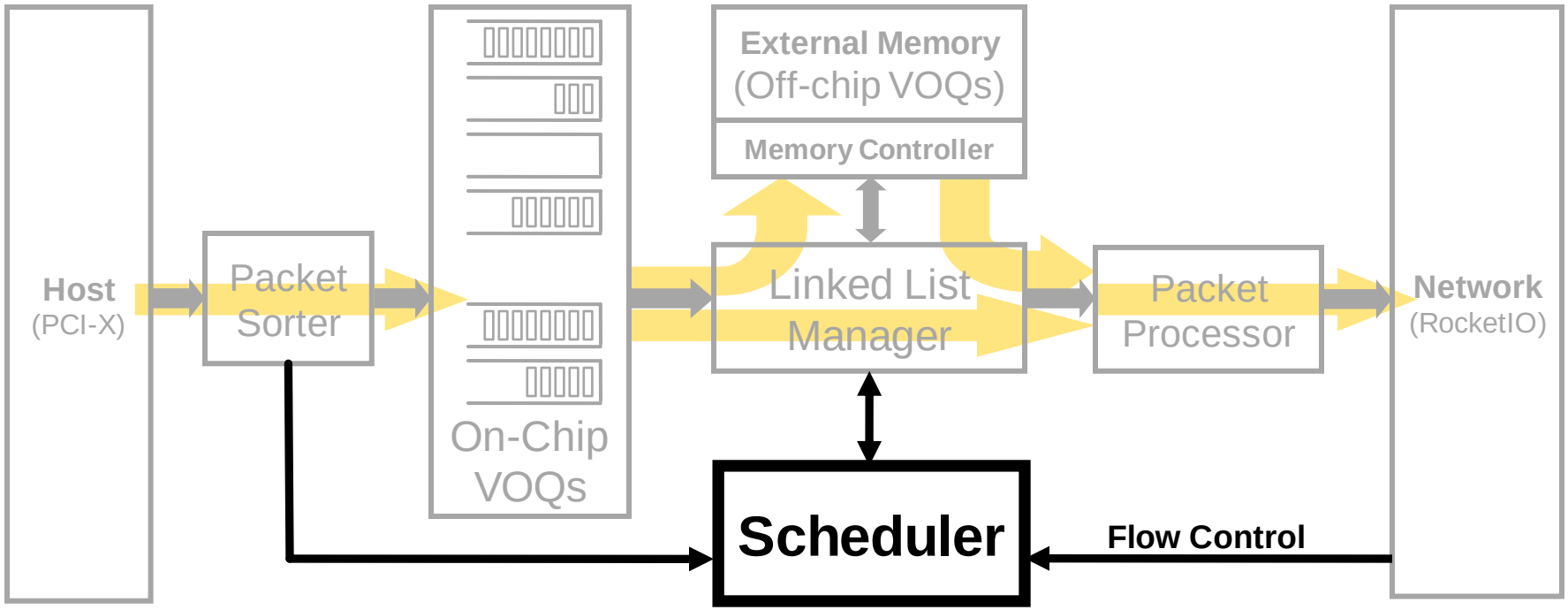
Finite State Machine (FSM)



Finite State Machine (FSM)



Scheduler


Host
(PCI-X)
Packet
Sorter
On-Chip
VOQs
External Memory
(Off-chip VOQs)
Memory Controller
Linked List
Manager
Packet
Processor
Network
(RocketIO)
Scheduler
Flow Control

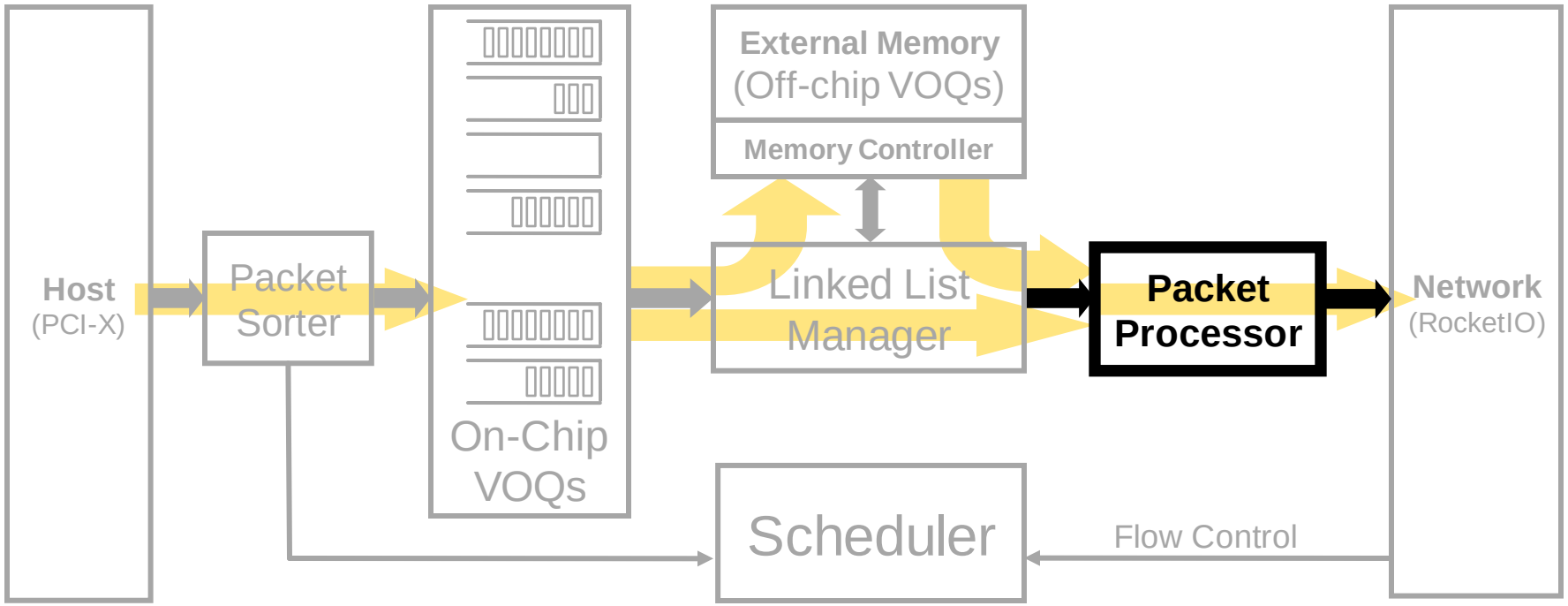
Scheduler

- **Keeps track of each VOQ**
  - On-chip occupancy
  - Off-chip occupancy
- **Employs Flow Control** (network & local)
  - Number of sent data words
  - Number of credits
- **Implements Scheduling**
  - Builds VOQ eligibility masks
  - Enforces scheduling policy
- **Instructs Linked List Manager**

Scheduling Issues

- **Determining Eligibility**
  - One eligibility mask for each kind of transfer
  - Eager approach
  - Lazy approach
- **Scheduling Policy**
  - Round-Robin
  - Weighted Round-Robin
  - Deficit Round-Robin
  - Strict Priority
    - Starvation?

Packet Processor

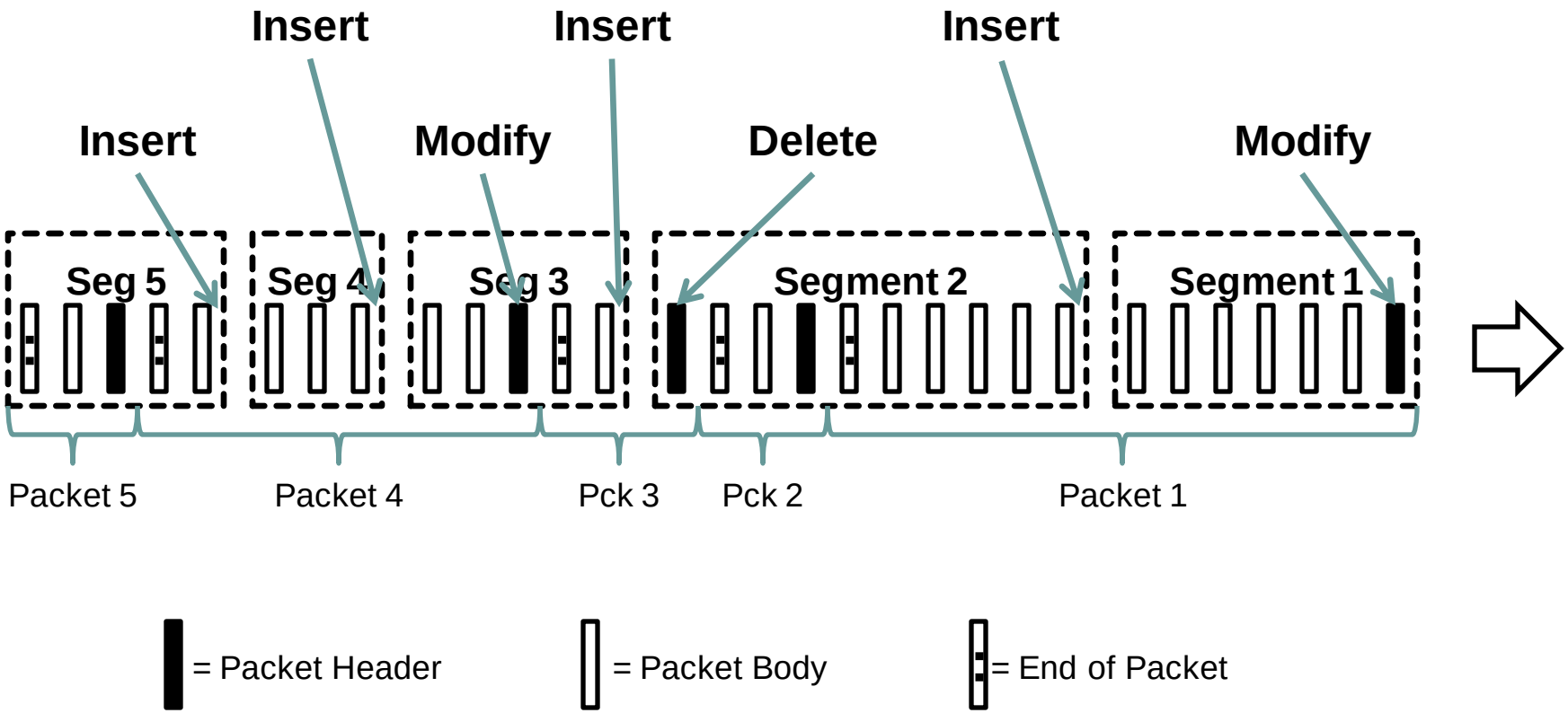


Packet Processor

- **Processes Network Traffic**
  - Receives variable-size segments
  - Creates autonomous network packets
- **Performs 3 Basic Operations**
  - Insert header
  - Modify header
  - Delete header
- **Implemented as 3-stage pipeline**
- **Greatly depends on packet nature**
  - RDMA packets well suited

Example of packet processing

Traffic passing through Packet Processor



Implementation

- **3 major versions**
  - Full
  - “No external memory”
  - “No VSMP segmentation”
- **Variations of individual modules**
  - Packet Processor
    - with/without segmentation
  - On-Chip VOQs
    - with BRAM/Xilinx FIFOs
  - Linked List Manager
    - with/without external memory support

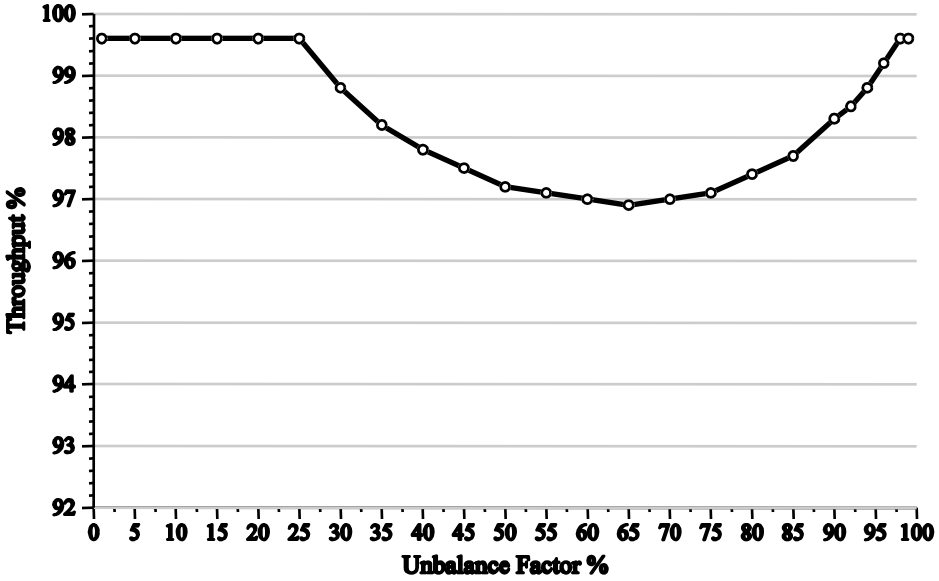
FPGA Hardware Cost Results (8 VOQs)

Queue Mgr Version	LUTs	Slices	Flip Flops	BRAMs	Gate Count
Full	2962 (7%)	2106 (10%)	1571 (3%)	34 (17%)	2263151
No External Memory	2713 (6%)	1900 (9%)	1467 (3%)	34 (17%)	2260321
No VSMPS	**3430 (8%)**	2639 (13%)	2286 (5%)	37 (19%)	2471047

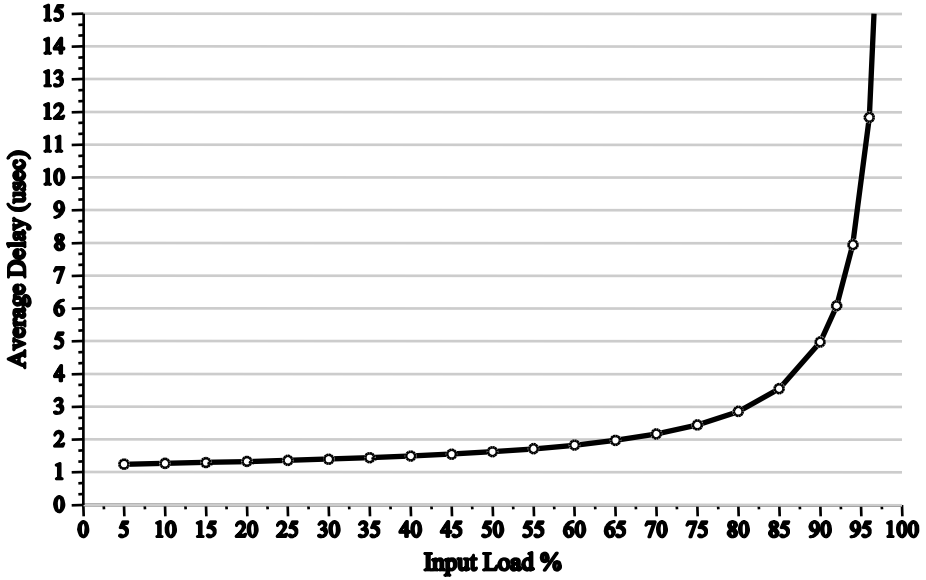
Module	LUTs	Slices	Flip Flops	BRAMs	Gate Count
Packet Sorter with segmentation	179 (1%)	135 (1%)	80 (1%)	0 (0%)	1909
Packet Sorter no segmentation	42 (1%)	25 (1%)	12 (1%)	0 (0%)	392
On-Chip VOQs BRAM	**320 (1%)**	236 (1%)	170 (1%)	**31 (16%)**	2035342
On-Chip VOQs Xilinx FIFOs	**904 (2%)**	1408 (7%)	1648 (4%)	**32 (16%)**	2119015
Scheduler	2240 (5%)	1256 (6%)	428 (1%)	1 (1%)	86226
Linked List Mgr with ext mem	680 (1%)	365 (1%)	425 (1%)	0 (0%)	7069
Linked List Mgr no ext mem	33 (1%)	23 (1%)	18 (1%)	0 (0%)	426
Packet Processor	521 (1%)	511 (2%)	617 (1%)	0 (0%)	9983

Network Performance Results*

- Verified previous theoretical and simulation results about the behavior and performance of buffered crossbar.



Throughput measurements using unbalanced traffic patterns.



Average Delay vs. Input Load under uniform traffic. Max load is 96%.

*The experiments were conducted by Vassilis Papaefstathiou, member of the CARV team

Presentation Outline

- NI Queue Manager
  - Introduction
  - Key Concepts
  - Architecture & Implementation
- **NI Design for CMPs**
  - **NI Design Goals**
  - NI Design Issues
  - Proposed NI Design
- Conclusions & Future Work

NI Design Goals

- **High Performance**
  - Low Latency
  - High Bandwidth
- **Scalability**
- **Reliability**
- **Protection & Security**
- **Communication Overhead Minimization**

Presentation Outline

- NI Queue Manager
  - Introduction
  - Key Concepts
  - Architecture & Implementation
- **NI Design for CMPs**
  - NI Design Goals
  - **NI Design Issues**
  - Proposed NI Design
- Conclusions & Future Work

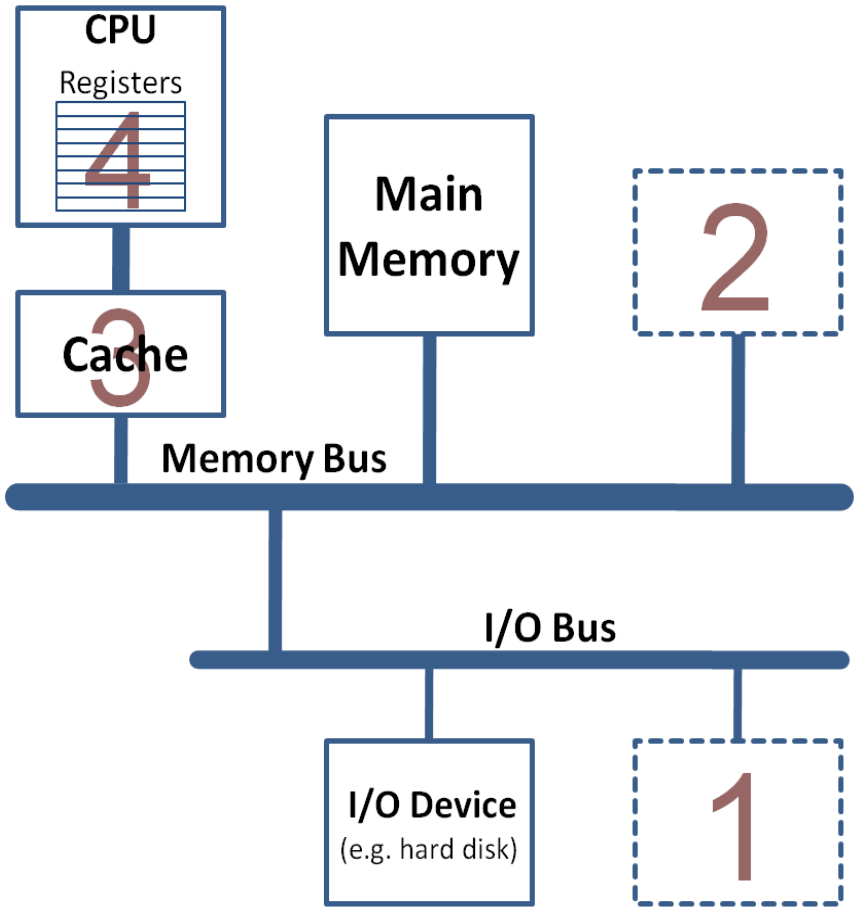
NI Placement

Lower Performance
Standard Interfaces
More Buffer Space

Processor Proximity

1. **I/O Bus**
2. **Memory Bus**
3. **Cache**
4. **CPU Registers**

Higher Performance
Proprietary Interfaces
Resource Sharing
Less Buffer Space

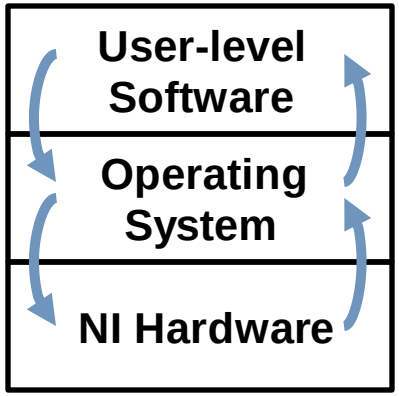


NI Data Transfer Mechanisms

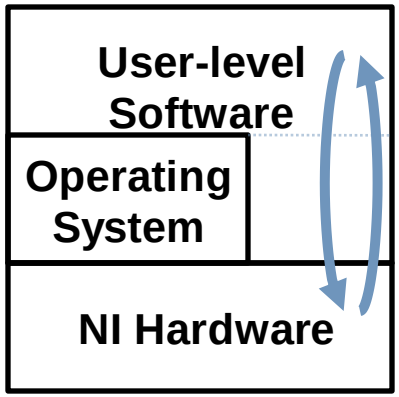
- **Programmed IO (PIO)**
  - Uncached stores, loads or I/O instructions
    - Low bandwidth
  - Cached stores, loads
    - data transferred in cache blocks
    - requires coherence
  - Occupies Processor to copy data
- **Using Direct Memory Access (DMA)**
  - Decouples Processor
  - Requires Virtual to Physical Address Translation

NI Virtualization & Protection

- **Traditional approach**
  - System Call to Access NI
  - Very Costly (high latency)



- **User-level NIs (ULNIs)**
  - Bypass the OS
  - Mapped to Virtual Memory
  - Uses OS paging mechanism for protection



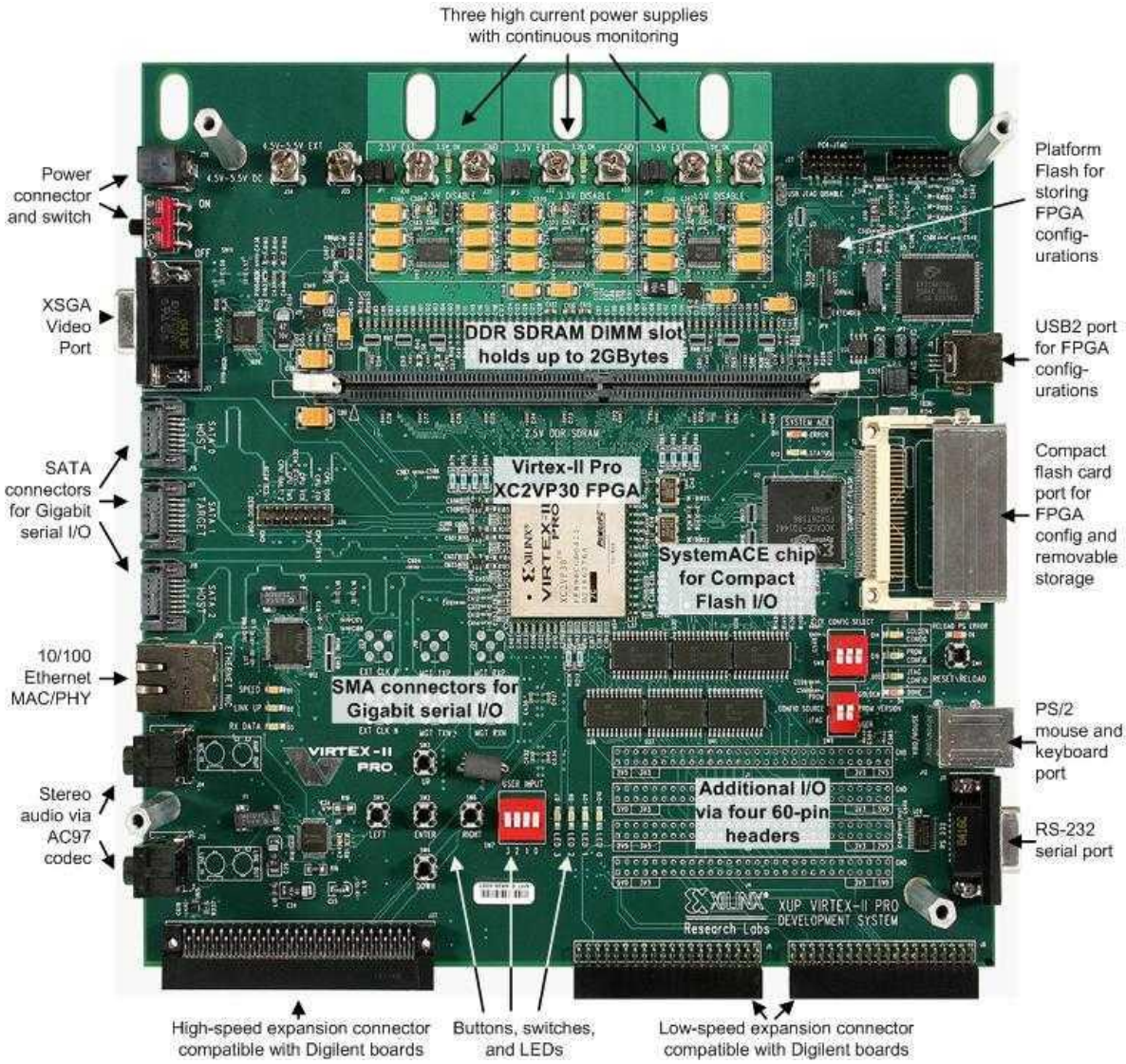
Presentation Outline

- NI Queue Manager
  - Introduction
  - Key Concepts
  - Architecture & Implementation
- **NI Design for CMPs**
  - NI Design Goals
  - NI Design Issues
  - **Proposed NI Design**
- Conclusions & Future Work

Design Goals / Desired Features

- **Targeted to future Chip Multiprocessors**
  - thousands to millions of nodes
- **Tightly coupled with the processor**
- **Low Complexity/Cost**
  - Compared to processor and local memory
- **Resources (e.g. memory)**
  - Shared with Processor
  - Dynamically allocated (only to active connections)
- **Low Overhead transfer initiation**
- **Powerful Communication Primitives**
  - Bulky Data Transfers
  - Control & Synchronization Traffic

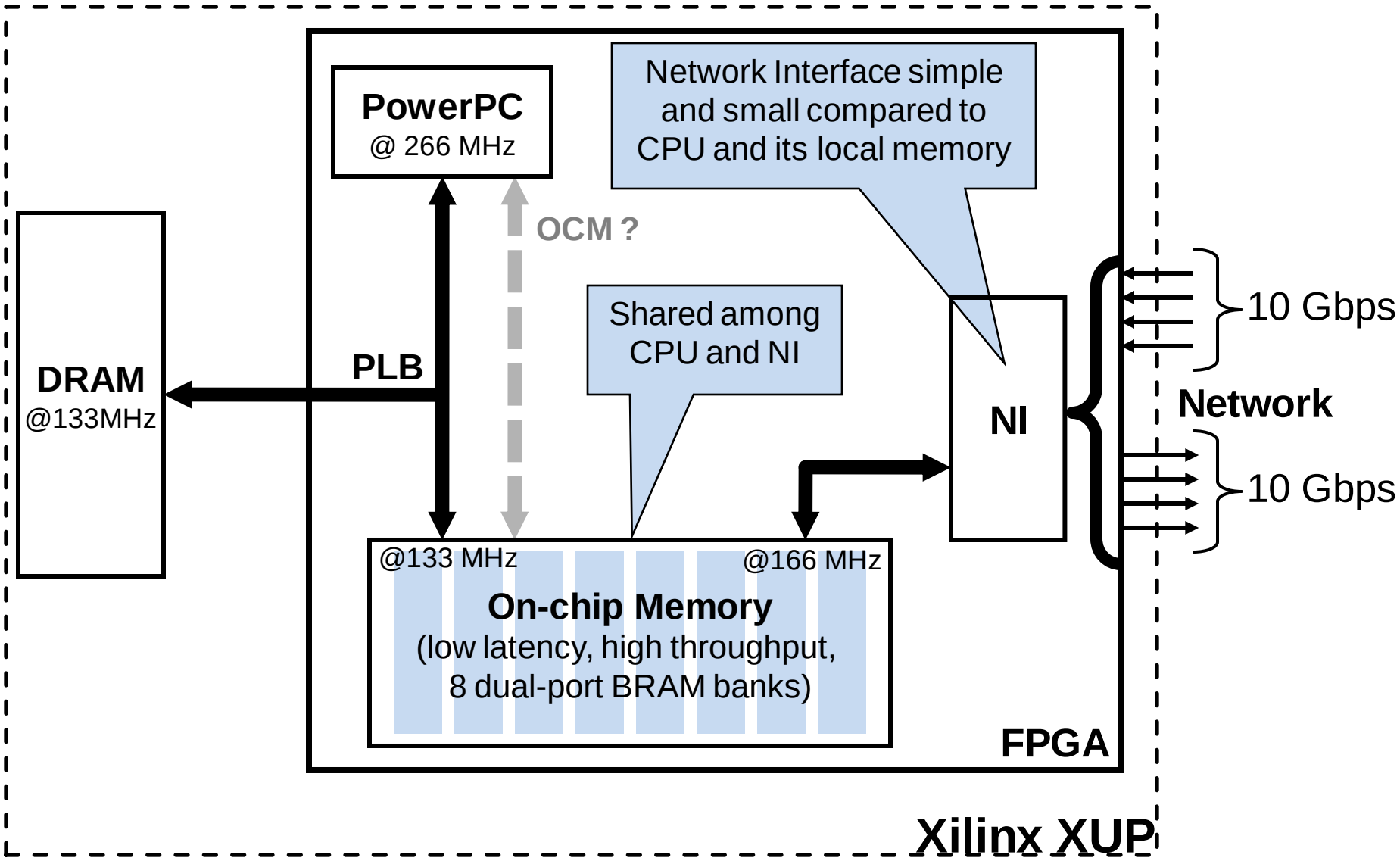
Target Hardware/System



Xilinx XUP board

- **CPU:** PowerPC
- **OS:** Linux
- **On-chip BRAM**
  - Scratchpad
  - Cache?
- **External DRAM**
- **10 Gbps Network**
  - RocketIO

Target Hardware/System

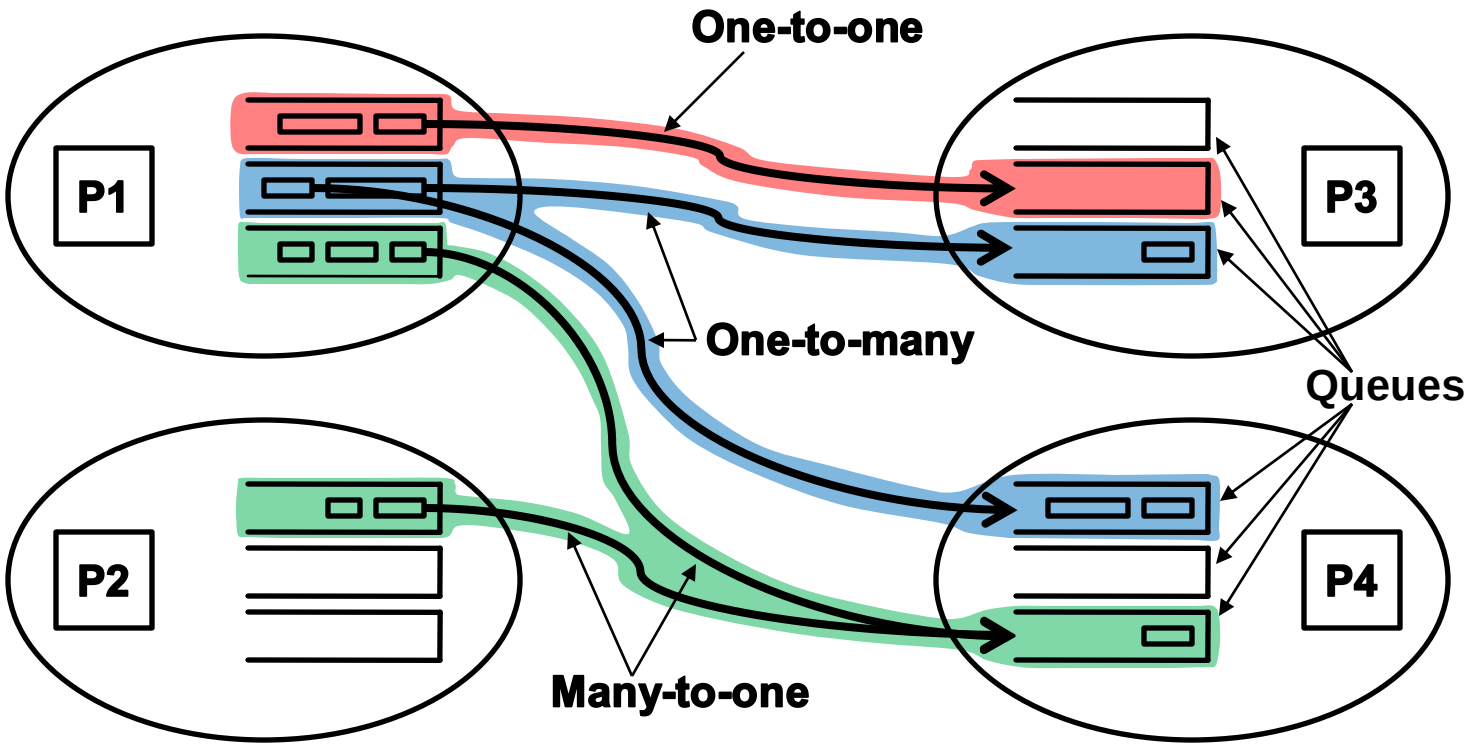


Communication Primitives

- **Remote DMA**
  - Bulky Data Transfers
  - Producer-Consumer
  - Facilitates Zero-Copy Protocols
  - Requires Virtual-to-Physical Translation
- **Message Queues**
  - Low Latency, Minimal Overhead communication
  - Small Data Transfers
  - One-to-One, One-to-Many, Many-to-One
  - Powerful synchronization primitive

Message Queues

- **One-to-One:** e.g. Small Data Transfers
- **One-to-Many:** e.g. Job Dispatching
- **Many-to-One:** e.g. Synchronization

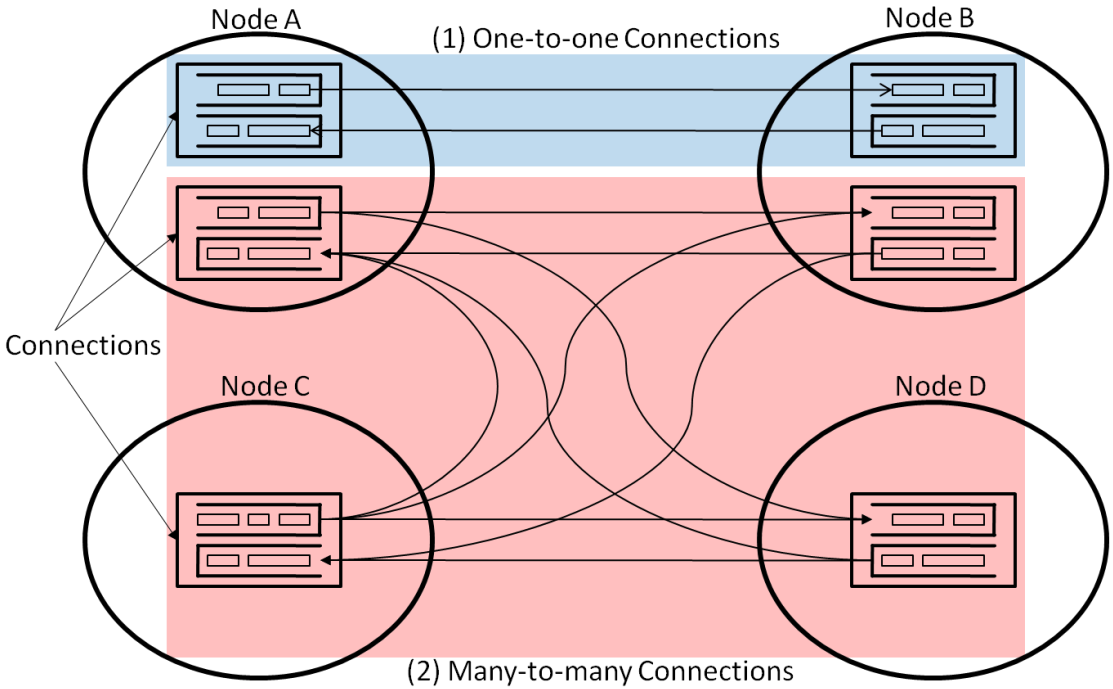


Connections

- **Mechanism for**
  - Sending Messages
  - Initiating RDMA operations
- **A Connection consists of:**
  - Queue Pair (Incoming & Outgoing)
  - Information needed for RDMA
  - Various Queue, Flow Control metadata
- **Connection Types**
  - One-to-One (lower overhead, higher security)
  - Many-to-Many (better resource utilization)

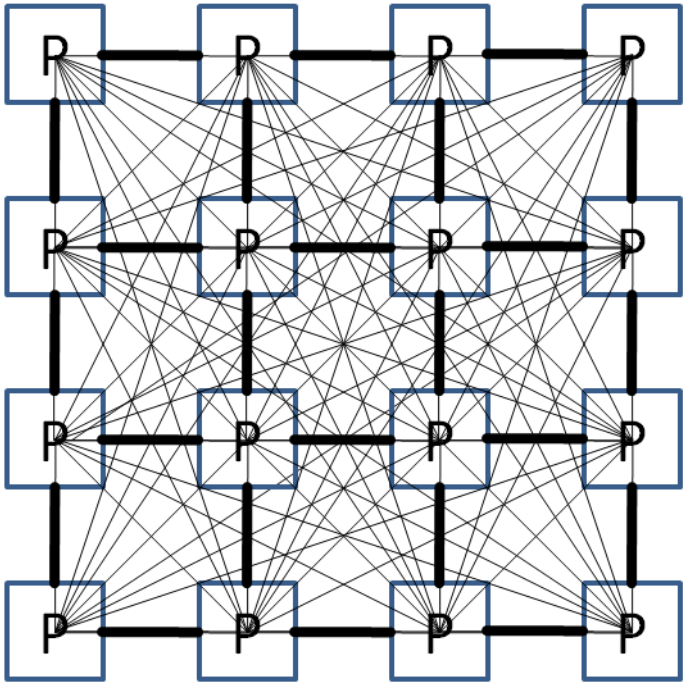
Connection Types

- **One-to-One**
  - Incoming & Outgoing Queue is One-to-One
- **Many-to-Many**
  - Incoming Queue is Many-to-One Or/And Outgoing Queue is One-to-Many



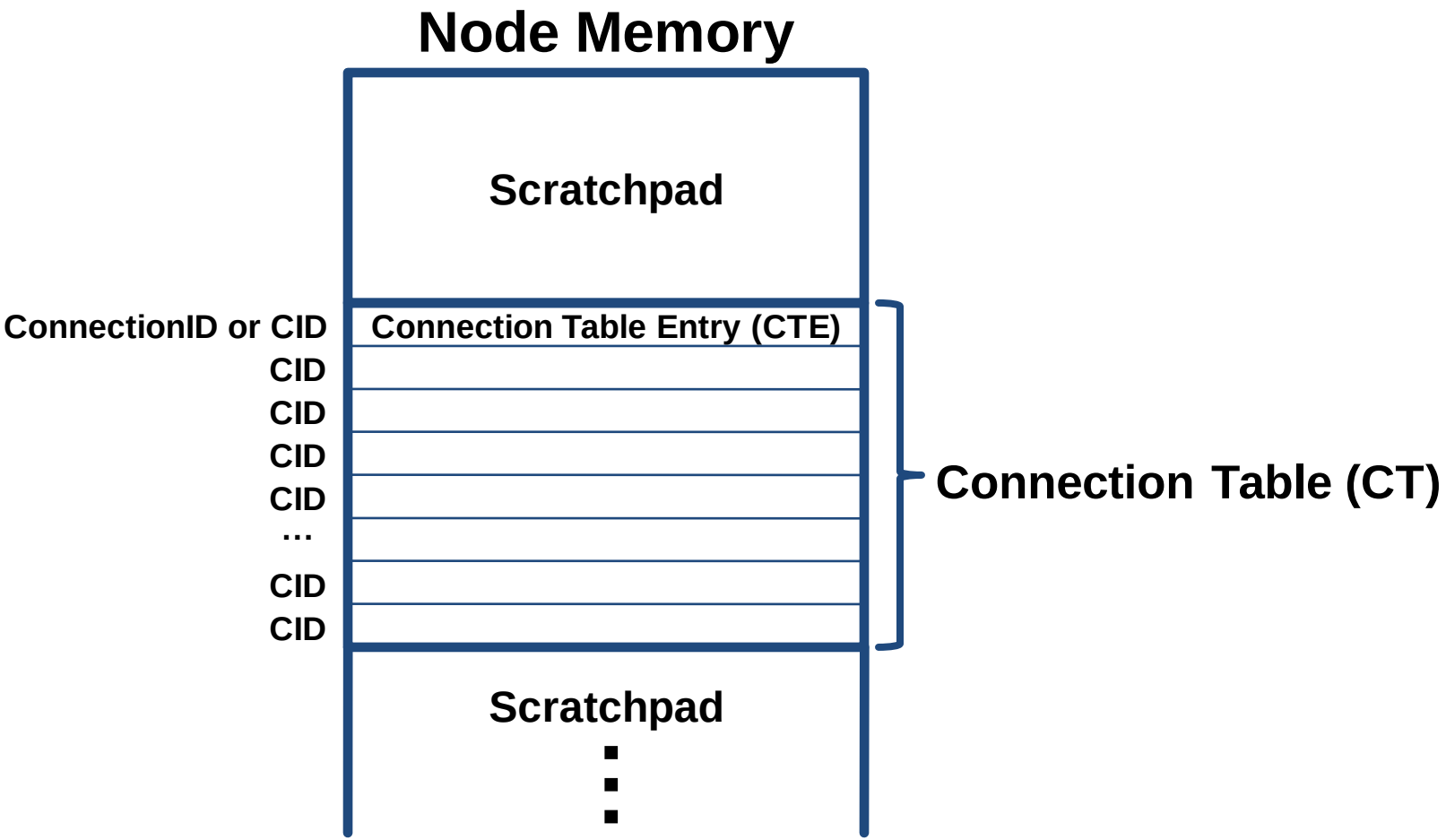
Connection Types - Example

- **One-to-One:** Local Communication
- **Many-to-Many:** Global Communication



Connection Table (CT)

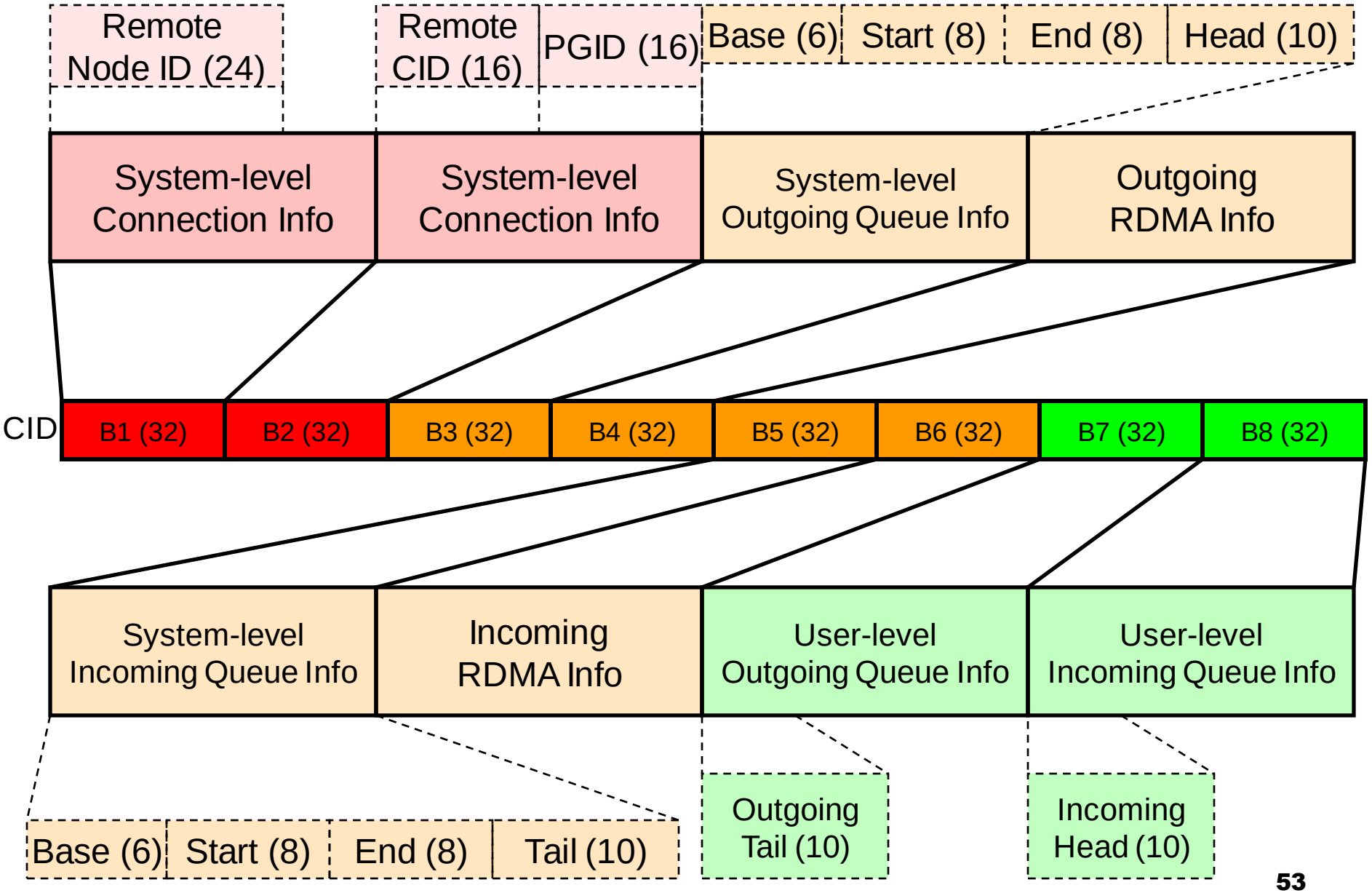
- Connections reside in Connection Table in the form of Connection Table Entries (CTEs)



Connection Table Entry (CTE)

- **Each CTE stores**
  - Destination Info (for one-to-one)
  - Protection Info (for many-to-many)
  - Incoming/Outgoing Queue Info
  - Incoming/Outgoing RDMA Info
  - Flow Control Info
  - Other Info
    - e.g. Notification Type

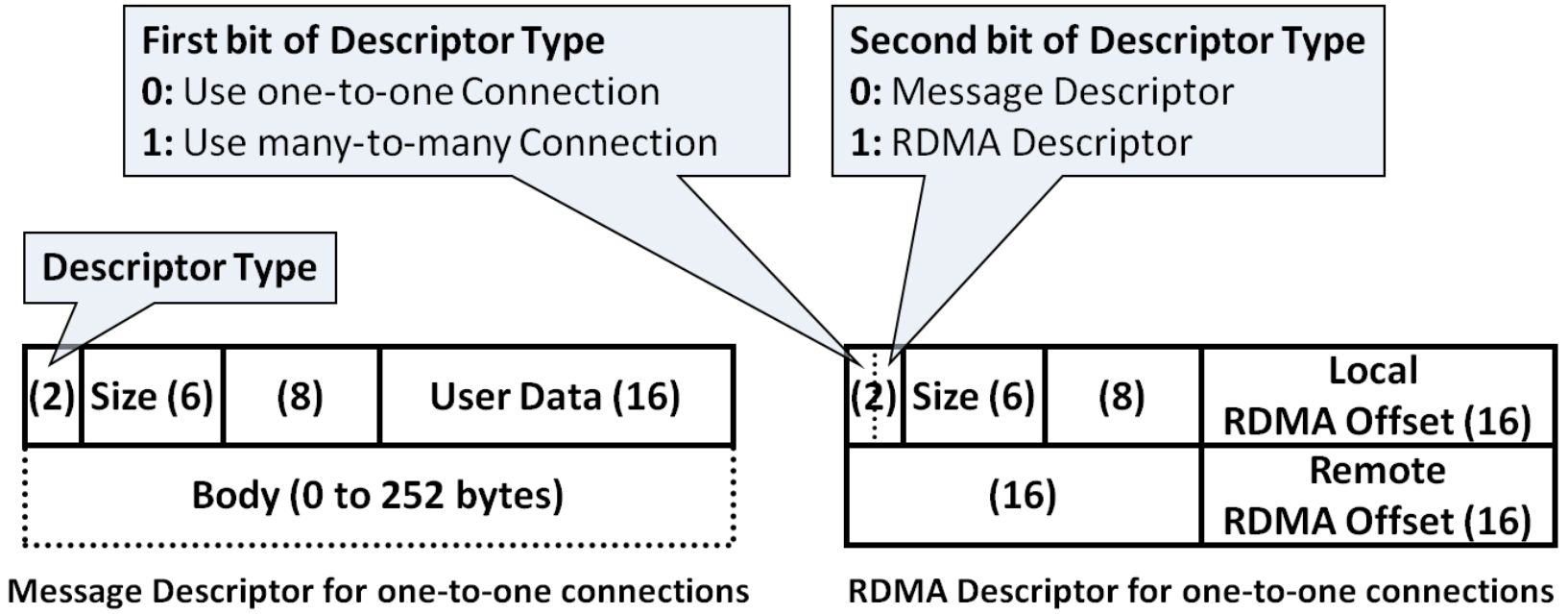
Connection Table Entry for Xilinx XUP



Software Interface

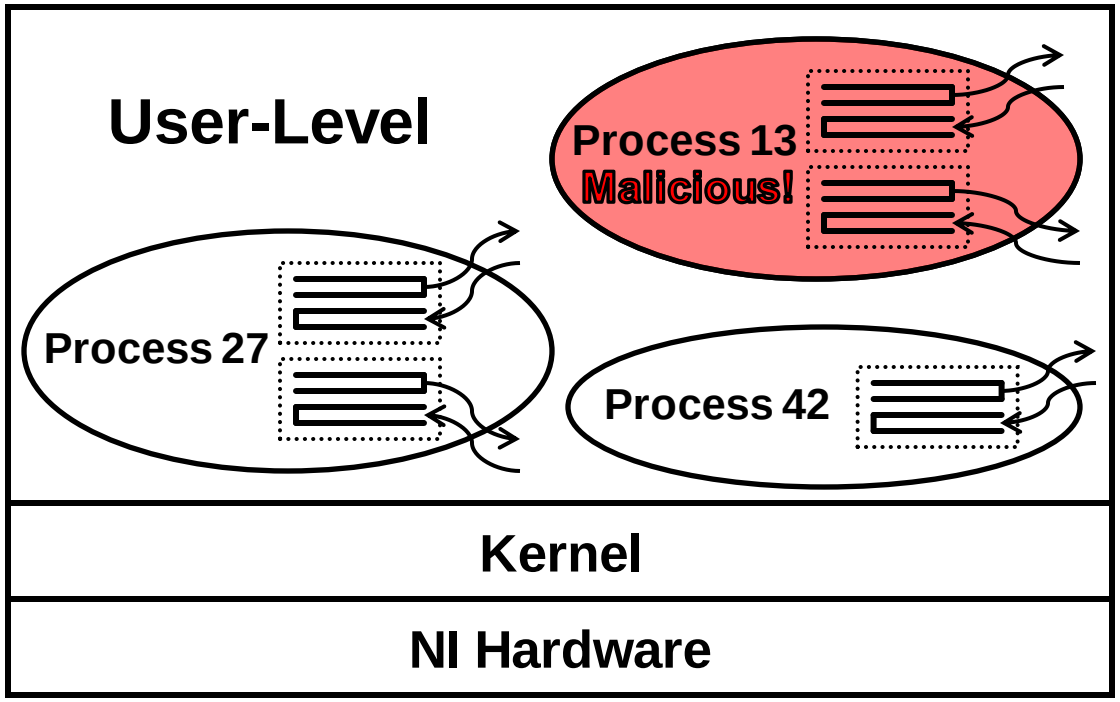
- **Send Message or Initiate RDMA**
  - Read Head/Tail of Outgoing Queue
  - Post Descriptor
  - Write Tail of Outgoing Queue
  - (Poll Head of Outgoing Queue)
- **Receive Message or RDMA Notification**
  - Poll Tail of Incoming Queue (or get Interrupt)
  - Read Message (or Descriptor)
  - Write Head of Incoming Queue

Software Interface - Descriptors



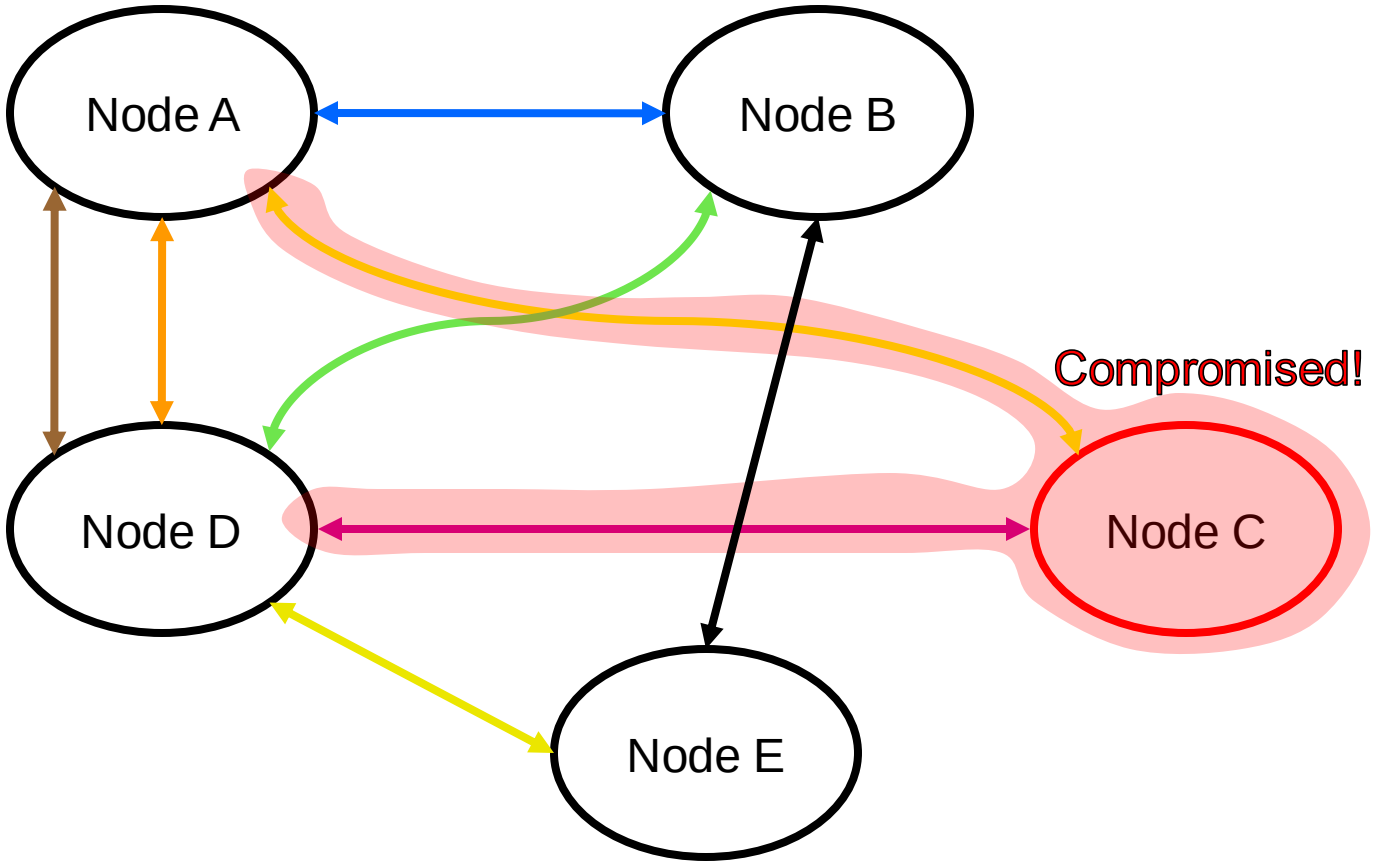
Protection – Intranode Protection

- **Isolation of malicious processes. Not allowed to:**
  - Read or write data of connections of other processes
  - Corrupt connections of other processes



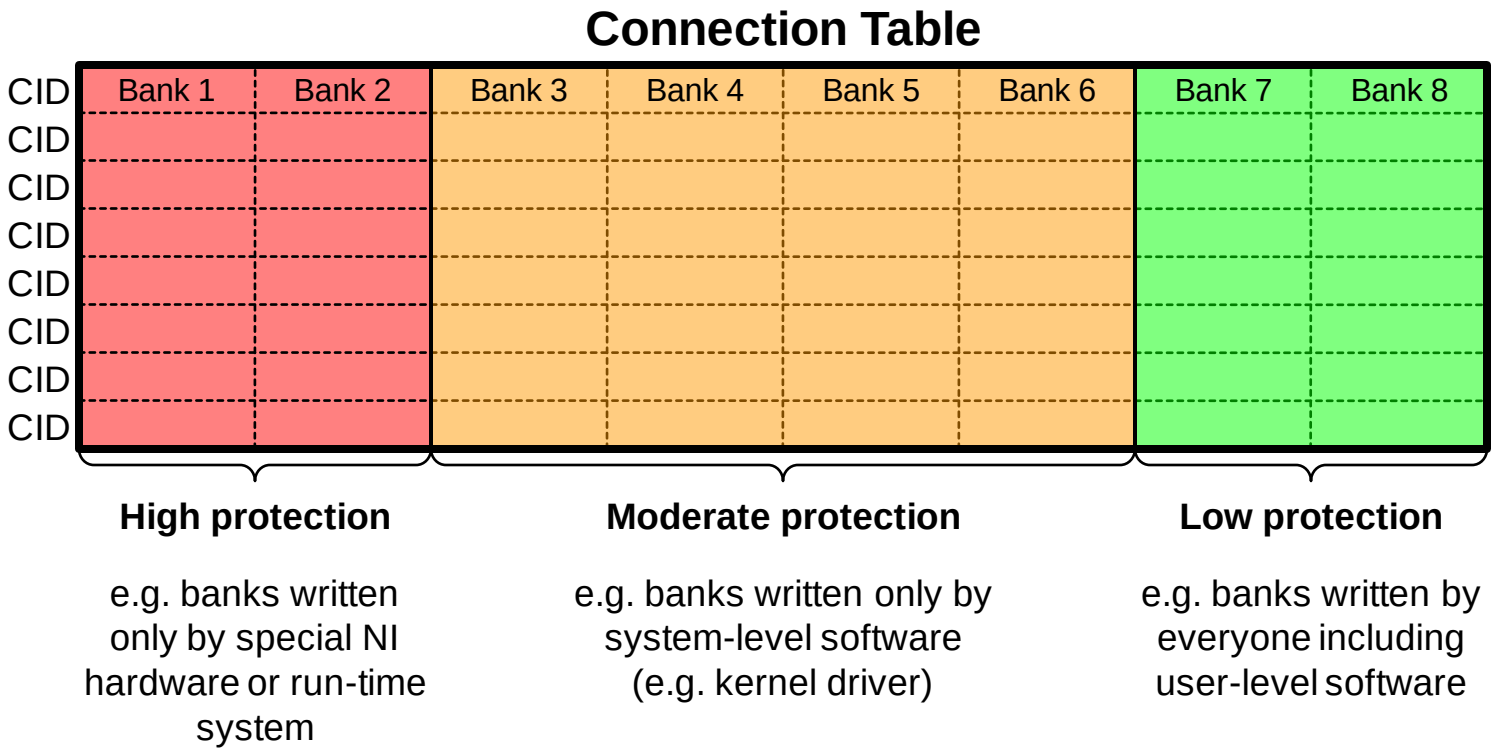
Protection – Internode Protection

- **Isolation of compromised node. Not allowed to:**
  - Compromise other nodes
  - Corrupt connections of other processes



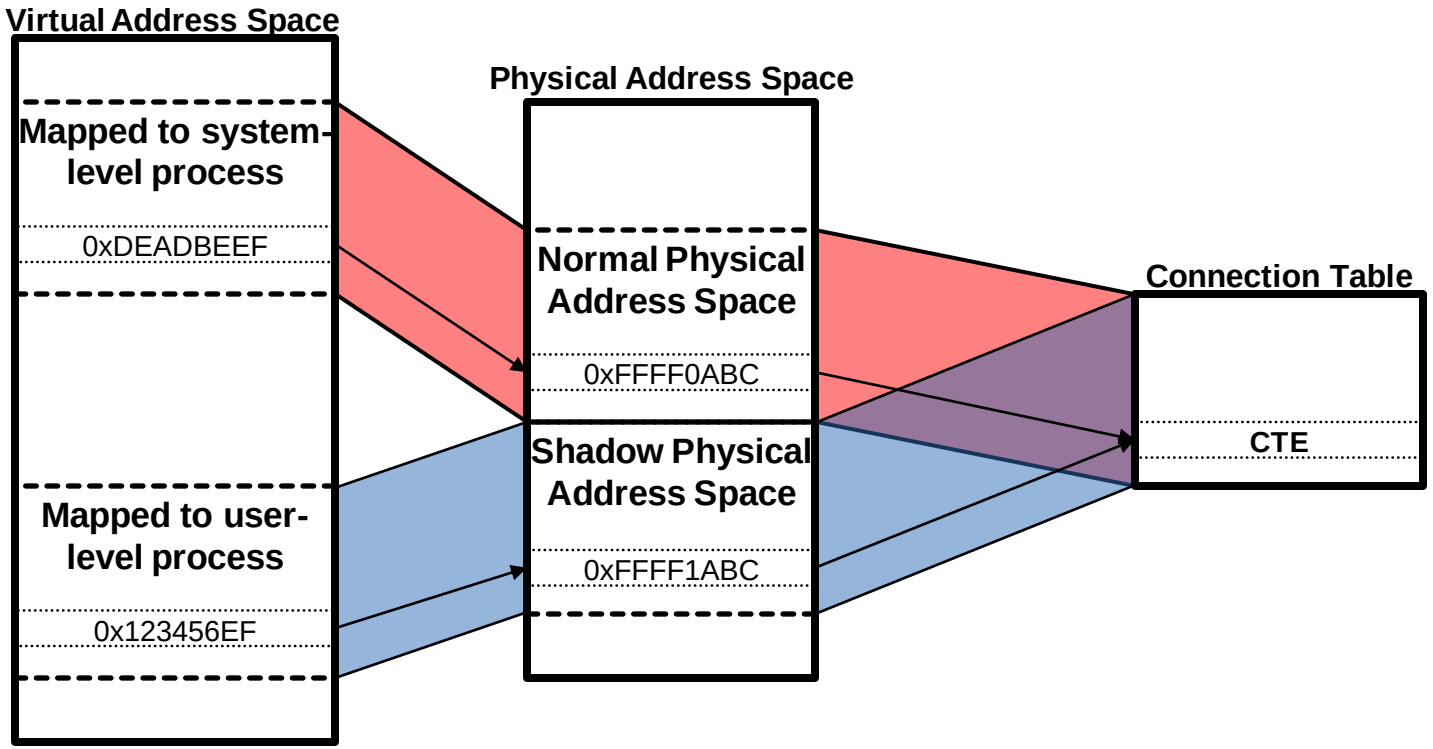
Protection in the Proposed NI

- **Creation of 3 Protection Zones**



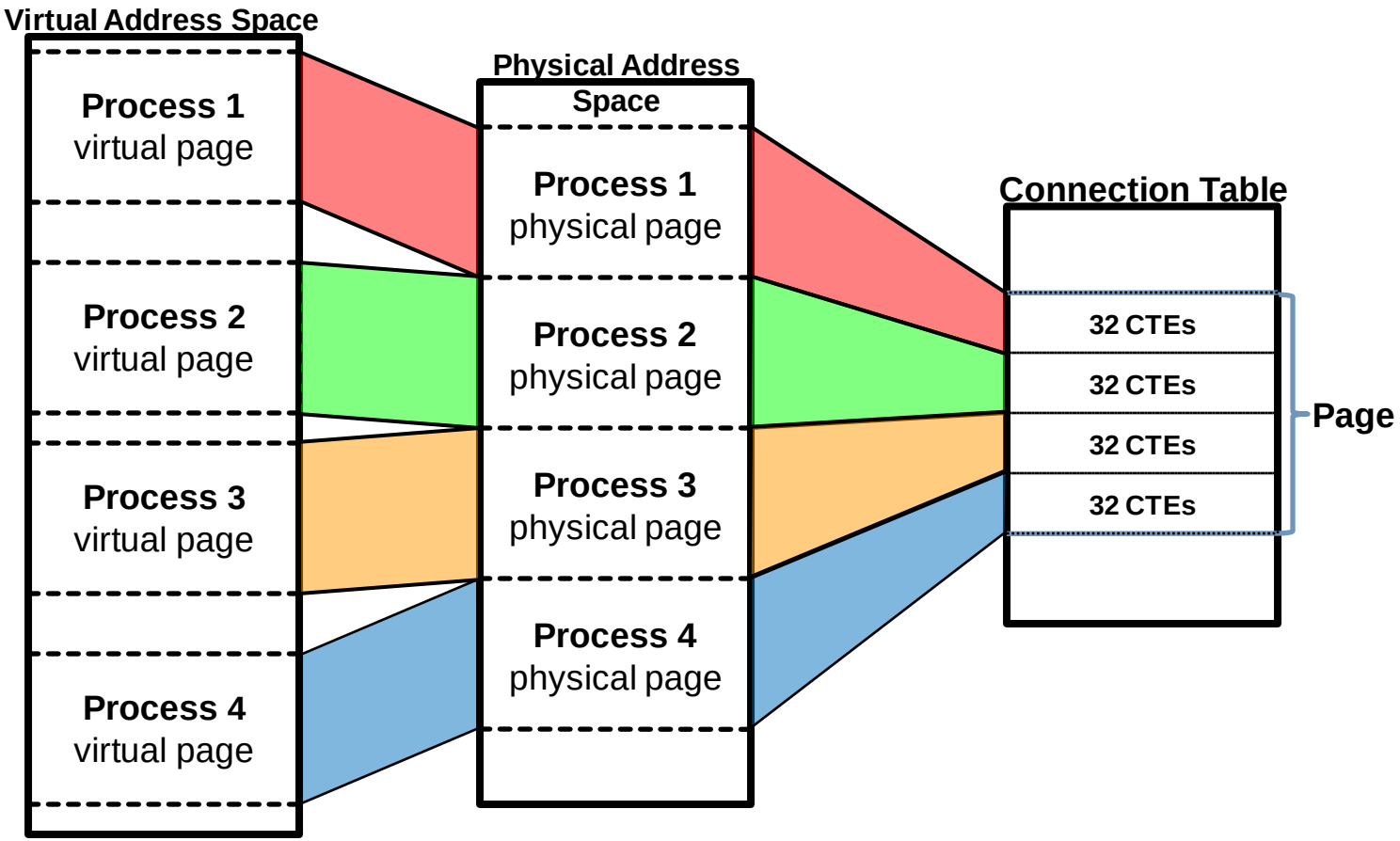
Controlling Access to the CT

- **Distinguish User-level from Kernel-level**
  - Use of Shadow Address Spaces
  - Double the required Address Space



Controlling Access to the CT

- **Distinguish different User-level processes**
  - Fine-grain protection



Presentation Outline

- Key Concepts
- NI Queue Manager
  - Architecture
  - Implementation
- IPC: NI Design Issues
- Proposed NI Design
- **Conclusions & Future Work**

Conclusions

- **NI Queue Manager**
  - Feasibility and Effectiveness of VSMPS
  - Confirmed buffered crossbar results
  - Novel Packet Processing Mechanism
    - Eliminates need for reassembly
- **Proposed NI Design**
  - Lightweight, well suited to future CMPs
  - Powerful communication primitives
    - Message Queues, Remote DMA
  - Notion of Connections/Queues
  - Versatile Protection & Security Mechanisms

Future Work

- **NI Queue Manager**
  - Scalability
    - Dynamic memory allocation for On-Chip VOQs
- **Proposed NI Design**
  - Process migration
  - Flow control
  - Efficient cache coherence support

Thank You!