PCIe® 3.0/2.1 Protocol Update

Mahesh Wagh

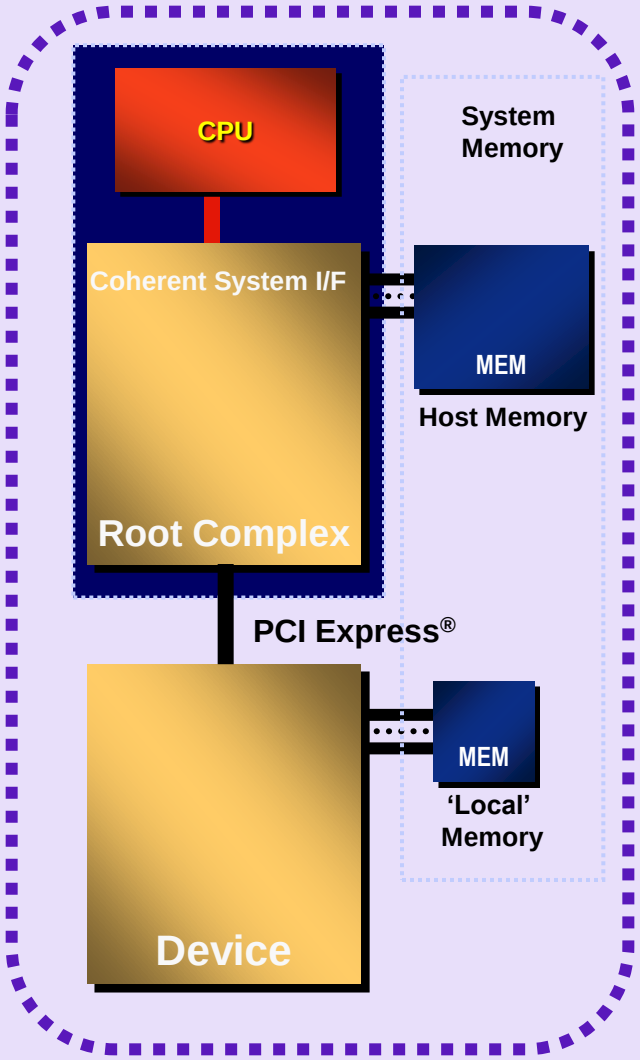
Intel Corporation

Protocol Extensions Update: Today's Presentation

- Protocol Extension ECNs Categorized
- Protocol Extension ECN Summary & Status
- Atomic Operations (AtomicOps)
- Multicast
- TLP Processing Hints (TPH)
- Dynamic Power Allocation (DPA)
- Latency Tolerance Reporting (LTR)
- Optimized Buffer Flush & Fill (OBFF)
- ASPM Optionality
- TLP Prefix

Protocol Extension ECNs Categorized

***Details in Today's Presentation**



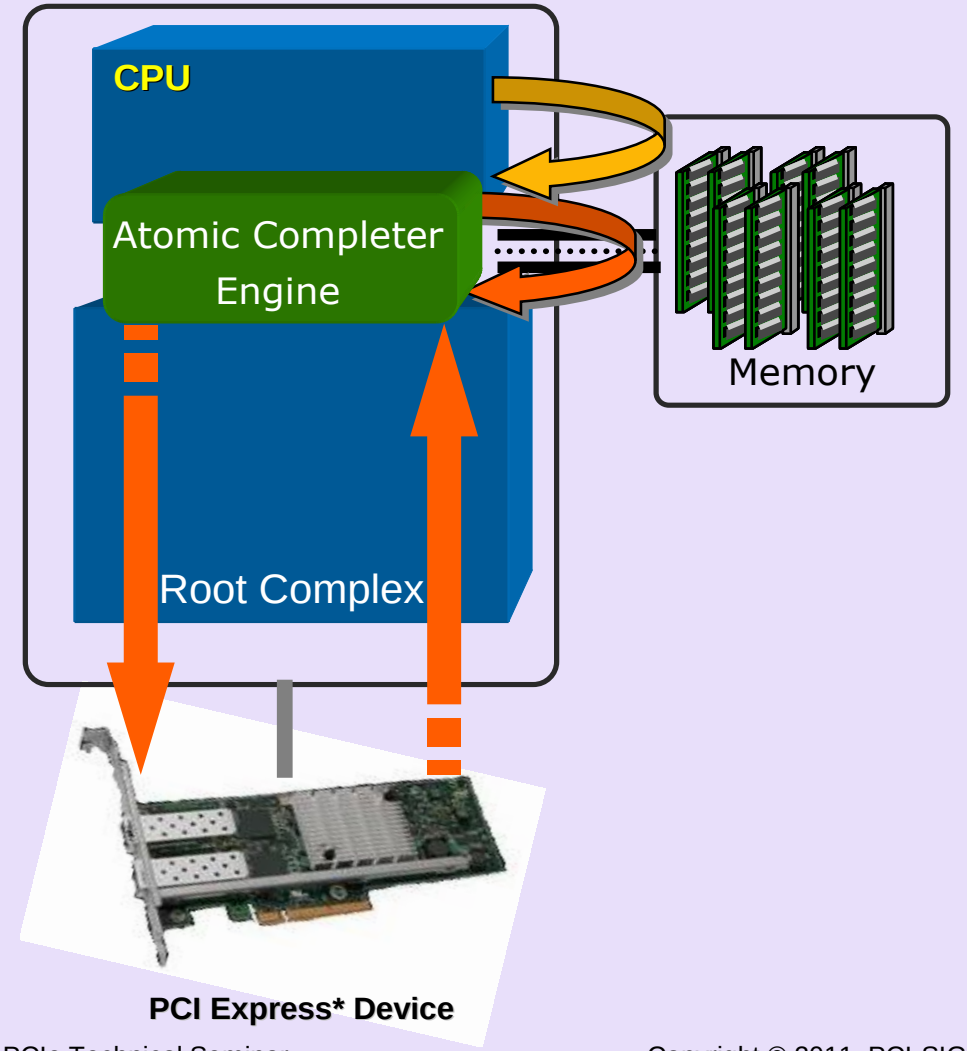
- Performance Improvements
  - ✓ ***TLP Processing Hints** – hints to optimize system resources and performance
  - ✓ ***TLP Prefix** – mechanism to extend TLP headers for TLP Processing Hints, MR-IOV, and future extensions
  - ✓ **ID-Based Ordering** – Transaction-level attribute/hint to optimize ordering within the RC and PCIe hierarchy
  - ✓ **Extended Tag Enable Default** – permits the default for the Extended Tag Enable bit to be implementation-specific
- Software Model Improvements
  - ✓ ***Atomic Operations** – new atomic transactions to reduce synchronization overhead
  - ✓ ***Page Request Interface*** *– mechanism in ATS 1.1 for device to request absent pages to be made available (IOV Work Group)*
- Communication Model Enhancements
  - ✓ ***Multicast** – mechanism to transfer common data or commands sent from one source to multiple recipients
- Power Management
  - ✓ ***Dynamic Power Allocation** – support for dynamic power operational modes through a standard configuration mechanism
  - ✓ ***Latency Tolerance Reporting** – Endpoints report service latency requirements for improved platform power management
  - ✓ ***Optimized Buffer Flush/Fill** – Mechanism for devices to synchronize DMA activity for improved platform power mgmt
  - ✓ ***ASPM Optionality** – ASPM enhanced to support full matrix of L0s and L1; L0s no longer required for all components
- Configuration Enhancements
  - ✓ **Resizable BAR** – Mechanism to support BAR size negotiation
  - ✓ **Internal Error Reporting** – Extend AER to report component internal errors and record multiple error logs

Protocol Extension ECN Summary & Status

Extension	Description	Status
Atomic Operations (AtomicOps)	Atomic Read-Modify-Write mechanism	In the PCIe 2.1 spec
Internal Error Reporting	Extend AER to report component internal errors and record multiple error logs	In the PCIe 2.1 spec
Resizable BAR	Mechanism to support BAR size negotiation	In the PCIe 2.1 spec
Multicast	Address-Based Multicast of Posted Request TLPs	In the PCIe 2.1 spec
ID-Based Ordering (IDO)	New type of relaxed ordering semantics to improve performance	In the PCIe 2.1 spec
Dynamic Power Allocation (DPA)	Dynamic power mgmt for substates of D0 (active state)	In the PCIe 2.1 spec
Latency Tolerance Reporting (LTR)	Endpoints report service latency requirements, enabling improved platform power mgmt	In the PCIe 2.1 spec
Extended Tag Enable Default	Permits default for Extended Tag Enable bit to be Function-specific instead of 0b	In the PCIe 2.1 spec
TLP Processing Hints (TPH)	Hints for optimized TLP processing within host memory/cache hierarchy	In the PCIe 2.1 spec
TLP Prefix	Mechanism to extend TLP headers	In the PCIe 2.1 spec
Optimized Buffer Flush/Fill (OBFF)	Mechanisms for devices to synchronize DMA activity for improved platform power mgmt	Finalized 5/2009 (post 2.1)
ASPM Optionality	ASPM now supports full matrix of L0s and L1; L0s no longer required for all components	Finalized 9/2009 (post 2.1)

Atomic Operations (AtomicOps)

Synchronization



Atomic Read-Modify-Write

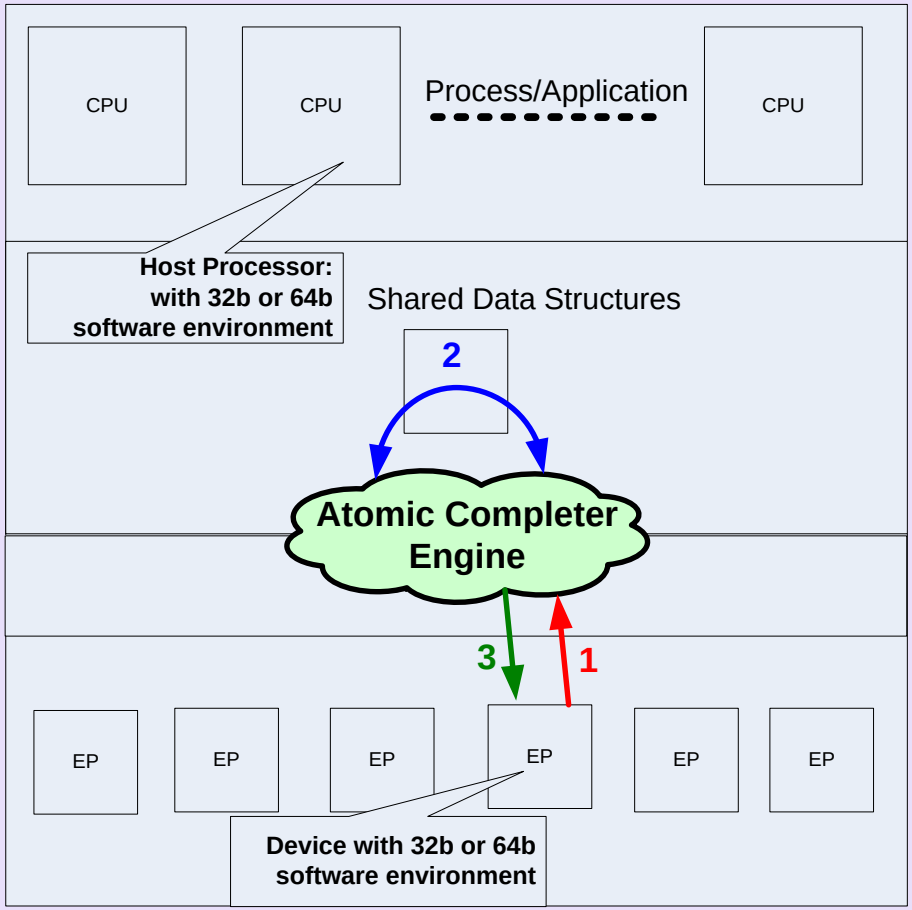
- Atomic transaction support for Host update of main memory exists today
  - Useful for synchronization without interrupts
  - Rich library of proven algorithms in this area
- Benefit in extending existing inter-processor primitives for data sharing/synchronization to PCI Express* (PCIe*) interconnect domain
  - Low overhead critical sections
  - Non-Blocking algorithms for managing data structures e.g. Task lists
  - Lock-Free Statistics e.g. counter updates
- Improve existing application performance
  - Faster packet arrival rates create demand for faster synchronization
- Emerging applications benefit from Atomic RMW
  - Multiple Producer – Multiple Consumer support
  - Example: Math, Visualization, Content Processing etc

Atomic Operations

AtomicOp	Description
FetchAdd	Data(Addr) = Data(Addr) + AddData
Swap	Data(Addr) = SwapData
CAS (Compare & Swap)	If (CompareData == Data(Addr)) then Data(Addr) = SwapData

- Each AtomicOp returns initial Data(Addr)
- Operation sizes supported
  - ✓ 32b and 64b operation sizes for FetchAdd and Swap
  - ✓ 32b, 64b, and 128b operation sizes for CAS
- AtomicOp address must be naturally aligned to operation size
  - ✓ AtomicOp data access guaranteed to not cross cacheline boundaries

AtomicOps Mechanism



1. AtomicOp Request
   - ✓ Non-Posted Request with Data
   - ✓ FetchAdd, Swap, or CAS
   - ✓ Multiple outstanding AtomicOps supported
   - ✓ Requestor has to allocate space for Completion before issuing each AtomicOp Request
2. Atomic Operation Execution
   - ✓ Atomic Completer Engine performs atomic read-modify-write operation
     - Read initial value
     - Compute and write new value
3. AtomicOp Completion
   - ✓ Completion returns initial value
   - ✓ "Standard" Completion for Non-Posted Request

***Device to System memory shown in figure for illustration purposes**

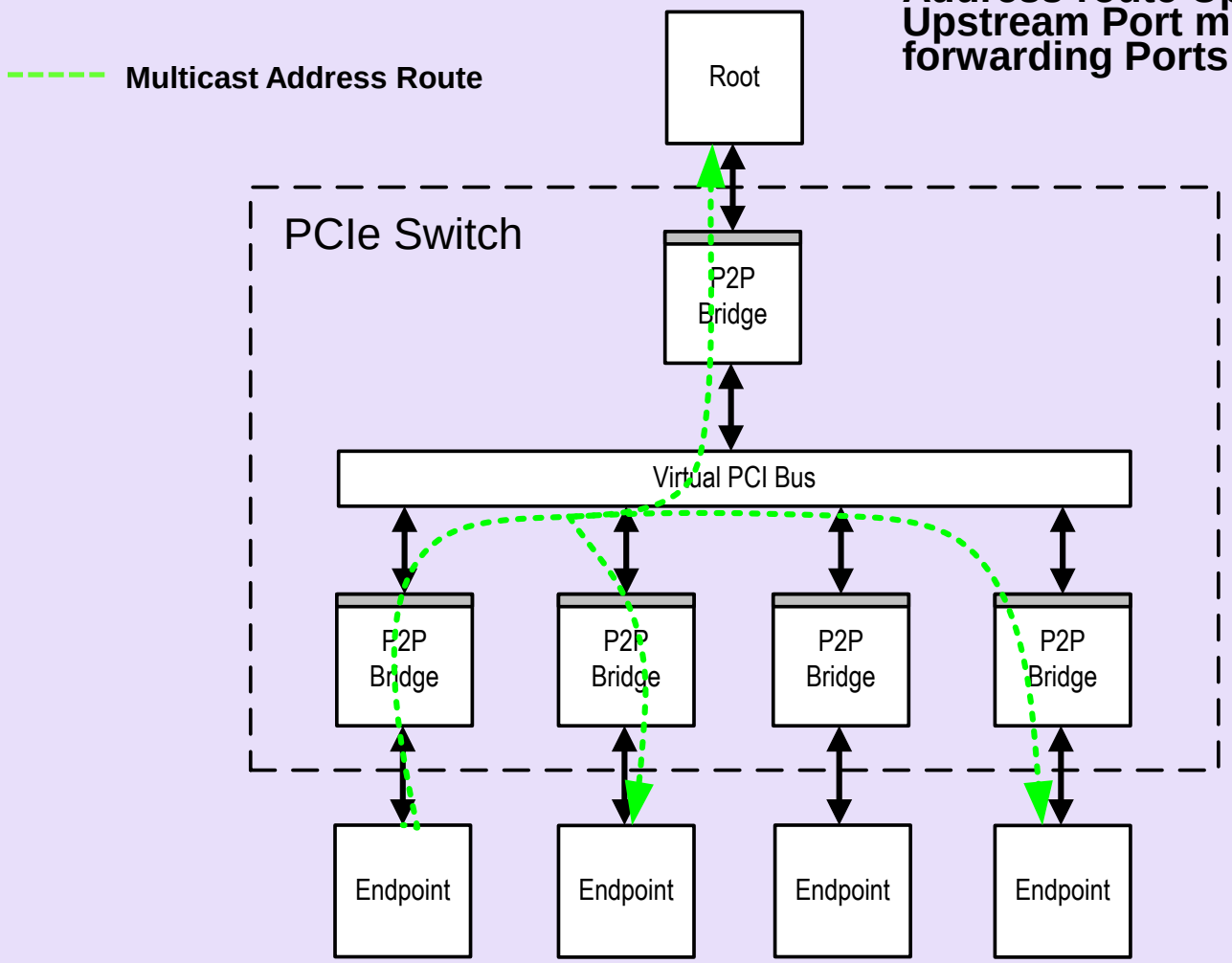
Multicast

Multicast Motivation & Mechanism Basics

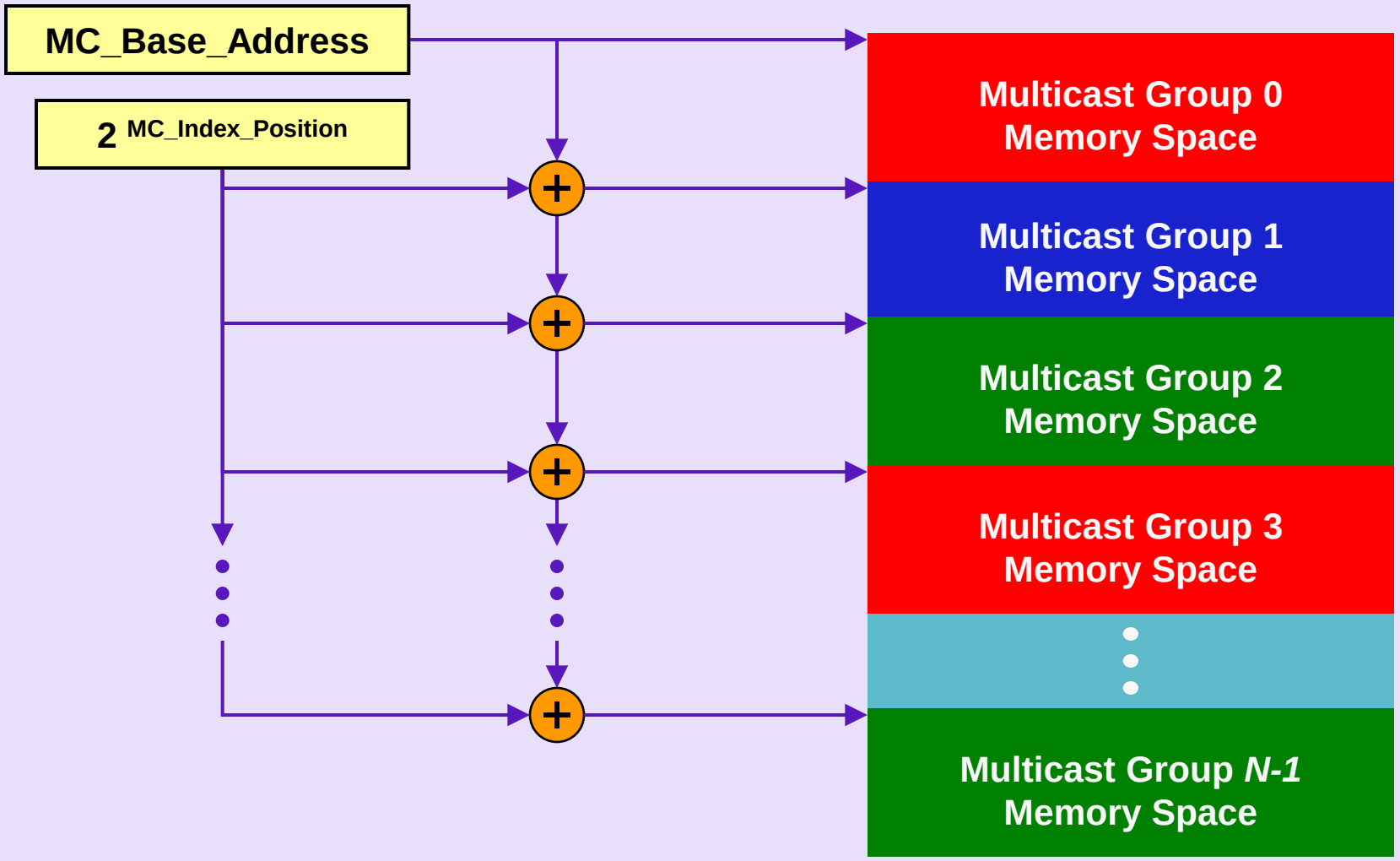
- Several key applications benefit from Multicast
  - ✓ Communications backplane (e.g. route table updates, support of IP Multicast)
  - ✓ Storage (e.g., mirroring, RAID)
  - ✓ Multi-headed graphics
- PCIe architecture extended to support address-based Multicast
  - ✓ New Multicast BAR to define Multicast address space
  - ✓ New Multicast Capability structure to configure routing elements and Endpoints for Multicast address decode and routing
  - ✓ New Multicast Overlay mechanism in Egress Ports allow Endpoints to receive Multicast TLPs without requiring Endpoint Multicast Capability structure
- Supports only Posted, address-routed transactions (e.g., Memory Writes)
  - ✓ Supports both RCs and EPs as both targets and initiators
  - ✓ Compatible with systems employing Address Translation Services (ATS) and Access Control Services (ACS)
  - ✓ Multicast capability permitted at any point in a PCIe hierarchy

Multicast Example

Address route Upstream – Upstream Port must be part of the forwarding Ports for Multicast



Multicast Memory Space



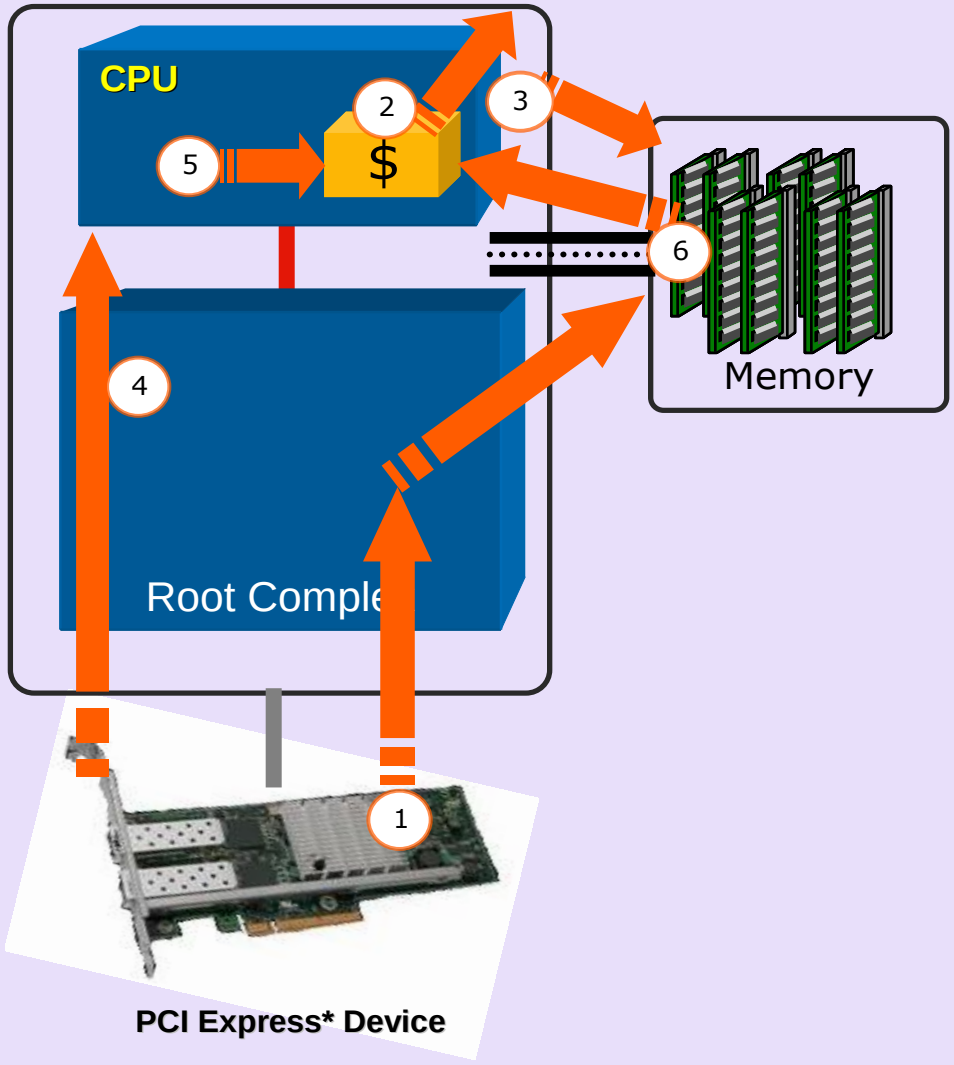
Multicast Overlay Mechanism

- Mechanism allows an Egress Port to remap the address of an outgoing MC TLP, effectively overlaying Multicast Memory Space on top of a target unicast address range
  - ✓ In a Downstream Port, this enables the MC TLP to target (unicast) Memory Space accepted by an Endpoint that lacks a Multicast Capability structure
  - ✓ At a Switch Upstream Port, Multicast space can be remapped onto a Memory Space range associated with Host Memory
- Due to the address being modified, ECRC (if present) must either be stripped or regenerated
  - ✓ ECRC rules for Multicast Overlay are as shown in table below

MC_Overlay Enabled	TLP has ECRC	ECRC Regeneration Supported	Action if ECRC Check Passes	Action if ECRC Check Fails
No	x	x	Forward TLP unmodified	
Yes	No	x	Forward modified TLP	
Yes	Yes	No	Forward modified TLP with ECRC stripped	
Yes	Yes	Yes	Forward modified TLP with regenerated ECRC	Forward modified TLP with inverted regenerated ECRC

TLP Processing Hints (TPH)

Basic Device Writes



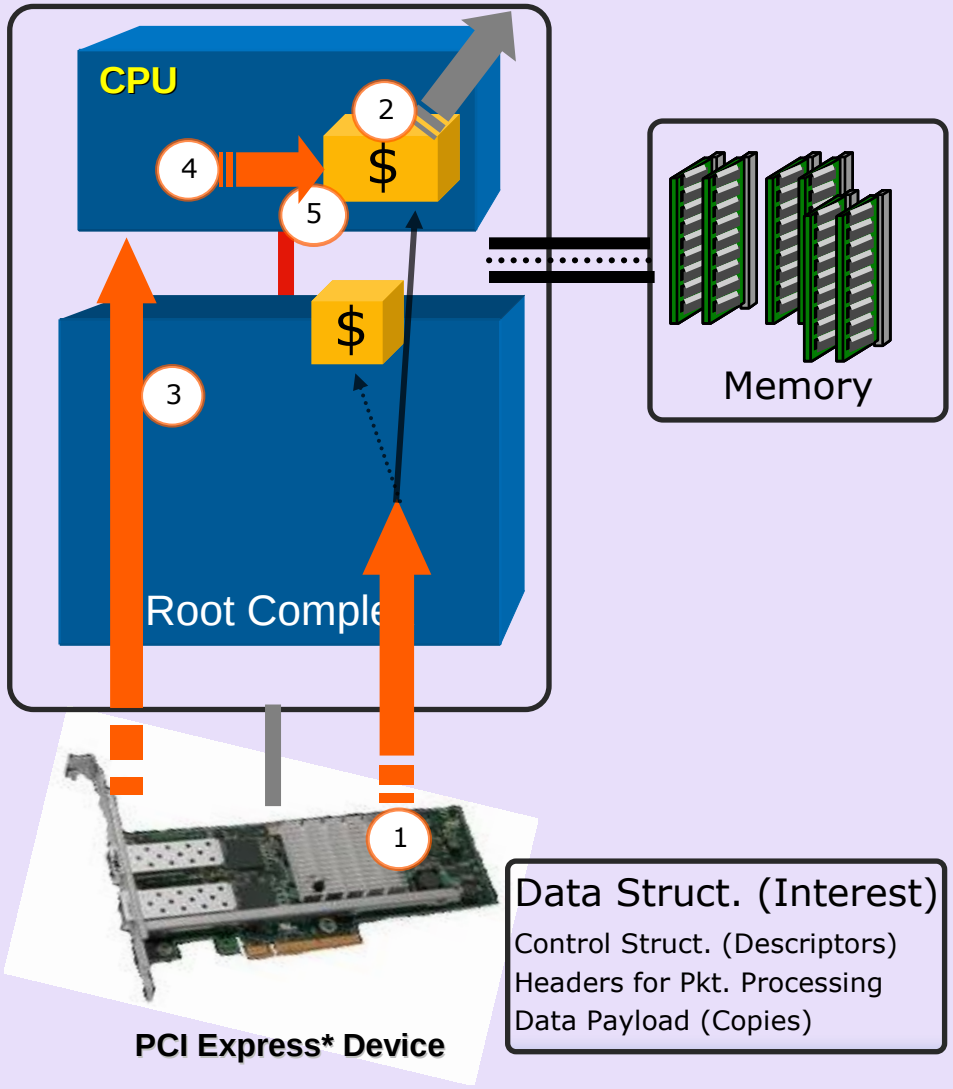
Device Writes Host Reads

1. Device Writes DMA Data
2. Snoop System Caches
3. Write Back (Memory)
   Device Write (Memory)
4. Notify Host (Optional)
5. Software Reads DMA Data
6. Host Read Completed

Transaction Flow does not take full advantage of System Resources

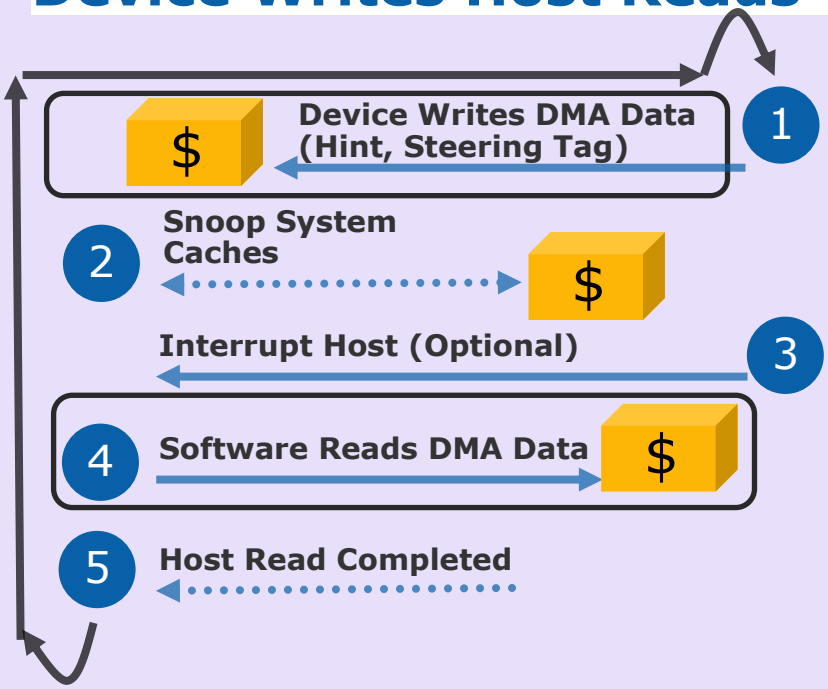
- System Caches
- System Interconnect

Device Writes with TPH



Data Struct. (Interest)
Control Struct. (Descriptors)
Headers for Pkt. Processing
Data Payload (Copies)

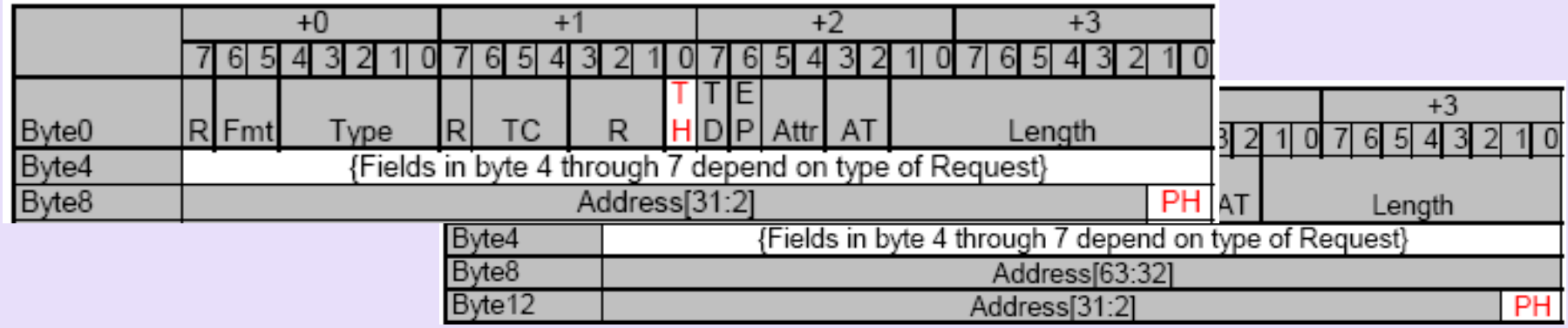
Device Writes Host Reads



Effective Use of System Resources

- Reduce Access latency to system memory
- Reduce Memory & system interconnect BW & Power

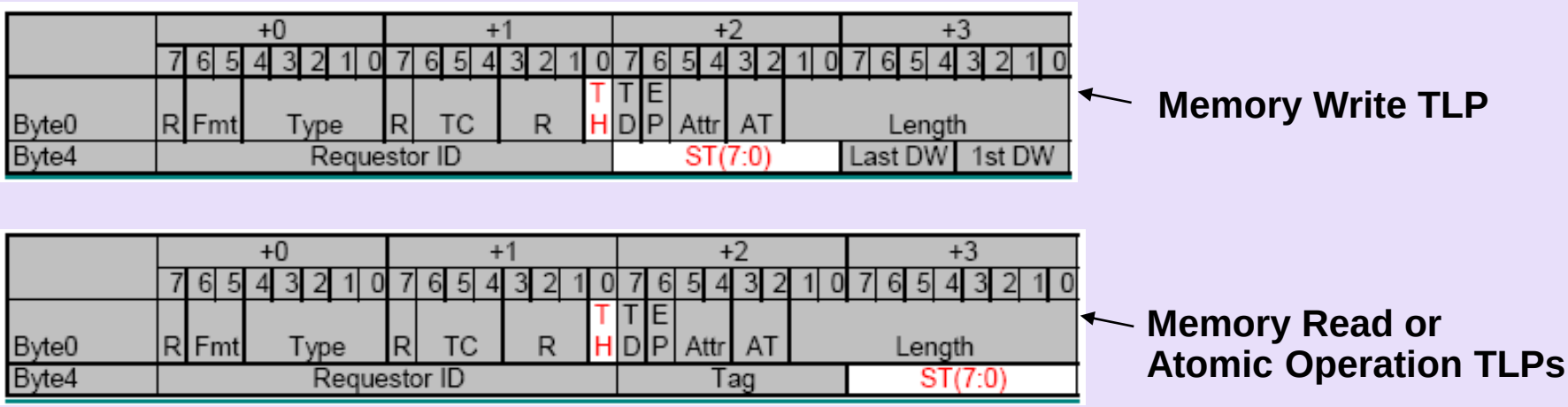
TPH Mechanism



- Mechanism to provide processing hints on per TLP basis for Requests that target Memory Space
  - ✓ Enable system hardware (ex: Root-Complex) to optimize on a per TLP basis
  - ✓ Applicable to Memory Read/Write and Atomic Operations

PH[1:0]	Processing Hint	Usage Model
00	Bi-directional data structure	Bi-Directional data structure
01	Requestor	D*D*
10	Target	DWHR HWDR
11	Target with Priority	DWHR (Prioritized) HWDR (Prioritized)

Steering Tag (ST)



- ST: 8 bits defined in header to carry System specific Steering Tag values
  - ✓ Use of Steering Tags is optional – 'No preference' value used to indicate no steering tag preference
  - ✓ Architected Steering Table for software to program system specific steering tag values

TPH Summary

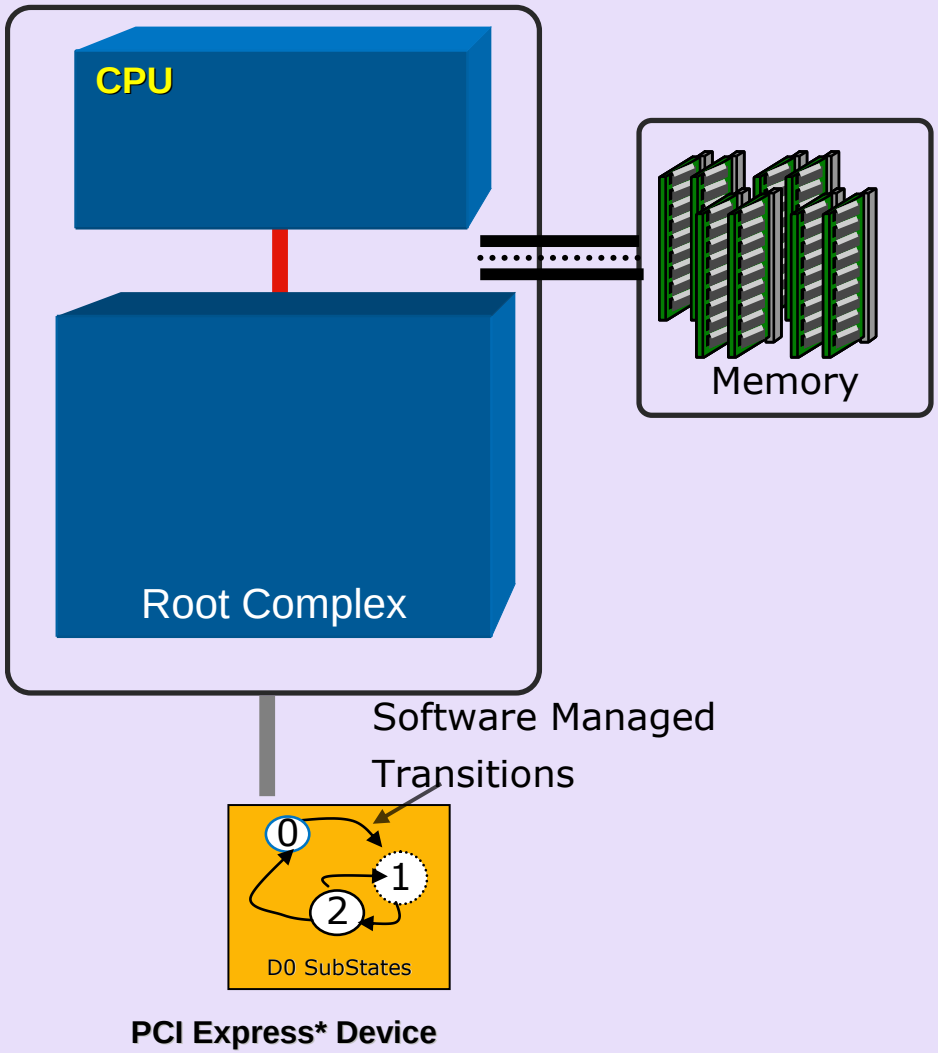
- Mechanism to make effective use of system fabric and improve system efficiency
  - ✓ Reduce variability in access to system memory
  - ✓ Reduce memory & system interconnect BW & power consumption
- Ecosystem Impact
  - ✓ Software impact is under investigation – minimally may require software support to retrieve hints from system hardware
  - ✓ Endpoints take advantage only as needed → No cost if not used
  - ✓ Root Complex can make implementation tradeoffs
  - ✓ Minimal impact to Switches
- Architected software discovery, identification, and control of capabilities
  - ✓ RC support for processing hints
  - ✓ Endpoint enabling to issue hints

Dynamic Power Allocation (DPA)

Motivation

- Current PCIe provides standard Device & Link-level Power Management
- PCIe 2.0 adds mechanisms for dynamic scaling of Link width/speed
- Devices are increasingly higher consumers of system power & thermal budget
- No architected mechanism for dynamic control of device thermal/power budgets
- New PCIe power management mechanisms to complement support on-going industry wide efforts to optimize component and platform power management to meet new customer and regulatory operating requirements

Dynamic Power Allocation (DPA)

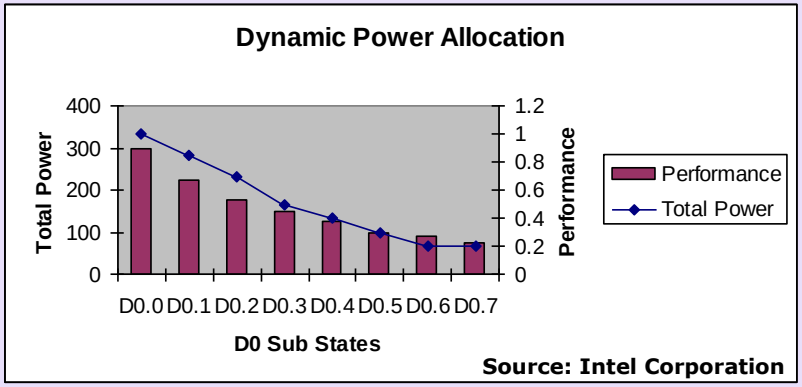


DPA Capability

- Extend Existing PCI Device PM to provide Active (D0) substates
  - Up to 32 substates supported
- Dynamic Control of D0 Active Substates

Benefits

- Platform Cost Reduction
  - Pwr/Thermal Management
- Platform Optimizations
  - Battery Life (Mobile)/Power(Servers)



Latency Tolerance Reporting (LTR)

Reducing Platform Power through Latency Tolerance Reporting

- Problem: Current platform PM policies guesstimate when devices are idle (e.g. w/inactivity timers)
  - ✓ Guessing wrong can cause performance issues, or even HW failures
  - ✓ Worst case: PM disabled to allow functionality at cost to power
  - ✓ Even best case not good – reluctance to power down leaves some PM opportunities on the table
    - Tough balancing act between performance / functionality and power

Wanted: Mechanism for platform to tune PM based on actual device service requirements

Latency Tolerance Reporting (LTR)

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte0	R	Fmt		Type					R	TC			Reserved				TD	EP	Attr		R				Length							
Byte4	Requester ID																Tag								Message Code							
Byte8	Reserved																Reserved															
Byte12	No-Snoop Latency																Snoop Latency															

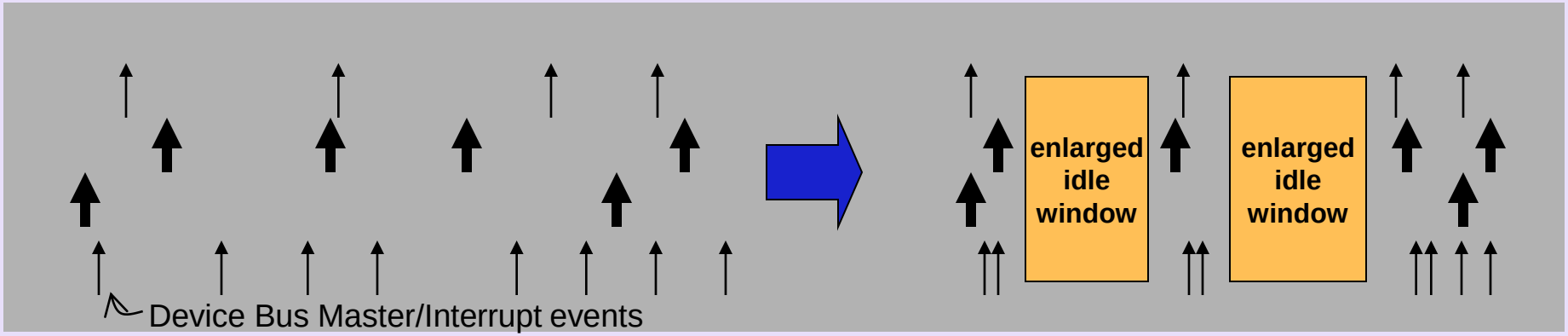
- Concept: Msg for device service latency reporting
  - ✓ PCIe Message sent by Endpoint with required service latency
    - Capability to report both snooped & non-snooped values
  - ✓ “Terminate at Receiver” routing
    - Multi-Function devices & Switches coalesce Messages from below & send single Message up
- Provides device benefit
  - ✓ Devices can dynamically limit platform PM state as a function of device activity level – avoids performance pitfalls
- Provides platform benefit
  - ✓ LTR removes guesswork from platform PM, enables greater power savings without impact to performance/functionality

LTR enables dynamic power vs. performance tradeoffs at minimal cost impact

Optimized Buffer Flush & Fill (OBFF)

Reducing Platform Power With Optimized Buffer Flush/Fill

- Problem statement: devices do not know power state of central resources
  - ✓ "Asynchronous" device activity prevents optimal power management of memory, CPU, RC internals by idle window fragmentation
  - ✓ Premise: If devices knew when to talk, most could easily optimize their Request patterns
    - Result: System would stay in lower power states for longer periods of time with no impact on overall performance
- Optimized Buffer Flush/Fill (OBFF) – a mechanism for broadcasting PM hint to device



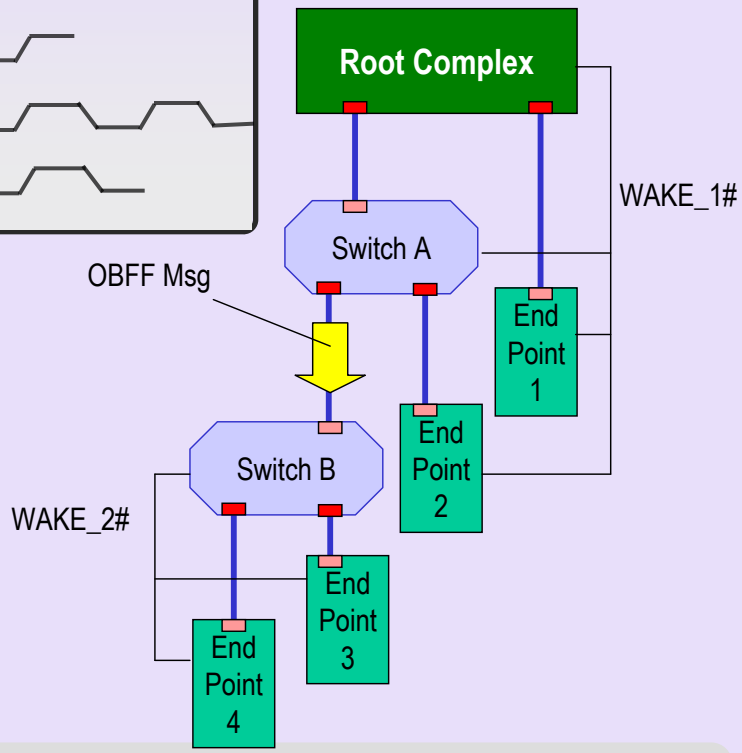
How to do OBFF?

Optimal Windows

- **CPU Active – Platform fully active. Optimal for bus mastering and interrupts**
- **OBFF – Platform memory path available for memory read and writes**
- **Idle – Platform is in low power state**

WAKE# Waveforms

Transition Event	WAKE#
Idle → OBFF	
Idle → CPU Active	
OBFF/CPU Active → Idle	
OBFF → CPU Active	
CPU Active → OBFF	



- Requirements:
  - ✓ Notify all Endpoints of optimal windows with minimal power impact
  - ✓ Keep it Simple – Maximize cost/benefit
- Solution 1: When possible, use WAKE# with expanded meanings
- Solution 2: WAKE# not available – Use PCIe Message

Greatest Potential Improvement When Implemented by <u>All</u> Platform Devices

ASPM Optionality

ASPM Optionality

- Permits full matrix of L0s and L1 support for ASPM
  - ✓ Prior to this ECN, all PCIe External Links were required to support ASPM L0s

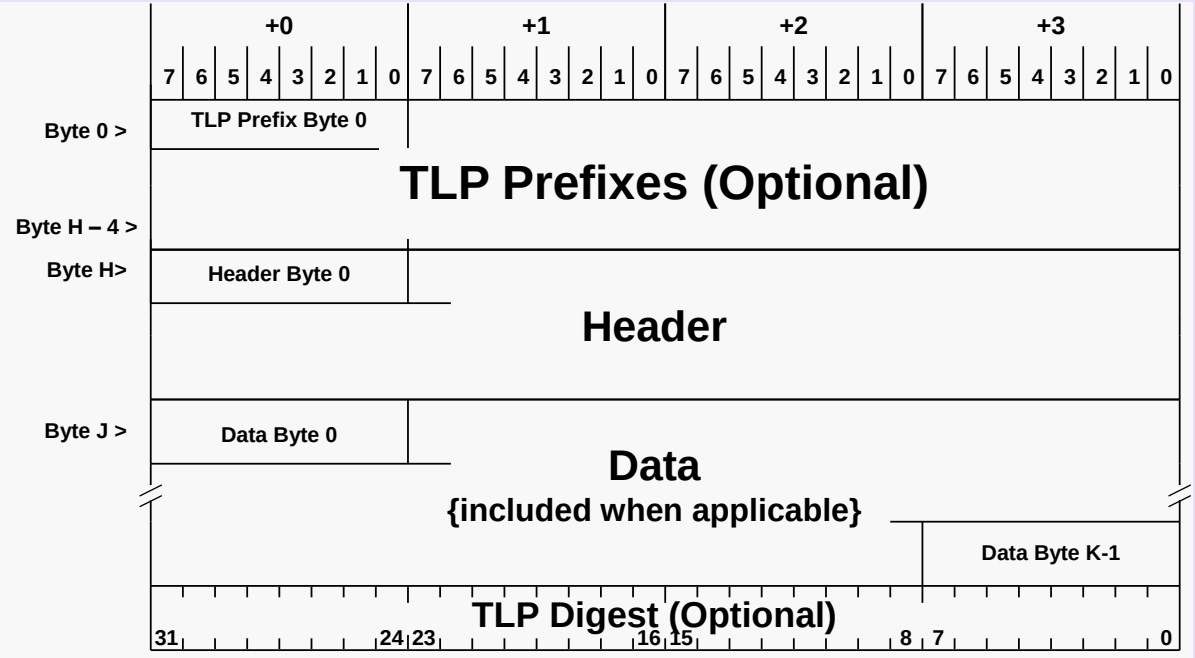
Table 5-3: Encoding of the ASPM Support Field

Field	Description
ASPM Support	00b – ~~Reserved~~No ASPM support 01b – L0s supported 10b – ~~Reserved~~L1 supported 11b – L0s and L1 supported

- Clarifies that software must not enable L0s in either direction on a given link unless components on both sides of the Link each support L0s
- Defines a new Capability bit ASPM Optionality Compliance, which software can use to help determine whether to:
  - ✓ Enable ASPM and/or
  - ✓ Run ASPM compliance tests

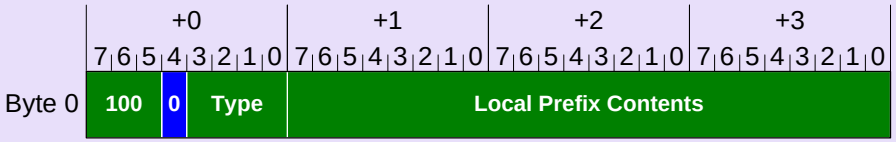
TLP Prefix

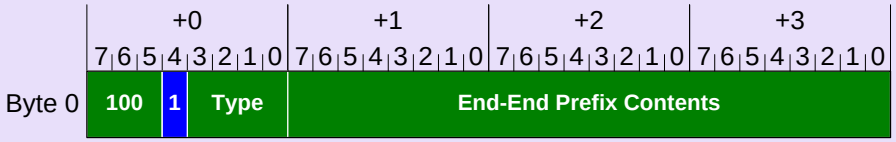
Motivation



- Emerging usage models require increase in header size to carry new information
  - ✓ Examples: Multi-Root IOV, Extended TPH
- TLP Prefix mechanism extends the header sizes by adding DWORDs to the front of headers

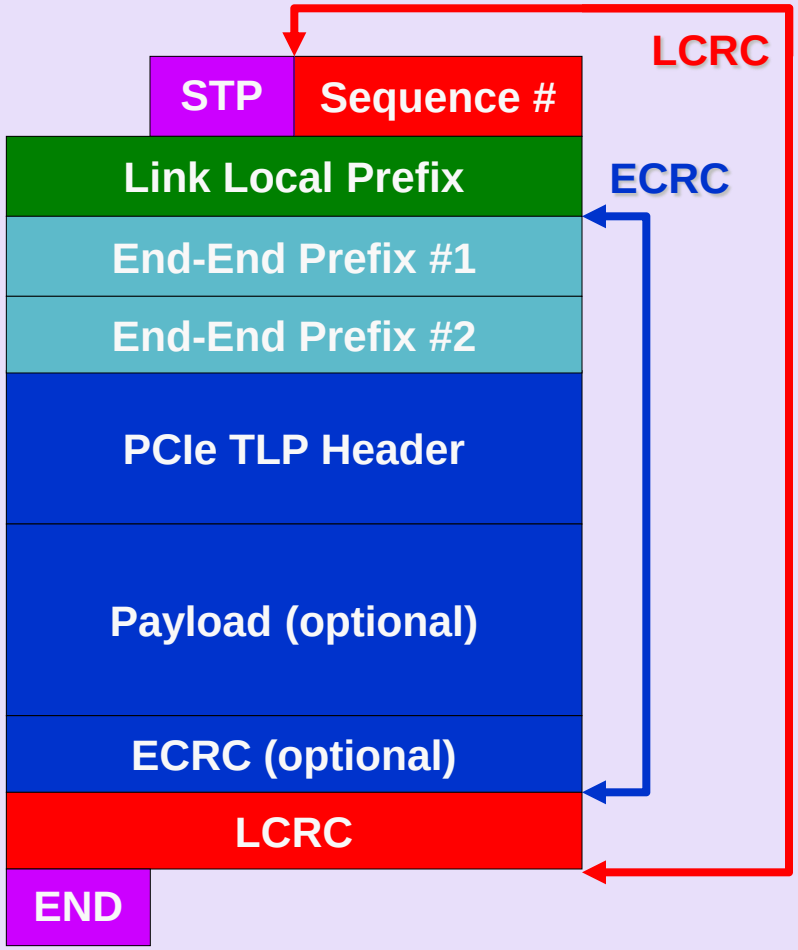
Prefix Encoding





- Base TLP Prefix Size – 1 DW
  - ✓ Appended to TLP headers
- TLP Prefixes can stacked or repeated
  - ✓ More than one TLP Prefix supported
- Link Local – Where routing elements may process the TLP for routing or other purposes.
  - ✓ Only usable when both ends understand and are enabled to handle link local TLP Prefix
  - ✓ ECRC not applicable
- End-End TLP Prefix
  - ✓ Requires support between the Requester, Completer, and routing elements
  - ✓ End-End TLP Prefix is protected by ECRC if present
  - ✓ Upper limit of 4 DWORDs (16 Bytes) for End-End TLP Prefix
- Fmt field grows to 3 bits
  - ✓ New error behavior defined
  - ✓ Undefined Fmt and/or Type values results in Malformed TLP
  - ✓ “Extended Fmt Field Supported” capability bit indicates support for 3 bit Fmt
    - – Support is recommended for all components (independent of Prefix support)

Stacked Prefix Example:



- Link Local is first
  - ✓ Starts at 0
  - ✓ $Type_{L1}$
- End-End #1 follows Link Local
  - ✓ Starts at 4
  - ✓ $Type_{E1}$
- End-End #2 follows End-End #1
  - ✓ Starts at 8
  - ✓ $Type_{E2}$
- PCIe Header follows End-End #2
  - ✓ Starts at 12
- Switch routes using Link Local and PCIe Header
  - ✓ … and Link Local Prefix if needed
  - ✓ Malformed TLP if Link Local Prefix not recognized
- Switch forwards End-End Prefixes unaltered
  - ✓ End-End Prefixes do not affect routing
  - ✓ Up to 4 DWORDs (16 Bytes) of End-End Prefix
- End-End Prefixes are optional
  - ✓ Different End-End Prefixes are unordered
    - affects ECRC but **does not** affect meaning
  - ✓ Repeated End-End Prefix sequence must be ordered
    - e.g. 1st Extended TPH vs. 2nd Extended TPH attribute
    - meaning of this is defined by each End-End Prefix

Thank you for attending the PCIe Technical Seminar

For more information please go to www.pcisig.com