PCIe® as a Multiprocessor System Interconnect

Kwok Kong

Director of Software Engineering

Integrated Device Technology, Inc

Agenda

- Introduction
- Issues and Solutions
- Software Architecture
- Summary

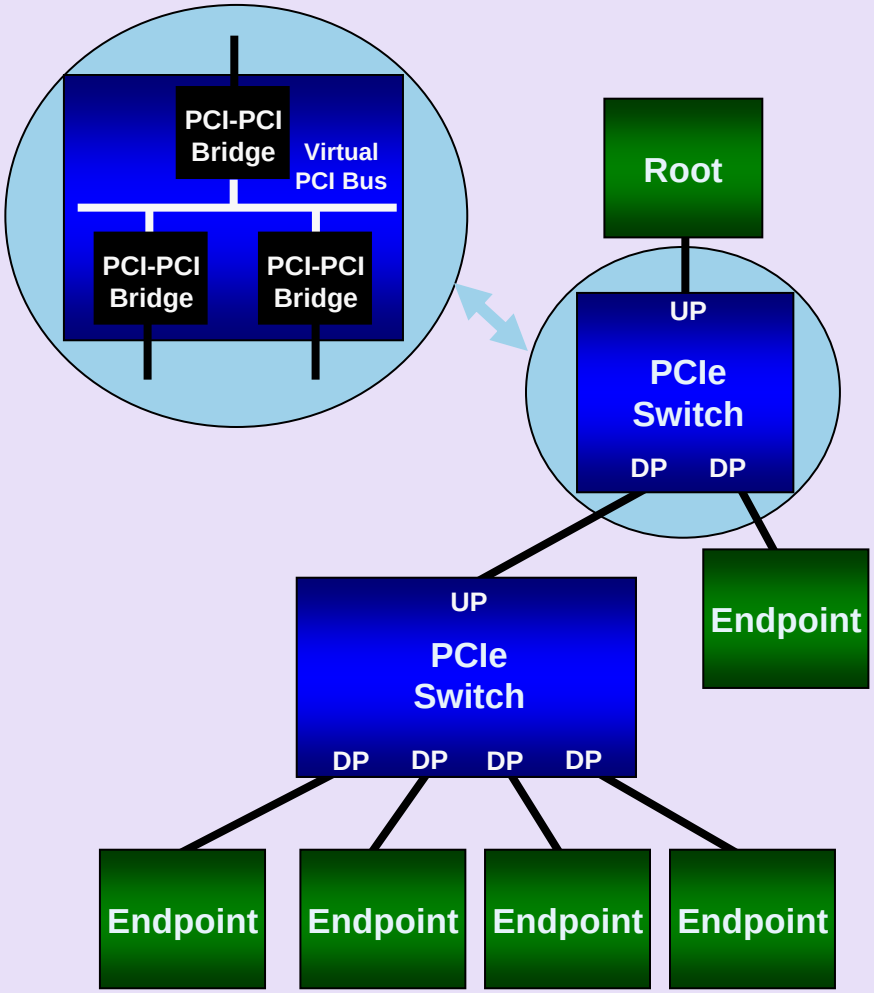
Introduction

- Standard PCI Express® supports a single processor tree
- There are systems that require multiple processors and hence multiple PCI Express trees or domains
- This session describes an example of what it takes to build a multiprocessor system using standard PCI Express and additionally inter-domain switching

Terminology

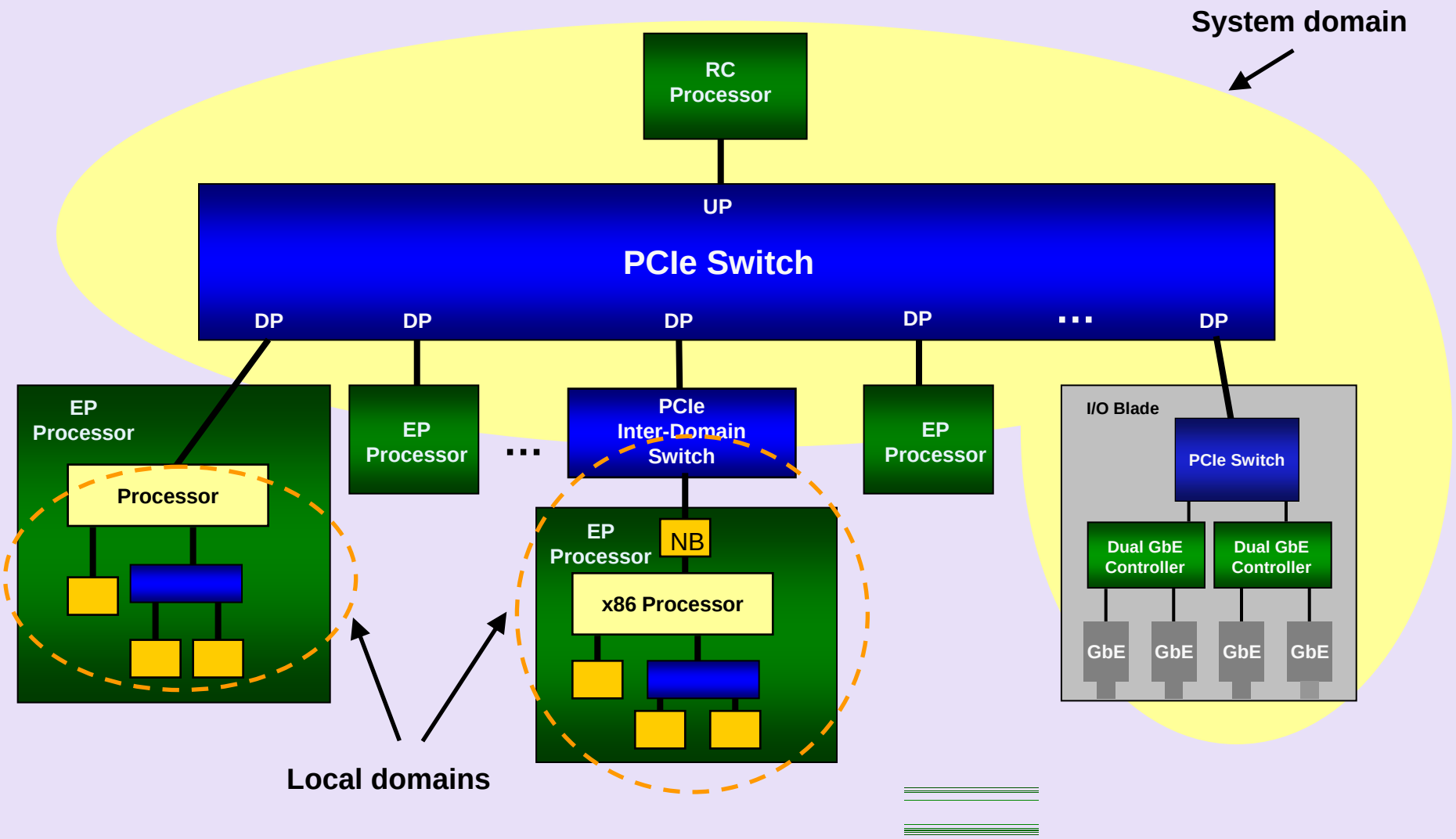
- Root Complex (RC) Processor
  - ✓ Only one RC processor per PCI Express (PCIe®) tree
  - ✓ Owns the enumeration and memory map
- Endpoint (EP) Processor
  - ✓ Any processor on a downstream leaf (i.e., endpoint) of a PCIe tree
- PCIe domain
  - ✓ Single PCIe tree with one RC and one or more endpoints
- Inter-Domain Switch
  - ✓ Provides Non-Transparent Bridging functionality and inter-processor communication mechanisms (not defined by PCI-SIG)
  - ✓ Switch data between PCIe domains

PCI Express Architecture



- Tree of buses topology with a single root
- Root functions:
  - ✓ Configures system (discovery and enumeration)
  - ✓ Interrupts
  - ✓ Error processing
- Roots and endpoints share a single common address space
- Primary means of communication
  - ✓ Memory reads and writes
  - ✓ Interrupts (MSIs and INTx)

PCIe Domains



System Interconnect Issues

- Memory Map Management
  - ✓ Separate per domain
- Enumeration and Initialization
  - ✓ System domain and local domain(s)
- Peer-to-peer Communication
  - ✓ Message passing mechanism
  - ✓ Data transfer
- Others
  - ✓ Interrupts
  - ✓ Error Reporting
  - ✓ Redundancy

Address Domain Separation

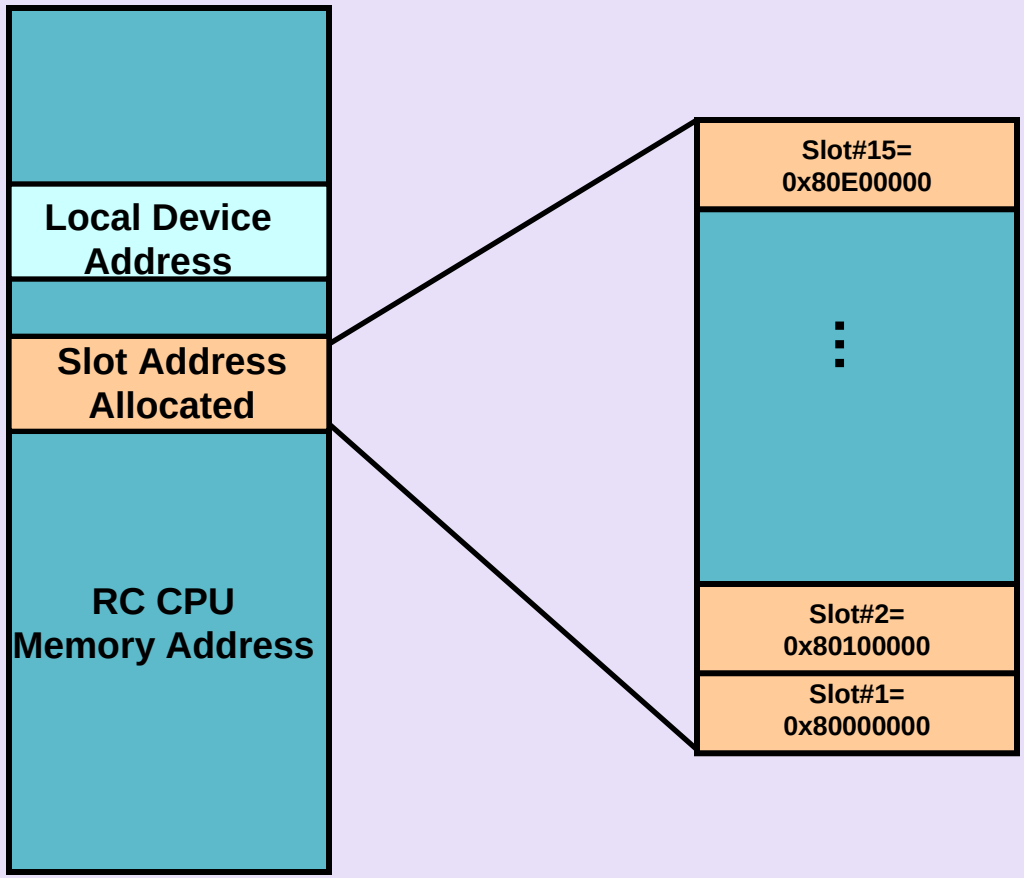
- Use a processor that has the endpoint function built-in
  - ✓ Examples include Intel IOP80333*, Freescale 8548E*, AMCC PowerPC 440SPe*
- Use an inter-domain PCIe switch
  - ✓ Example includes IDT PES16NT2* to support x86 CPU

*Third party marks and brands are the property of their respective owners

RC Processor Enumeration and Initialization

- System PCIe domain needs unique memory map and enumeration
  - ✓ RC processor owns memory map and enumeration
  - ✓ Designated single RC processor for PCIe tree
- RC processor scans the PCIe devices (discovery)
  - ✓ Detects and configures address spaces (via BAR)
  - ✓ All endpoint processors must look like PCIe endpoints to the RC
- System memory address ranges allocated dynamically
  - ✓ System domain PCIe Address is allocated dynamically on the discovery of an endpoint processor
  - ✓ Memory map may change on plug in/out
- At the end of device discovery, RC processor sends messages to all endpoint processors
  - ✓ Notify them the existence and mapping of all other endpoint processors in the system
  - ✓ Endpoint processors do not attempt to enumerate/discover system domain

System Domain Address

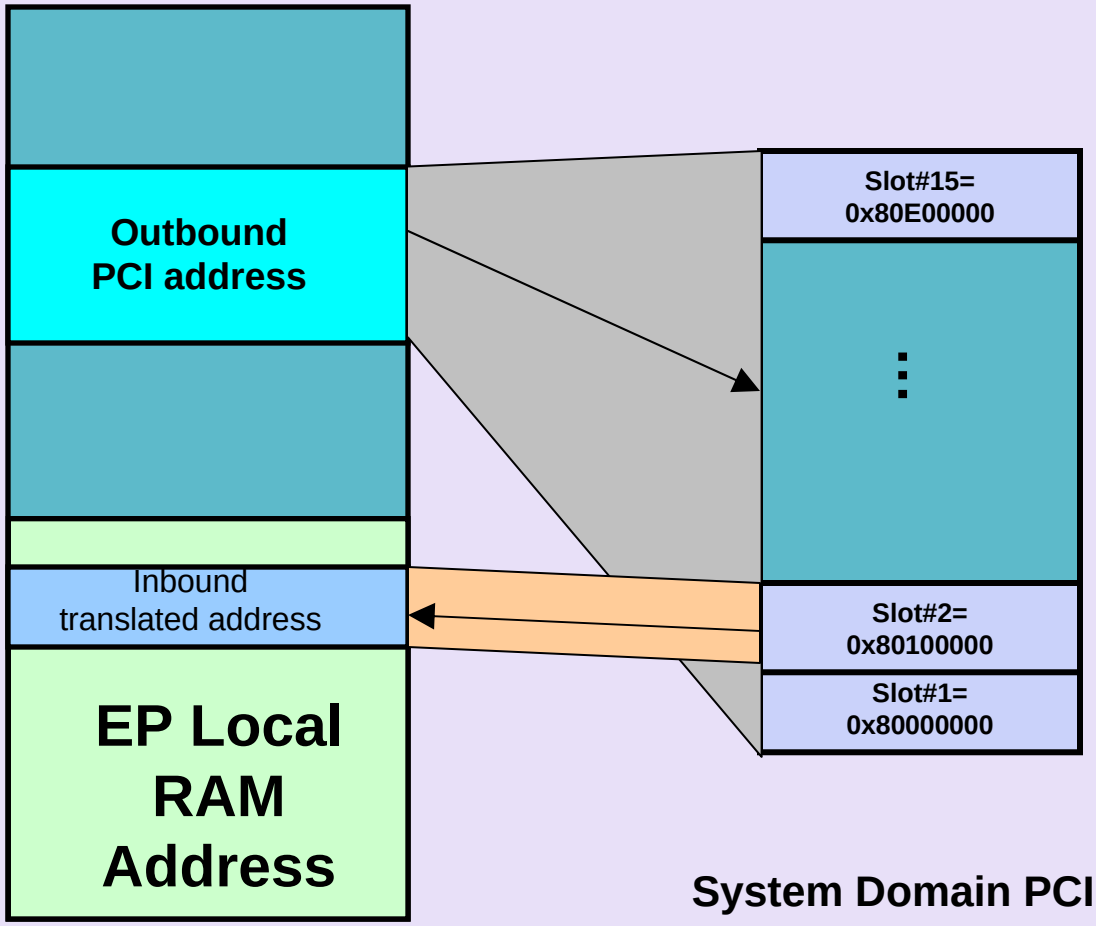


- RC processor goes through normal enumeration procedure
- RC processor allocates address dynamically

Local Domain Initialization

- Local PCIe domain needs unique memory map and enumeration
  - ✓ Endpoint processor owns memory map and enumeration on local domain
- Normal device discovery and enumeration first
  - ✓ Stops at the PCIe interface attached to system domain
  - ✓ PCIe interface is treated as an endpoint to the local domain
- Bus Device Function (BDF) values enumeration is also first done locally
  - ✓ Need translation mechanism of BDF values
  - ✓ Need message mechanism
- System domain RC processor notifies all endpoint processors that a new endpoint processor is discovered
  - ✓ The system domain PCIe address is allocated dynamically
  - ✓ This is added to the address map of the local domain, via address translation mechanism
  - ✓ System domain RC processor notifies all endpoint processors when an endpoint processor is removed

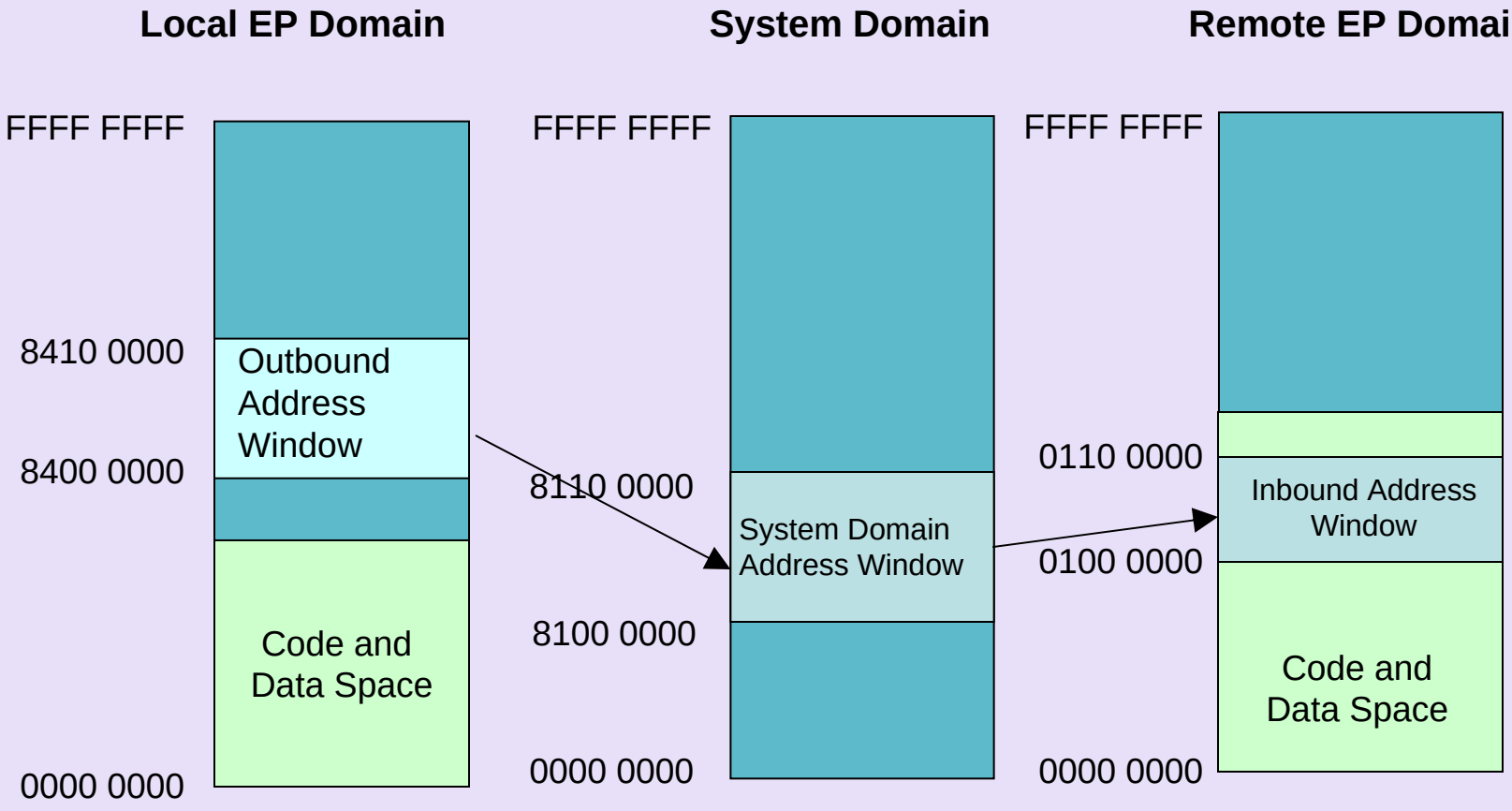
EP Processor Address Translation



- Endpoint has address translation unit
- Translate inbound "myslot" PCIe address to internal local memory
- Translate outbound address to System domain PCIe Address

EP Processor Local Domain Address Space

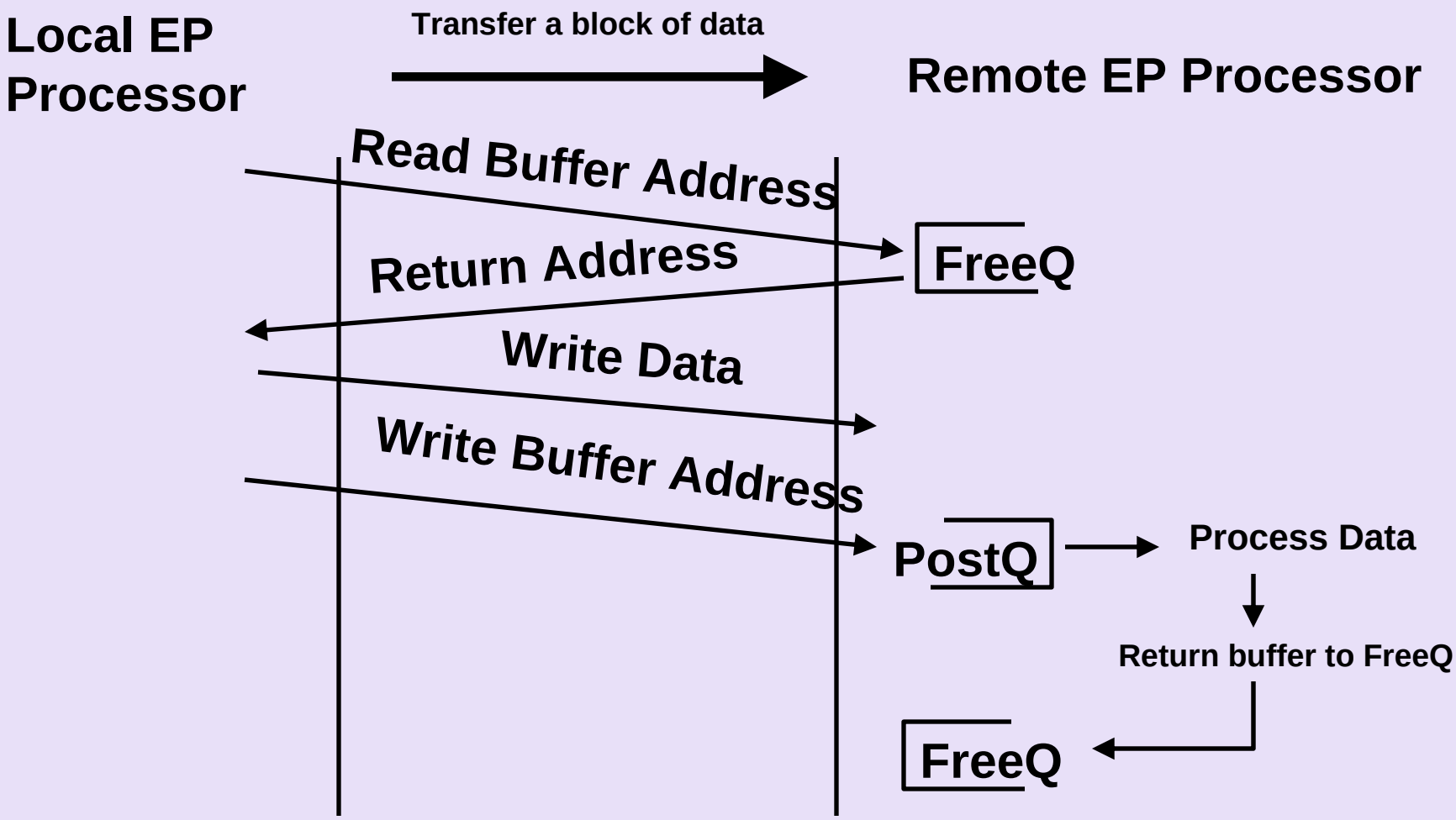
Address Translation Windows Example



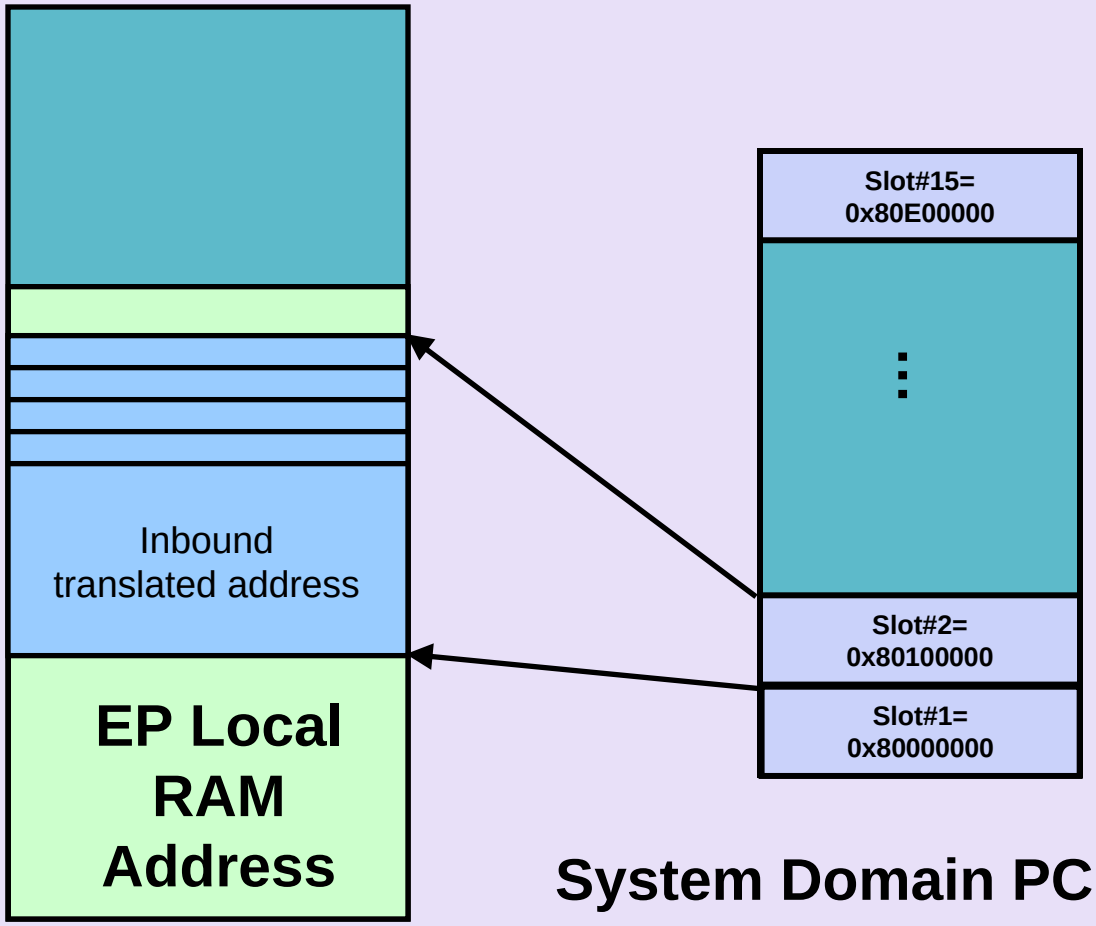
High Performance Peer-to-Peer Communication

- Requires advanced queues
  - ✓ Multiple outstanding data blocks
  - ✓ Received from multiple sources
- Shared memory space
- Descriptor rings with address, length, status indicators
- Free Queue
  - ✓ Contains Free Buffer descriptors at destination endpoint
- Post Queue
  - ✓ Post data to destination endpoint after data transfer

Communication Protocol

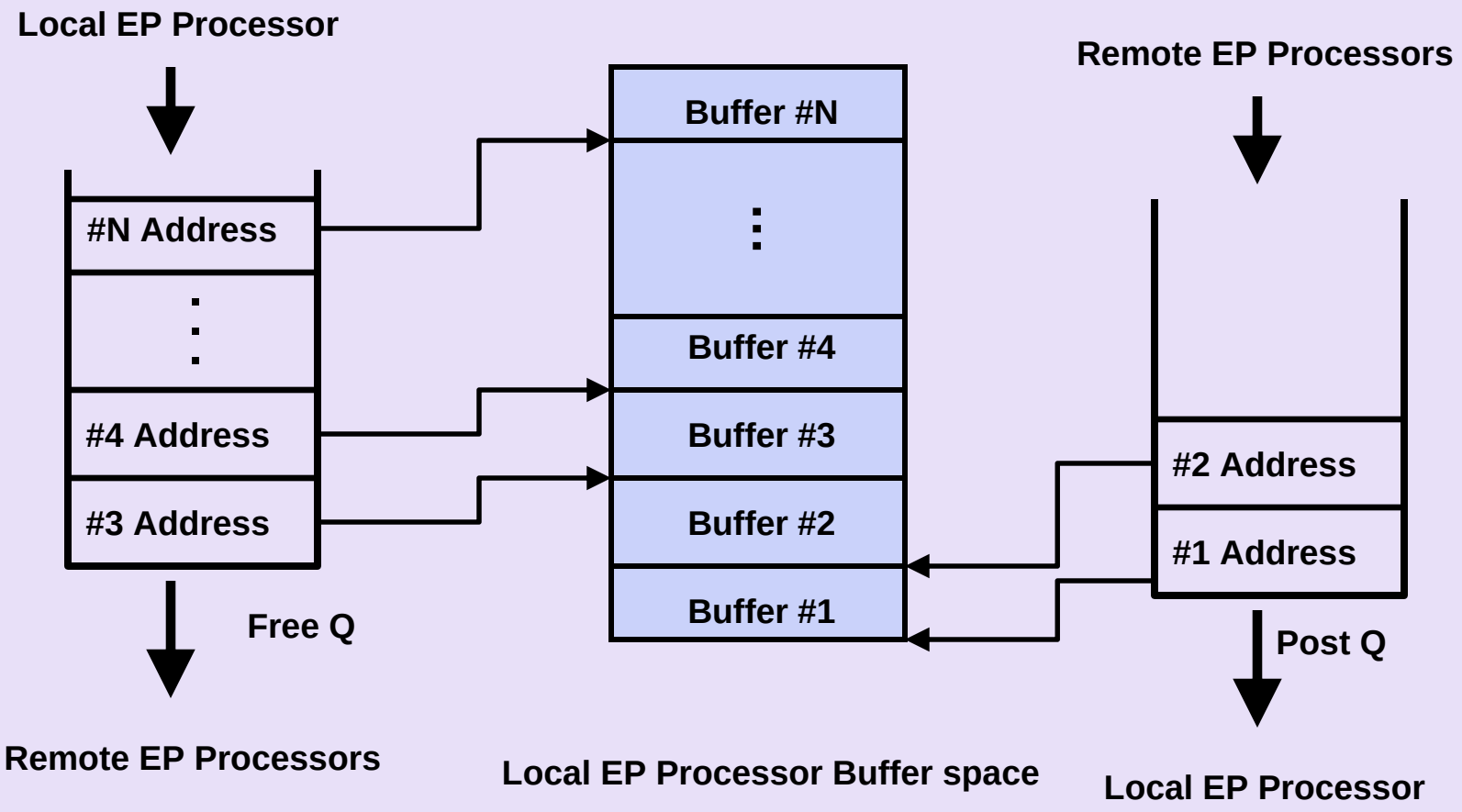


Queue and Buffer Configuration

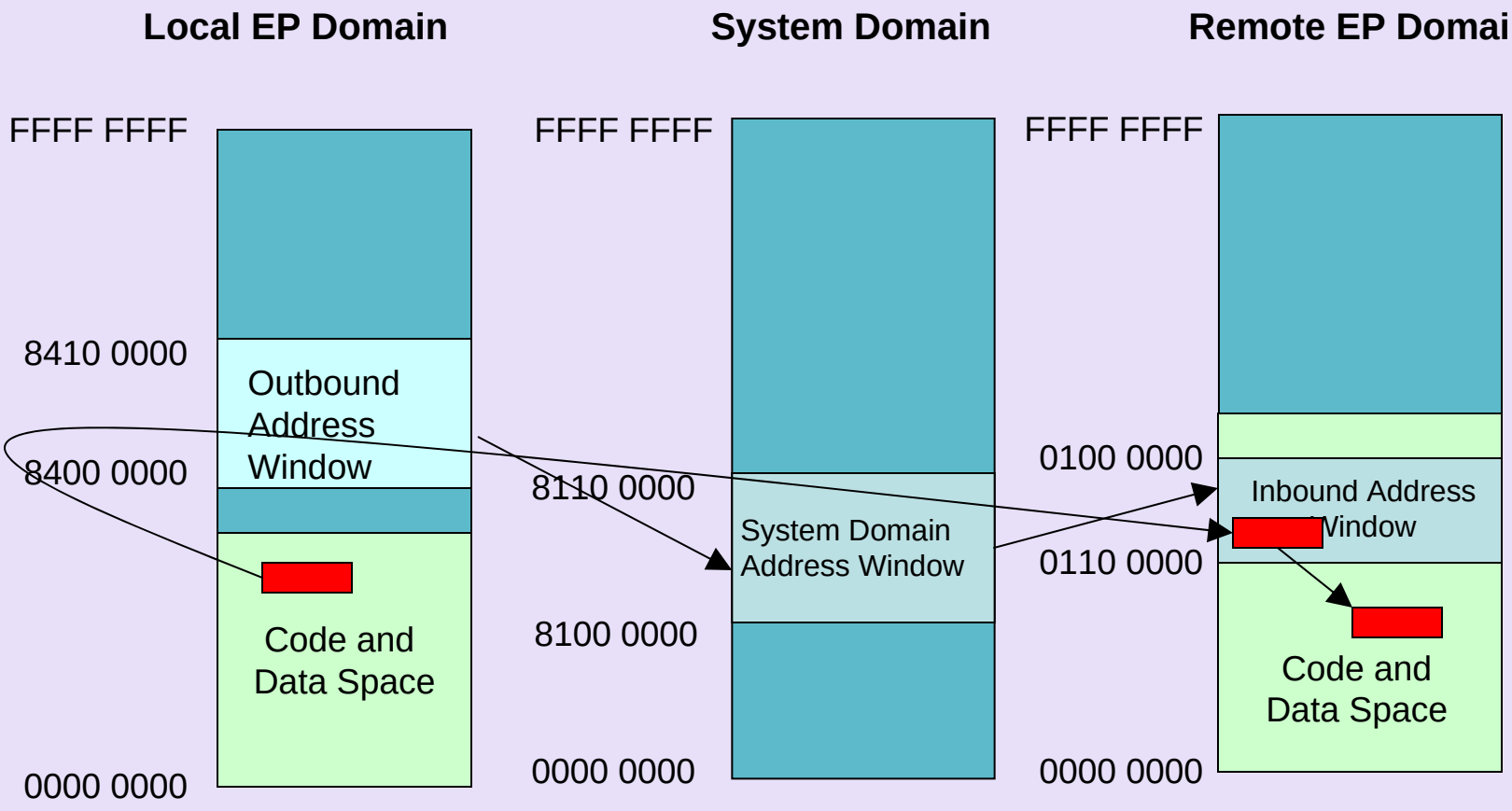


- Inbound translated address block is divided into multiple data buffer blocks
- Corresponding PCIe address of each block is written to the Free Queue by the endpoint processor

Queue Usage



Data Transfer Example



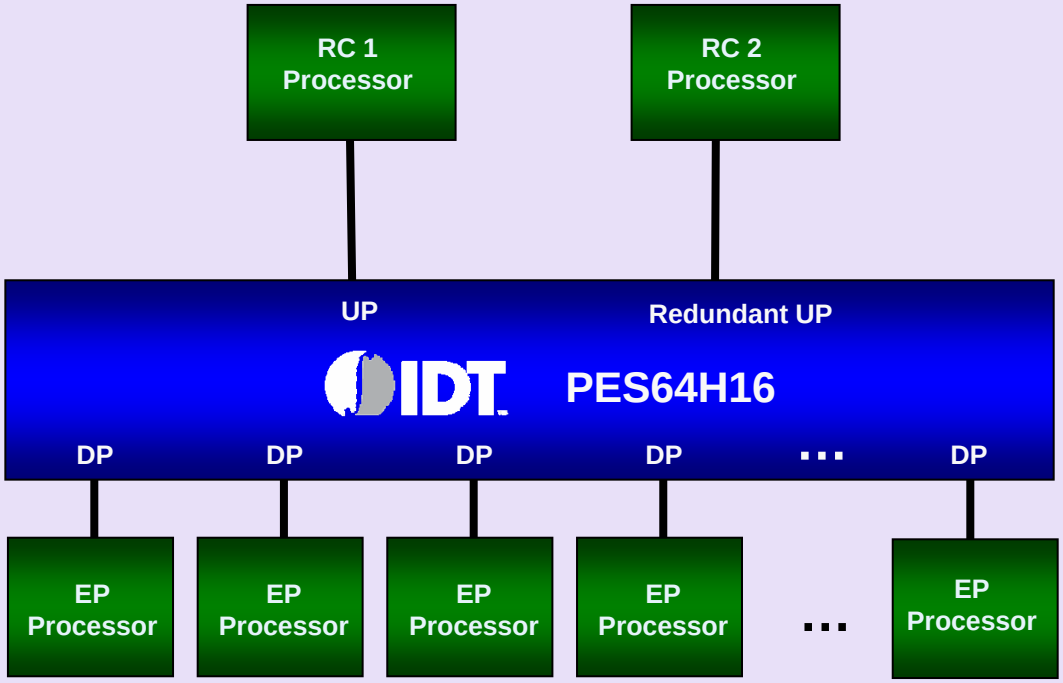
Interrupt and Error Handling

- Interrupt
  - ✓ Implicit (including INTx) interrupt handled by the RC processor
  - ✓ Endpoint processor can interrupt another endpoint processor using endpoint specific mechanism such as doorbell register or MSI
- Error Reporting
  - ✓ Error Message sent to the RC processor and then relayed to the endpoints as needed
  - ✓ Completion Status reported back to the requester

Redundancy

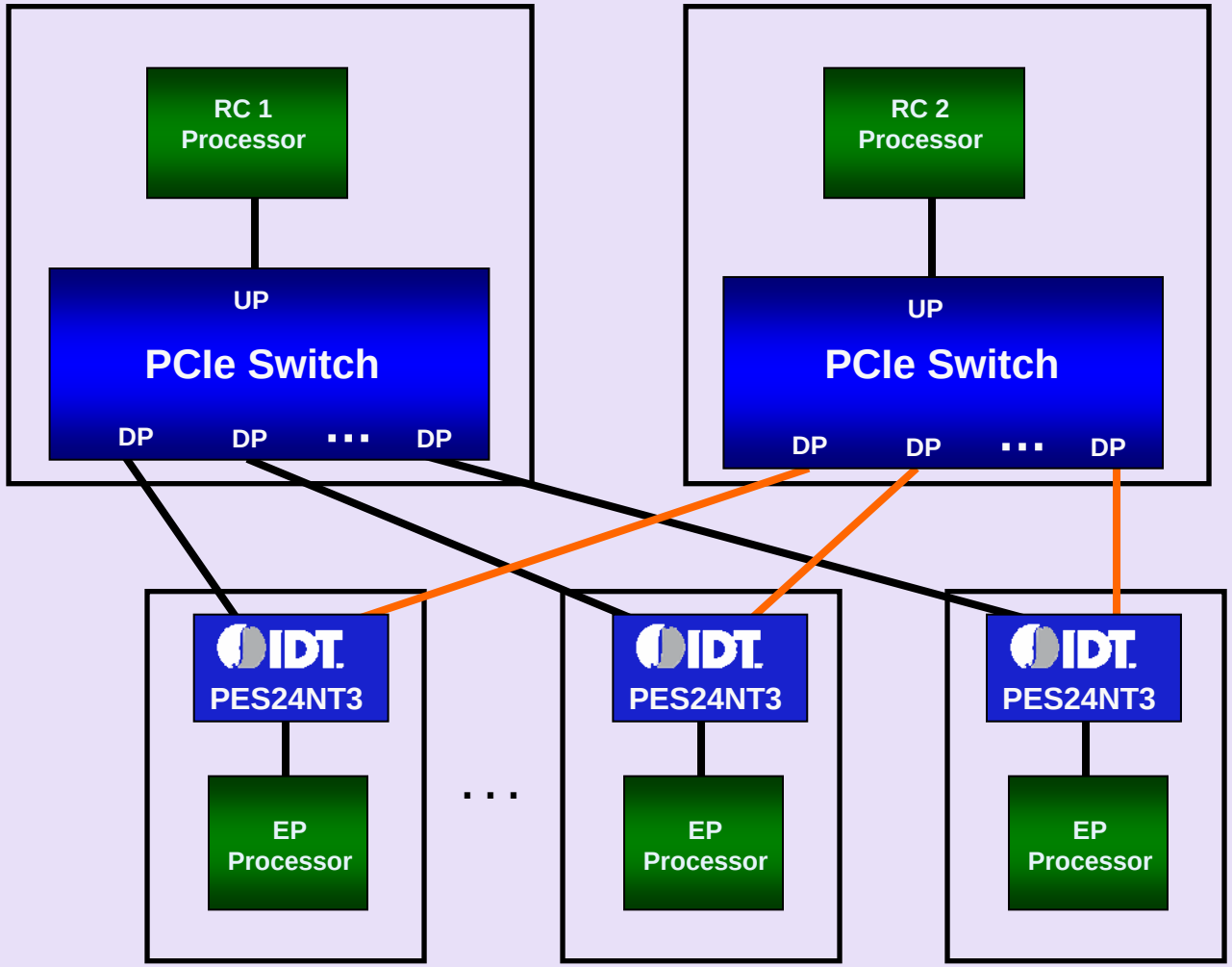
- Redundancy is important in systems where high availability is a concern
- Redundancy can be implemented in PCIe systems even though PCIe does not specifically address it
- Common redundancy topologies
  - ✓ Dual host
  - ✓ Dual star

Dual Host Redundancy Example



- Active and Standby RC processors
- Standby RC processor takes over after a failure in the active RC processor
- Redundant upstream port is a feature of the System Interconnect PCIe switch (e.g. PES64H16)

Dual Star Topology Example

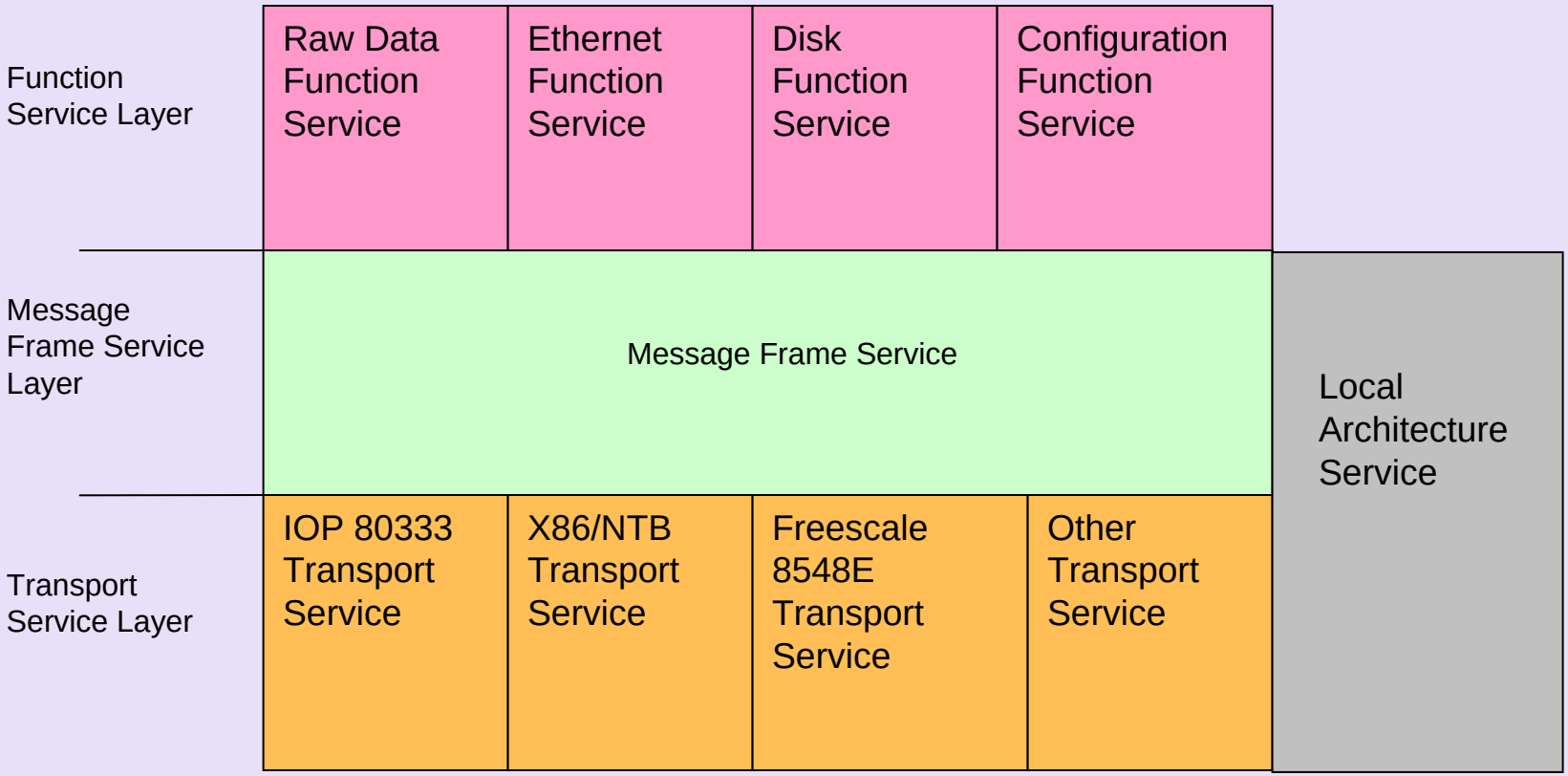


- Active and Standby PCIe switch as well as RC processors
- Inter-Domain Switch (e.g. PES24NT3) controls the path to the active PCIe switch
- No single point of interconnect failure with dual-star switching topology

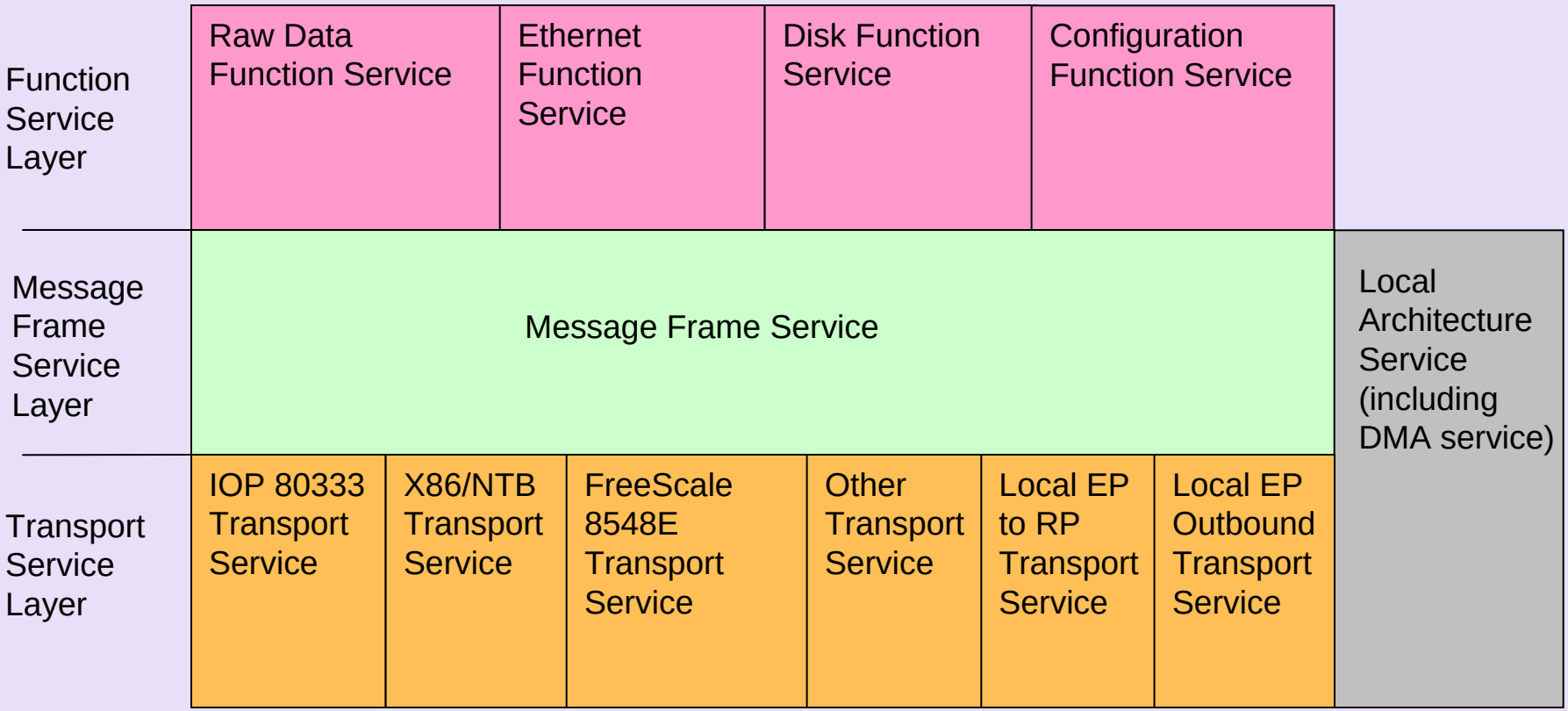
Software Components

- Software implemented as Linux device driver
- Endpoint processor discovery and initialization
- System and local domains address translation management
- Inter-processor communication protocol
- Message passing/Queuing architecture
- Data movement application
- Virtual Ethernet device driver
- Sharing of Ethernet interface on an endpoint processor

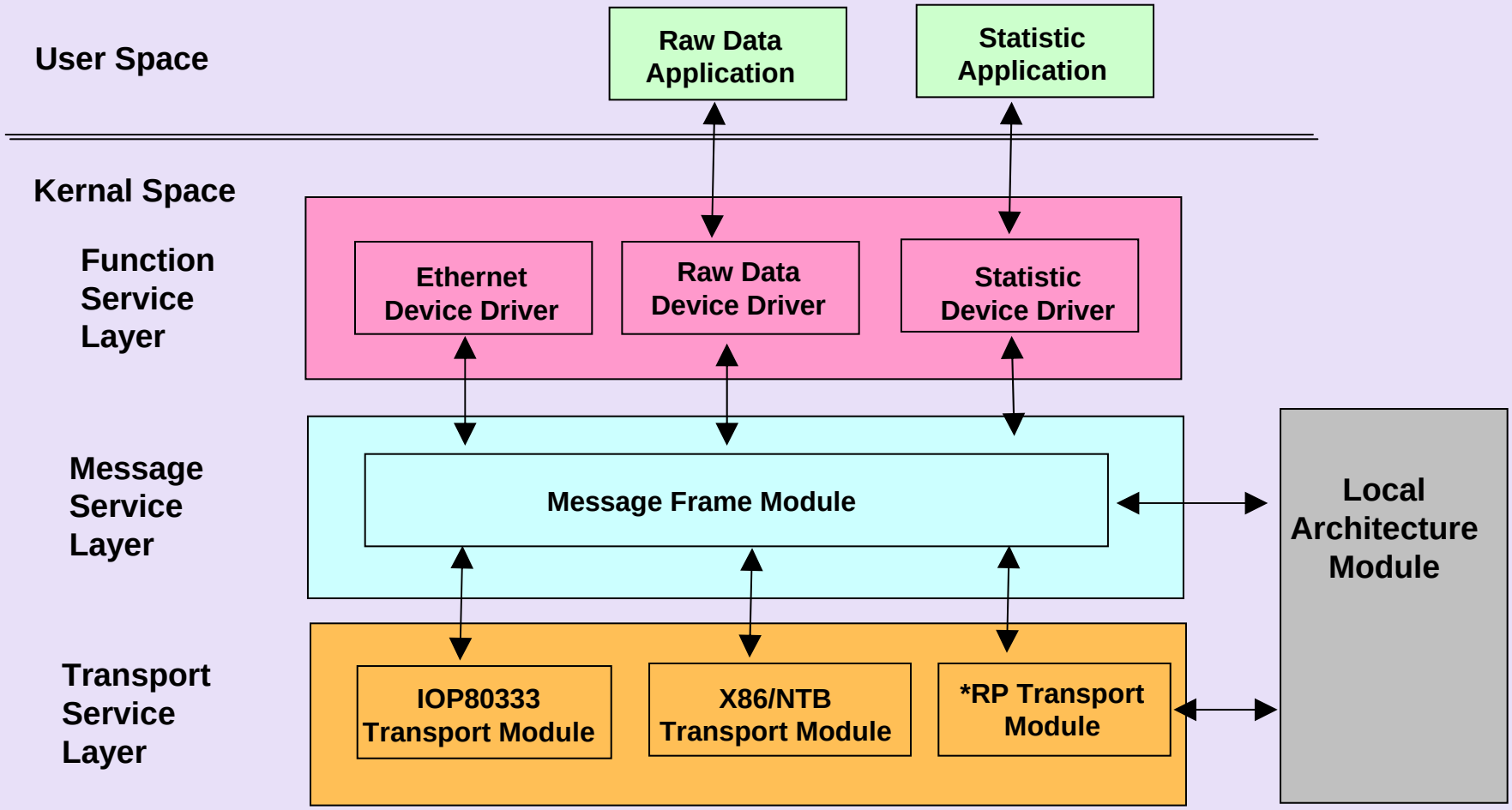
RC Processor Software Architecture



Endpoint Software Architecture

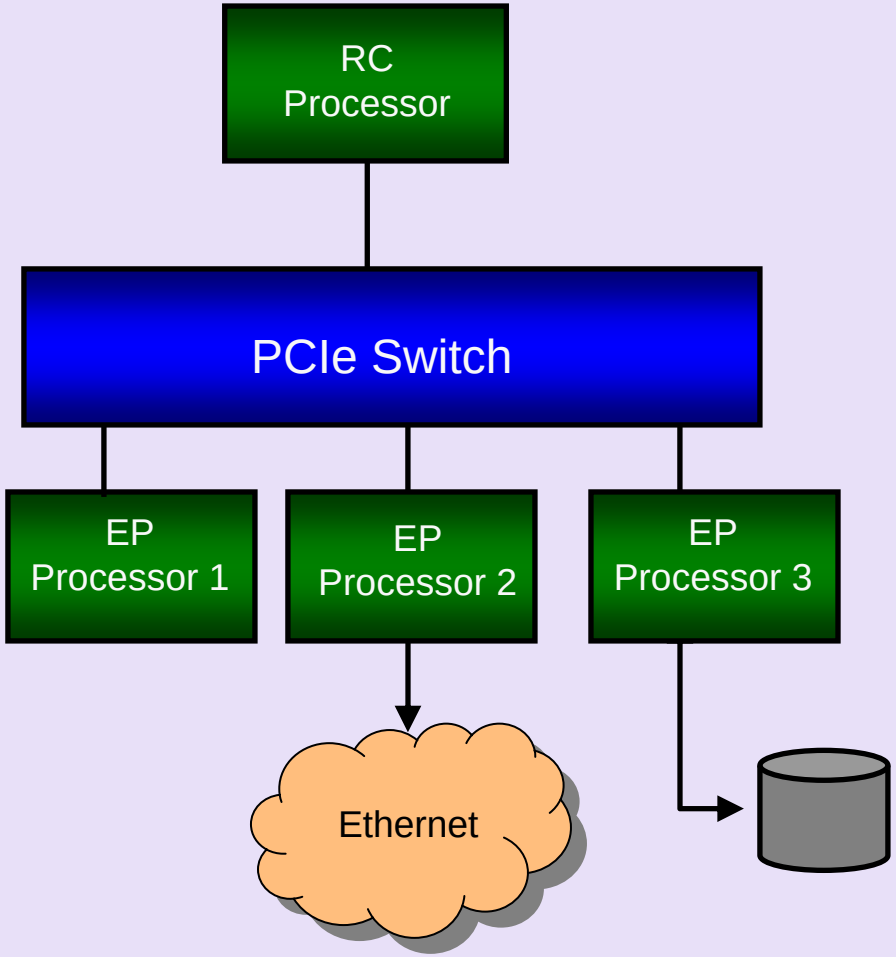


Software Modules



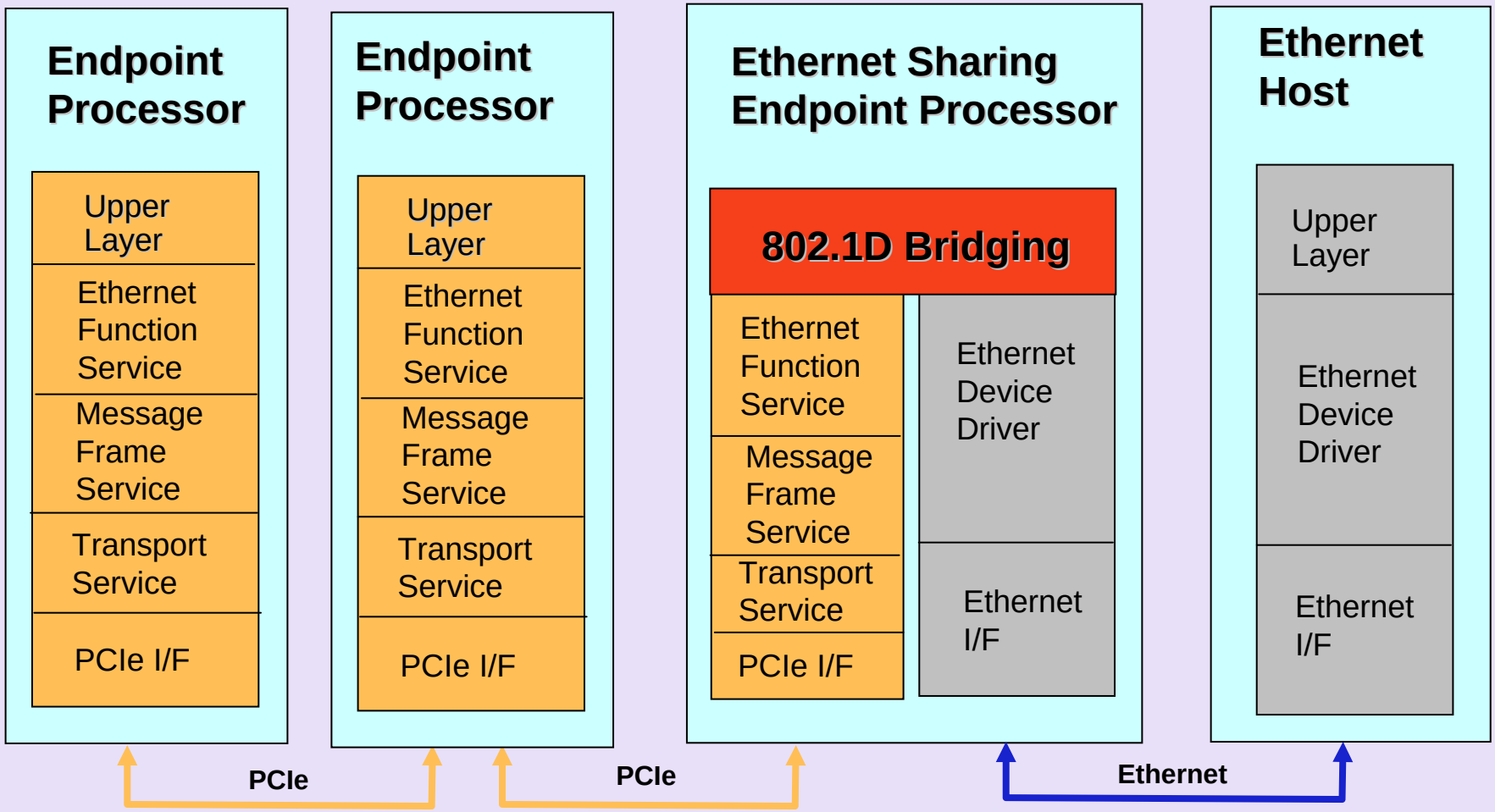
*RP Transport module
Exists in EP only

IO Sharing Application Examples

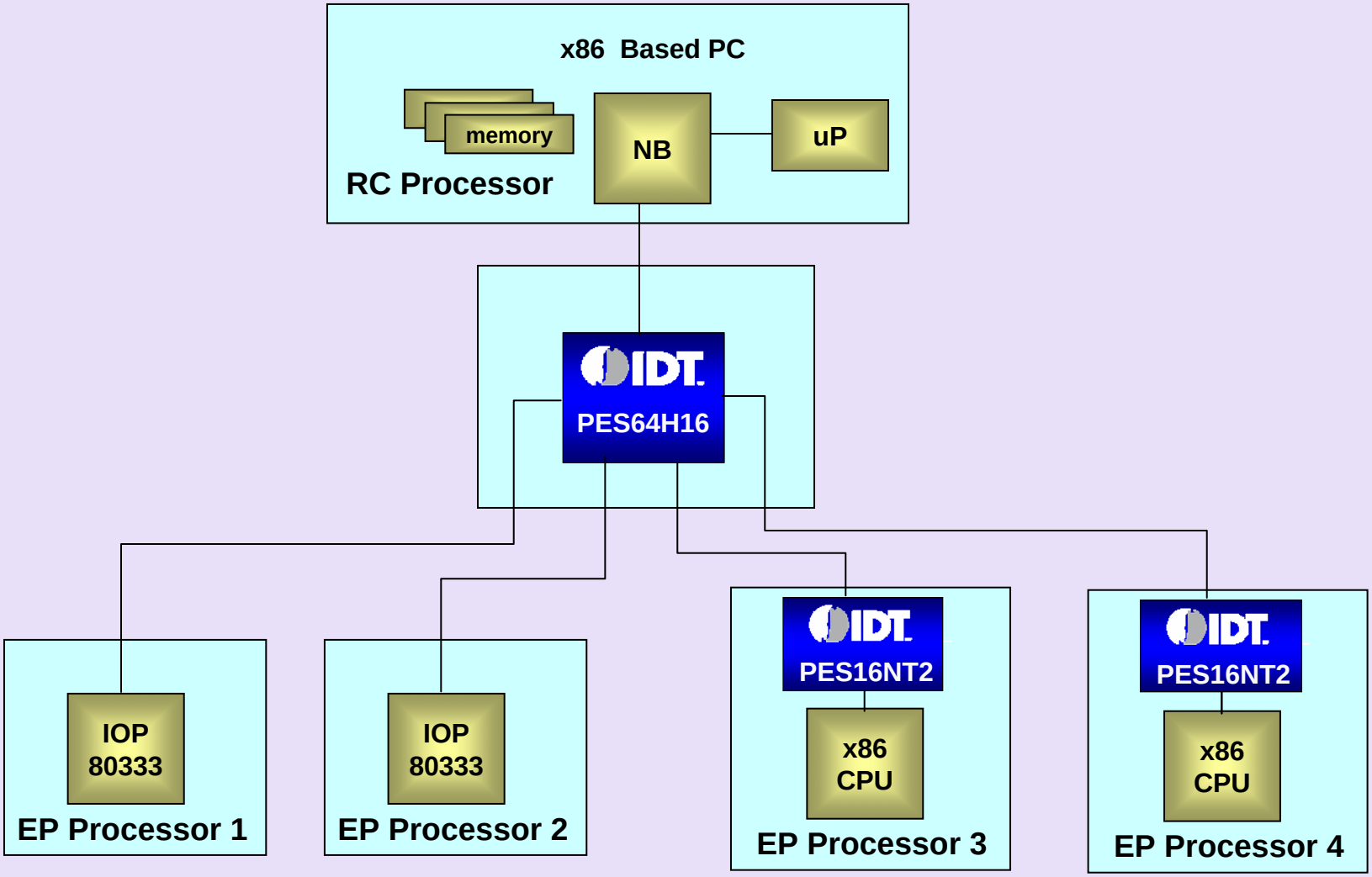


- RC Processor, Endpoint Processor 1 and 3 can share network interface on Endpoint Processor 2
- RC Processor Endpoint Processor 1 and 2 can share storage system on Endpoint Processor 3

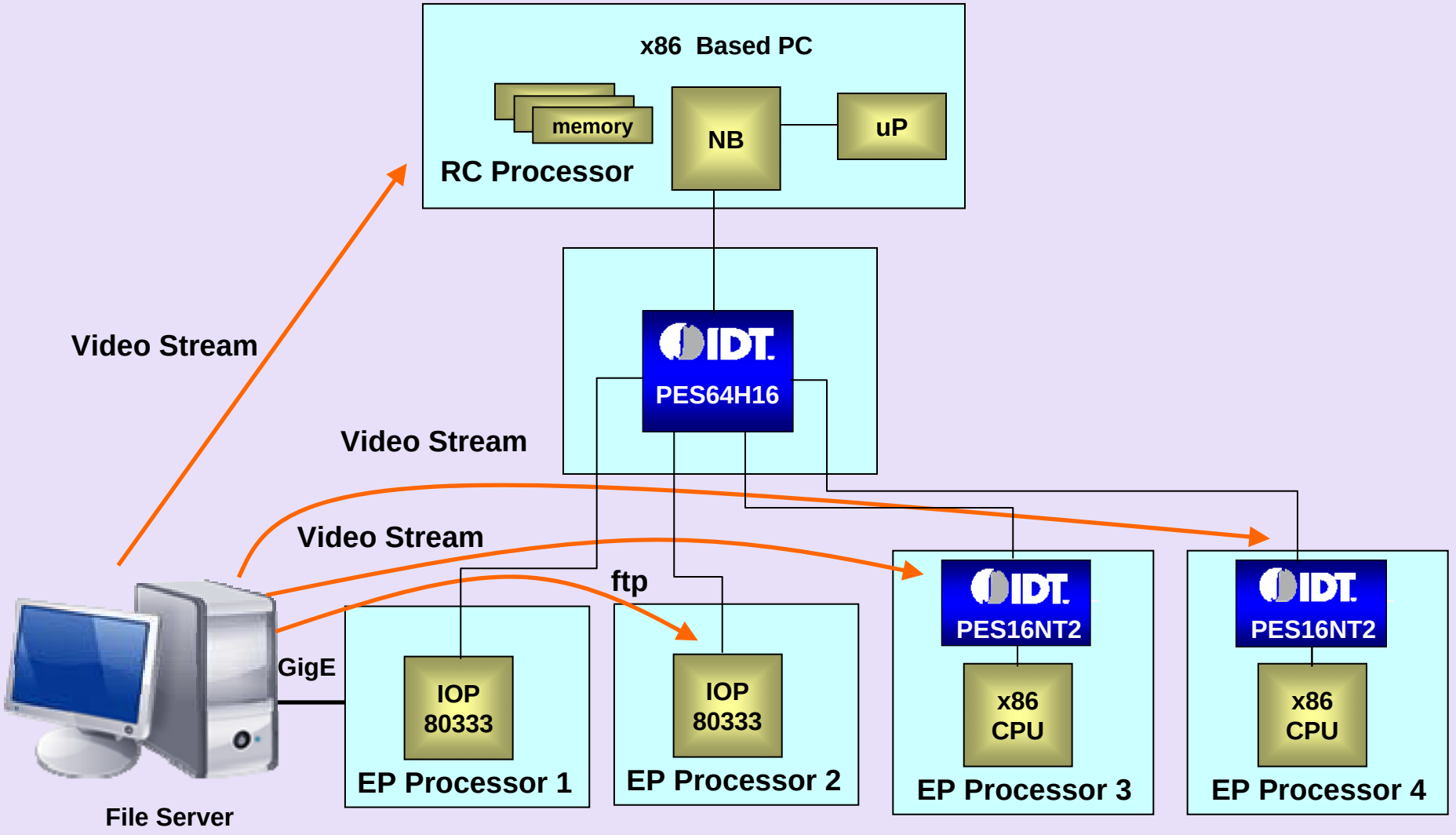
Ethernet Sharing Protocol Diagram



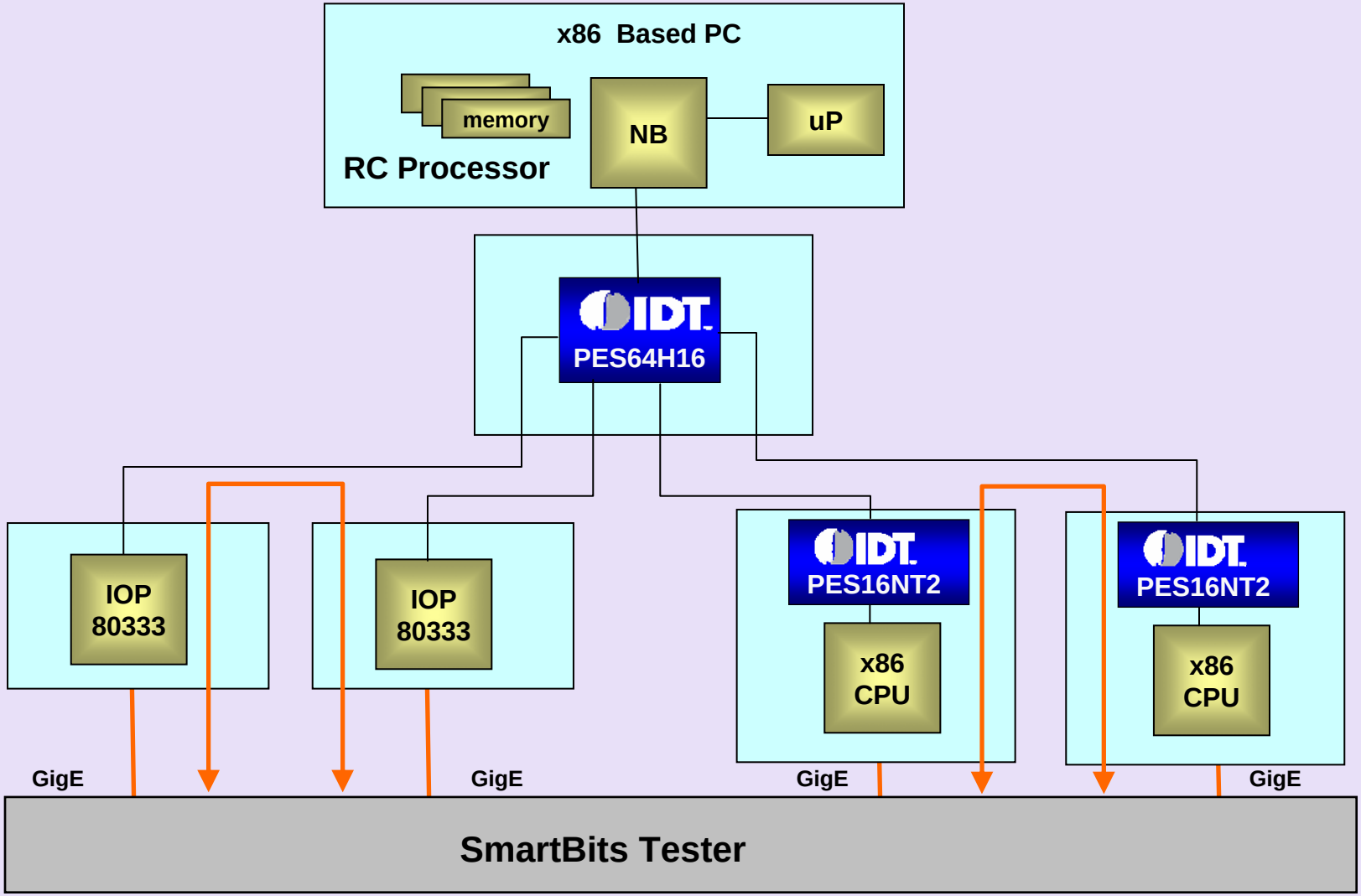
IDT Reference System Topology



Ethernet I/O Sharing Application



4-Port Ethernet Switch Application



Summary

- PCI Express forms the basis for a viable system interconnect solution for multiprocessor systems
  - ✓ Endpoint processors need to support address translation and queues for data passing
  - ✓ Standard PCIe initialization and enumeration procedures
- Experience with PCIe as a system interconnect
  - ✓ Platform: standard PCIe switch for system interconnect, PCIe Inter-Domain Switch, multiple endpoint processors (including IOP80333, x86)
  - ✓ Features implemented include I/O sharing across multiple processors

Thank you for attending the
PCI-SIG Developers Conference 2007.

For more information please go to
www.pcisig.com