PCIe Transaction Layer

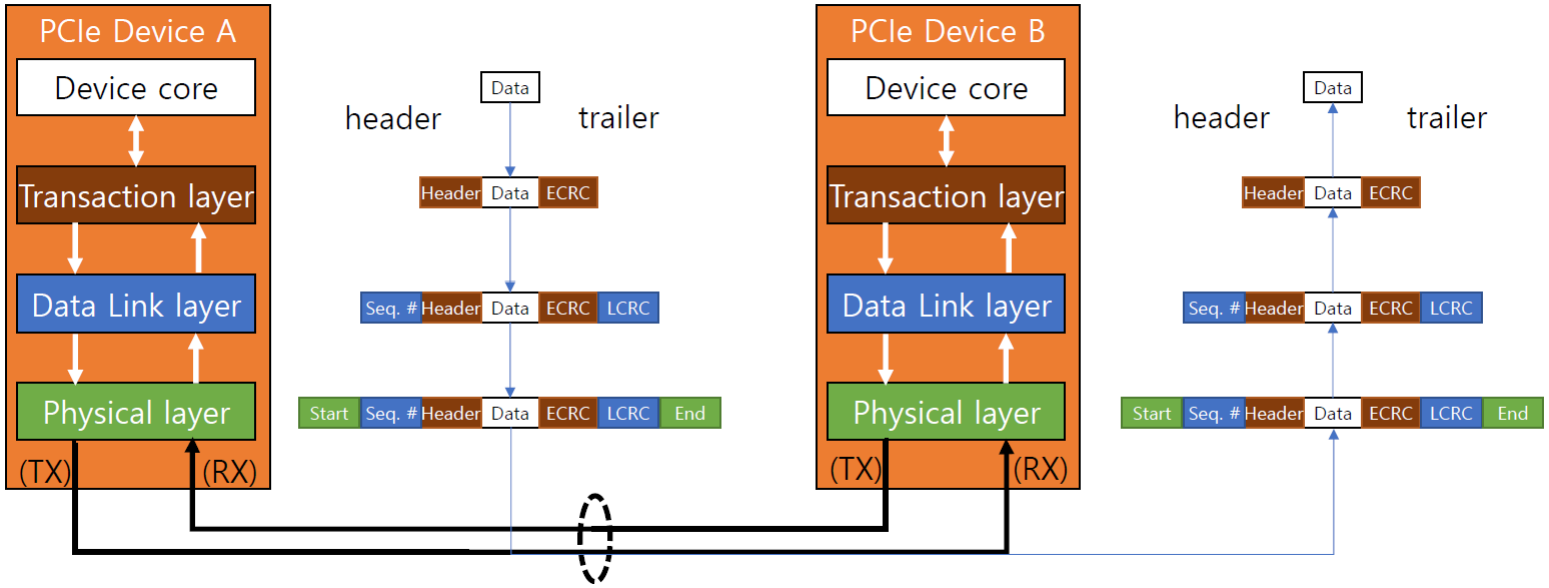
Team 2 – 최원기, 임용성, 김주성, 이승로

Contexts

- What is Transaction Layer (TL)?
- Roles of TL
- Transaction Types
- Overview of TLP Header
- TLP Request Packets
- TPH Rule
- Message Request Rule
- Completion Rule
- TLP Prefix Rule
- Handling of Received TLPs
- Transaction Ordering
- Data Integrity
- Flow Control
- Virtual Channel

What is Transaction Layer?

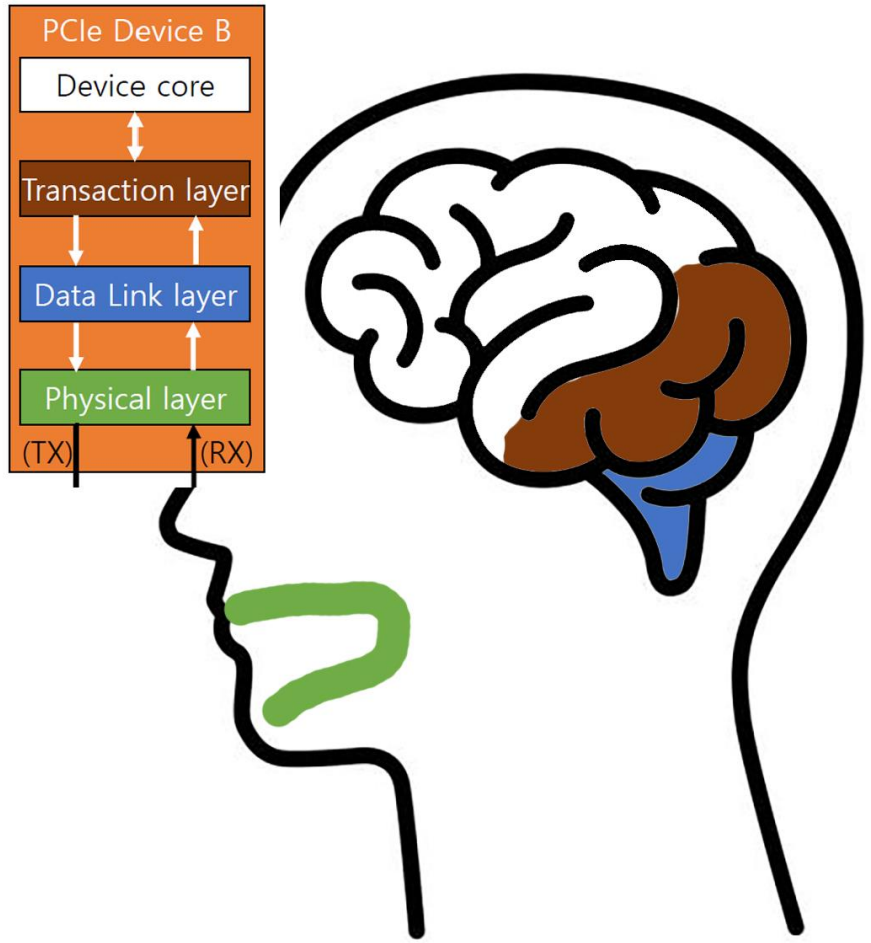
- The 'Head' of PCIe Communication
- Transaction Layer (TL) has a primary responsibility of …
  - Assembly and disassembly of Transaction Layer Packets (TLPs)



What is Transaction Layer? (cont'd)

- 두 사람의 대화를 가정…

- Transaction Layer: 뇌의 언어 영역
  - 현재 대화 '내용'을 인지하고 '의도'를 인지
- Data Link Layer: 뇌의 감각 영역
  - 현재 대화가 제대로 이뤄지고 있는지 '감지'
- Physical Layer: 입, 귀와 말초 신경
  - 현재 대화를 물리적으로 말하고 들음
- Link, Lane: 공기



What is Transaction Layer? (cont'd)

- Summary

PCI Express Controller의 우두머리.

Transaction의 내용 및 의도를 알고 있음

Transaction Flow Control을 수행

Roles of Transaction Layer

- TLP construction and processing
  - Receive request (Read, Write, Configuration …) from device core and construct, analyze packets (전송할 TLP 생성 및 전송받은 TLP 분석)

- Association of transaction-level mechanisms
  - Flow Control (Credit-based)
  - Virtual Channel management

  (Make the same channel into virtually divided channels)
  - Traffic Class (Priority)

Roles of Transaction Layer (cont'd)

- TLP 요청 및 응답 처리 (ID, Order에 대한 매칭)
- 오류 처리 및 상태 보고
- 다양한 Transaction 유형에 따른 다양한 TLP 제공
  - Memory Read/Write
  - I/O Read/Write
  - Configuration
  - Message

Transaction Types

Category	Address Space	Type	Require Completion?
Request	**Configuration**	**Read**	Yes (Non-Posted Request)
		Write	Yes
	I/O	**Read**	Yes
		Write	Yes
	Memory	**Read**	Yes
		Write	No* (Posted Request)
	Message	**R/W**	No* (Posted Request)
Completion	**Completion without Data**		(Write, Configuration)
	Completion with Data		(Read, AtomicOp)

Memory Transactions

- Data, Execution Contexts …

- Read Request/Completion
- Write Request
- AtomicOp Request/Completion
  - Locks (Compare-and-Swap / Fetch-and-Add)

- 32-bit / 64-bit address

I/O Transactions

- 지금은 별로 사용되지 않음 (Memory-mapped I/O)

- Read Request/Completion
- Write Request/Completion

- 32-bit address

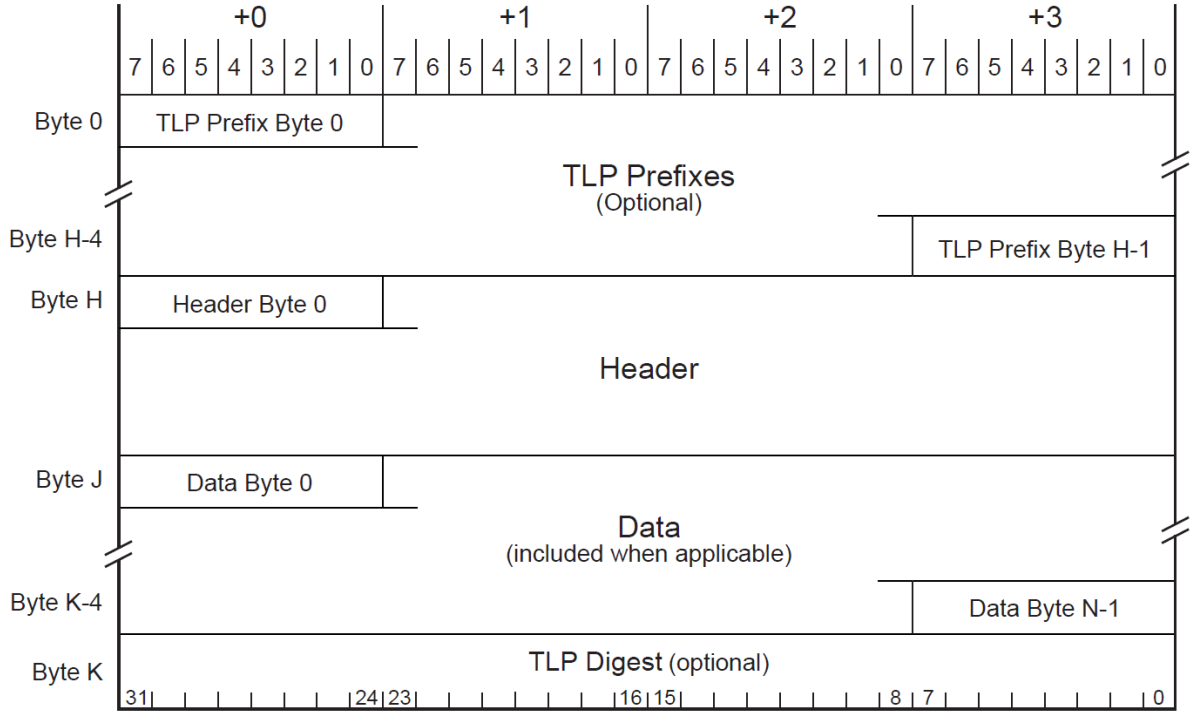
Configuration / Message Transactions

- Configuration Transactions …
  - Read Request/Completion
  - Write Request/Completion

- Message Transactions …
  - Status (Busy, Error, …)
  - Power management
  - Interrupt

Transaction Layer Packet (TLP)

- TLP Prefix – TLP Header – Data Payload – TLP Digest
  - >>> TLP Header – Data – ECRC
  - *** 낮은 번호부터 높은 번호 순서로 전송 (e.g., 0x00, 0x04, 0x08)



TLP Header Fields – Common

Field	Function	Location
Fmt[2:0]	Format of TLP (Has Data Payload? Number of headers?)	Byte 0 [7:5]
Type[4:0]	Type of TLP (Mem Read/Write, I/O, …)	Byte 0 [4:0]
Length[9:0]	Length of data payload 0b00_0000_0011: 3 DWs (12B) 0b11_1111_1111: 1023 DWs (4092B) 0b00_0000_0000: 1024 DWs (4096B)	Byte 2 [1:0], Byte 3 [7:0]
TC[2:0]	Traffic Class. Defines QoS or Priority	Byte 1 [6:4]
Attr[2:0]	Attributes. (No Snoop, Relaxed Ordering, ID-Based Ordering)	Attr[2]: Byte 1 [2] Attr[1:0]: Byte 2 [5:4]
TD	TLP Digest. Presence of TLP Digest (ECRC, …)	Byte 2 [7]
EP	Error Poisoned. Indicates the TLP is poisoned	Byte 2 [6]

Rules for TLP Header

- Length
  - # of DWs (32-bit) (** if 0, 1024 DWs)
  - If Message transaction, Length[9:0] is reserved

- Length - Max Payload
  - Device Control에 따른 max payload 제한을 어기면 안됨
  - Receiver의 max payload 제한을 어기면 안됨
  - 길이가 실제 데이터와 반드시 일치, 일치하지 않을 시 오류 처리

TLP Digest

- Optional, presence determined by TLP Header – TD Field

- End of TLP, commonly ECRC (end-to-end CRC)

*** CRC: Cyclic Redundancy Check, 오류 검출 방식

*** If Receiver does not support ECRC checking, ignore Digest

Little Endian

- There is a value of 0x12345678 …
  - Big Endian: [12, 34, 56, 78] at memory
  - Little Endian: [78, 56, 34, 12] at memory
  - PCI Express use ‘Little Endian’, transmit by LSB >> … >> MSB

- Little Endian is common
  - x86/AMD64, PCI Express
  - ARM Processor
  - File Formats (Image, …)

** Little Endian이 AtomicOp 등에서 연산에 유리함 (Compare, Add)

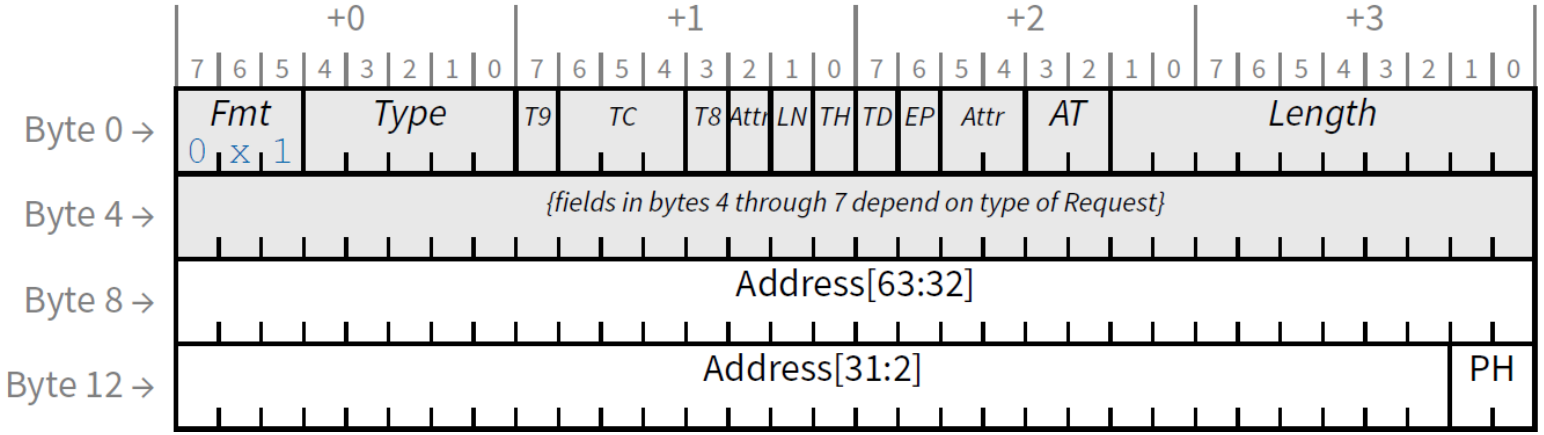
Routing and Addressing Rules (Header)

- Address-Based Routing Rules
  - Memory, I/O Requests
  - 주소가 포함된 Transaction의 경우 주소를 이용해 패킷 경로 결정
  - 스위치 등은 어느 포트로 TLP가 전달되어야 할 지 결정

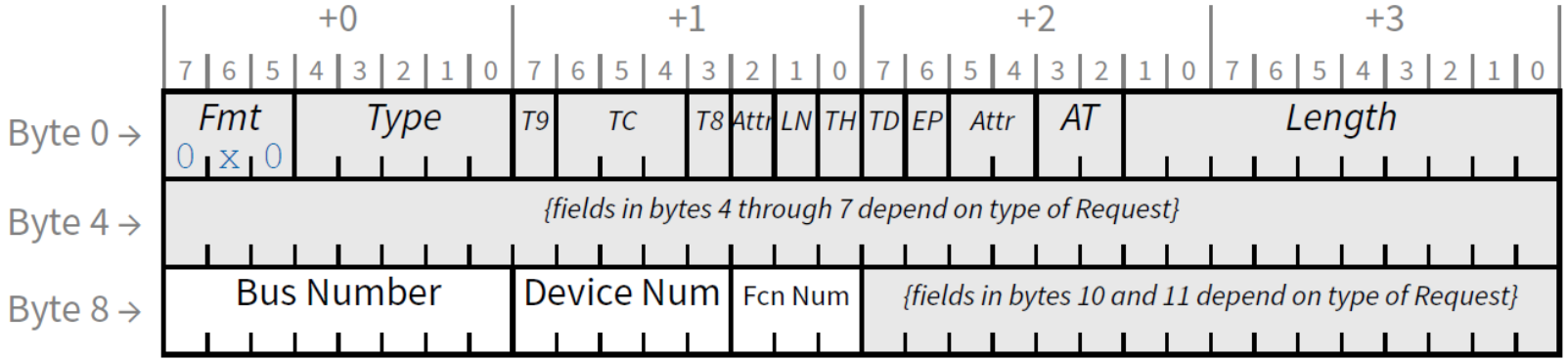
- ID-Based Routing Rules
  - Configuration Requests, Completions, Messages
  - TLP Header에 포함된 Requester ID로 패킷 경로 결정
  - 주소가 포함되지 않은 정보를 전달하기 위해 사용
  - BDF (Bus, Device, Function) ID를 통해 접근
  - *최신 디바이스의 경우, ARI (Bus, Function만 포함) 방식 사용*

Routing and Addressing Rules (Header)

- Address-Based Routing Rules

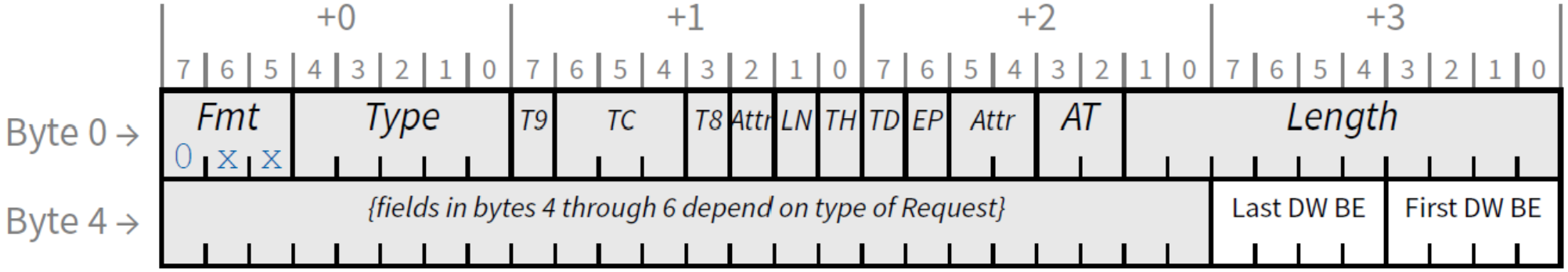


- ID-Based Routing Rules



First/Last DW Byte Enables (BE) Rules

- 첫번째/마지막 바이트 중 일부가 4바이트 정렬에서 벗어날 수 있음 >> 일부 바이트 비활성화/활성화 구분 필요
- 8bit >> 4bit/4bit for Last DW BE / First DW BE
- Valid contiguous, no non-valid bits in valid byte
- 중간 바이트는 모두 유효한 것으로 간주



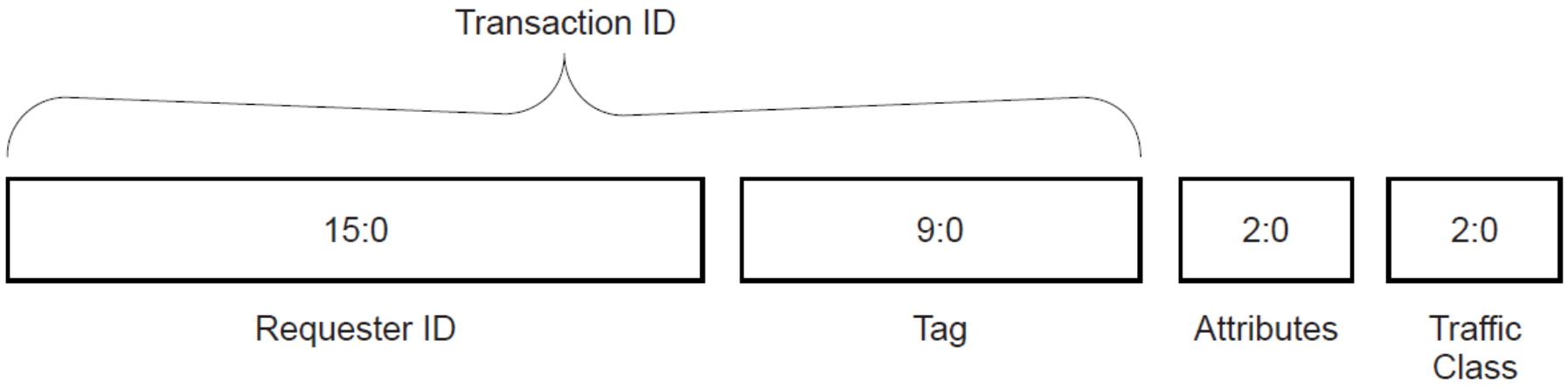
Transaction Descriptor

- Transaction information between the Requester/Completer
  - Transaction ID: identifier for request/completion (AXI rid, wid …)
  - Attributes: settings for transaction
  - Traffic Class: QoS (Virtual Channel), Priority

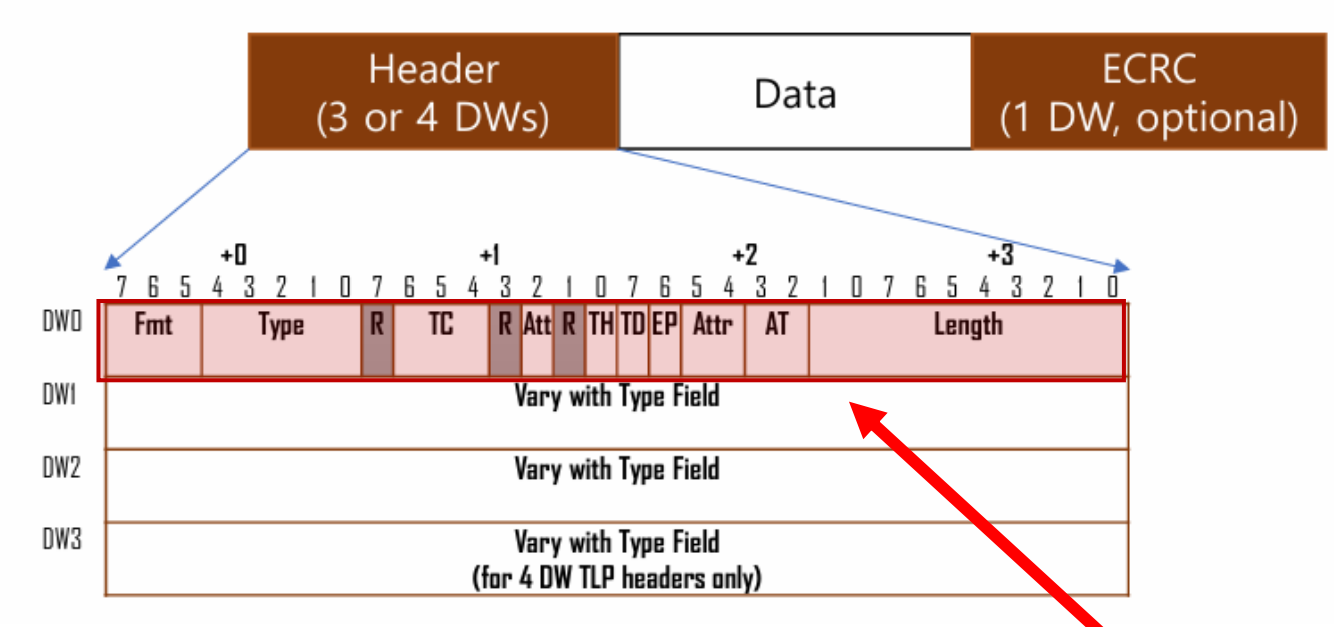
- Attributes
  - Relaxed Ordering: 순서 없이 데이터 처리
  - No Snoop: 캐시 일관성 수행하지 않음
  - ID-Based Ordering: 같은 ID에 대해서만 순차 처리

Transaction Descriptor

- Transaction ID has 16-bit Requester ID, 10-bit Tag
  - Requester ID: BDF 등으로 구성, 요청을 보낸 장치 식별
  - Tag: 여러 요청 중 각 요청을 구분하는 값



TLP Request Packet Request Rules



TLP Request Packet Type

1) Memory
2) Atomic Operations
3) IO
4) Configuration
5) Message

→ 5가지 존재, Request Rule(패킷형태)를 각각 갖는다.

→ 5 type모두 **DW0는 공통으로 포함**

Memory Request Rules

[Memory Request]

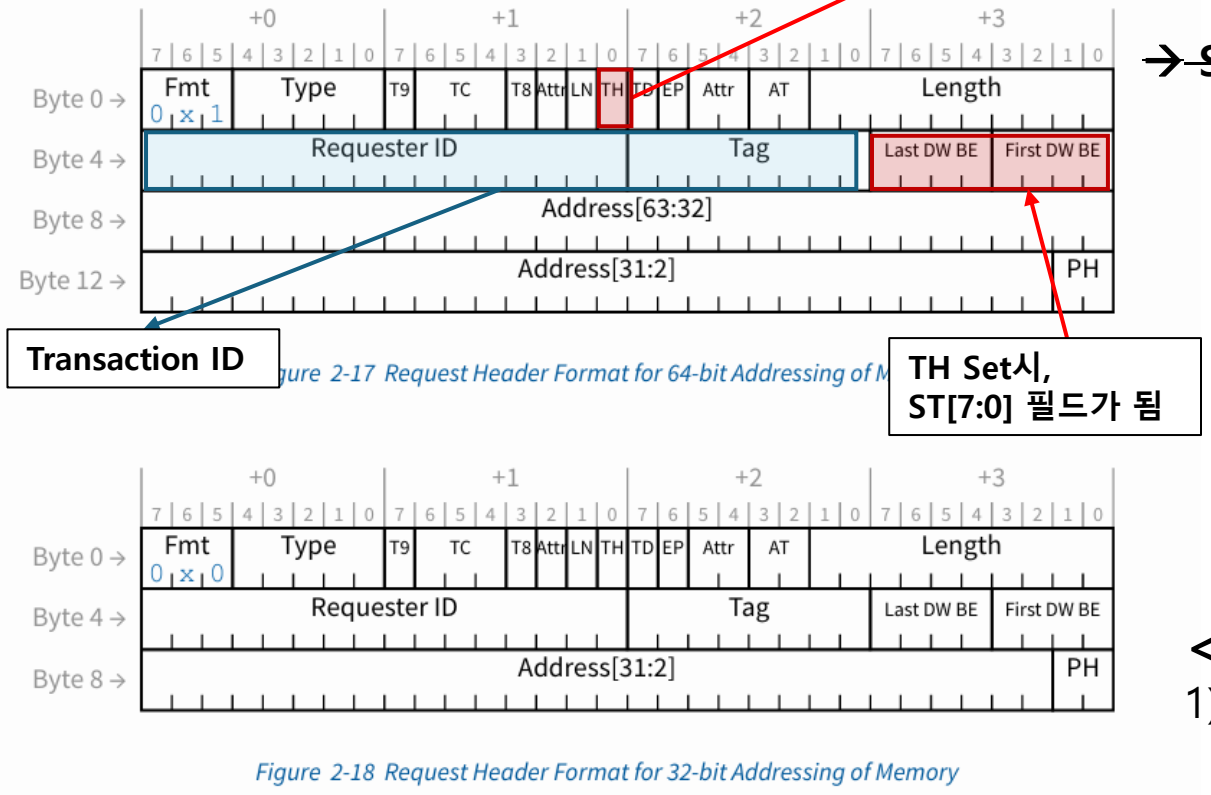


Figure 2-17 Request Header Format for 64-bit Addressing of M

Figure 2-18 Request Header Format for 32-bit Addressing of Memory

※ TH Field란?

~~Traffic Class Header Present의 약자로,~~
~~**TLP Header**에 추가적인 **Traffic Class 관련 정보**가 포함되는지 나타냄~~

~~→ **Set된다면 QoS 및 패킷 우선순위 제어 용도의 패킷**이라는 의미~~

TH bit : Set(1)
: Last DW BE, First DW Be 필드가 **ST[7:0]필드로 재할당됨**
: Byte Enable의 기능 X , **Steering Tag(**특정 CPU 코어, 캐시, 인터럽트 벡터를 지정하는 값**)**
: AtomicOp요청에서는 DW BE 필드가 "Reserved"로 처리됨

TH bit : Clear(0)
: Last DW BE, First DW Be 필드가 Byte Enable의 기능
: Memory Read Request에서 First/Last DW Byte Enable 규칙 적용

<Memory Read Request Rule>

1) Length < MAX_READ_REQUEST_SIZE
   → 틀릴시, UR(Unsupported Request) 로 처리됨

2) Address + Length가 4KB를 초과해선 안됨
   → 틀릴시, Malformed(잘못된 형식의) TLP로 처리됨

AtomicOp Request Rules

[AtomicOp Request]

Table 2-12 Length Field Values for AtomicOp Requests

AtomicOp Request	Length Field Value for Architected Operand Sizes		
	32 Bits	64 Bits	128 Bits
FetchAdd, Swap	1 DW x2	2 DW x2	N/A
CAS	2 DW	4 DW	8 DW

※ CAS는 2개의 Operand를 사용하므로,
Operand Length는 2배
CAS = Compare and Swap
1. Compare Value
2. Swap Value

<왜 필요할까?>
1. 단일 메모리 접근을 보장하여 원자성 유지
→ 정렬X 주소에서 수행 시, 연산이 여러 메모리 접근으로 나눠짐
2. 캐시 일관성 유지
→ 정렬X 시, 여러 개의 캐시 라인에 걸쳐 연산이 분산될 수 있음

<AtomicOp Request Rule>

■ Length 검증 규칙
1) Length 필드가 좌측 표와 불일치
→ Malformed TLP (잘못된 형식의 TLP)

2) Length 필드가 Completer가 지원하는 Operand 크기와 불일치
→ UR (Unsupported Request)

■ 4-KB Boundary Crossing Rule
Atompic 요청은 4KB 메모리 경계를 넘어서 접근하면 안됨.

Address + Length Size > 4KB
→ Malformed TLP

■ Address Alignment 규칙
AtomicOp Request는 Operand 크기에 맞게 자연 정렬(Naturally Aligned) 되어야 함.
→ 불일치 시 : Malformed TLP

Ex) 32-bit Operand (4B) 라면, Address는 4의 배수여야함.
→ 0x0000, 0x0004, 0x0008
64-bit Operand (8B) 라면, Address는 8의 배수여야함.
→ 0x0000, 0x0008, 0x0010

I/O & Config. Request

[I/O]

우선순위 제일 낮음

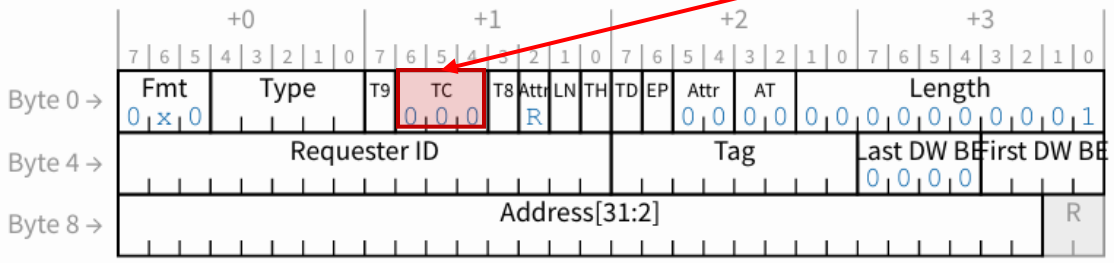


Figure 2-19 Request Header Format for I/O Transactions

[Configuration]

Receiver가 check할 필요 X

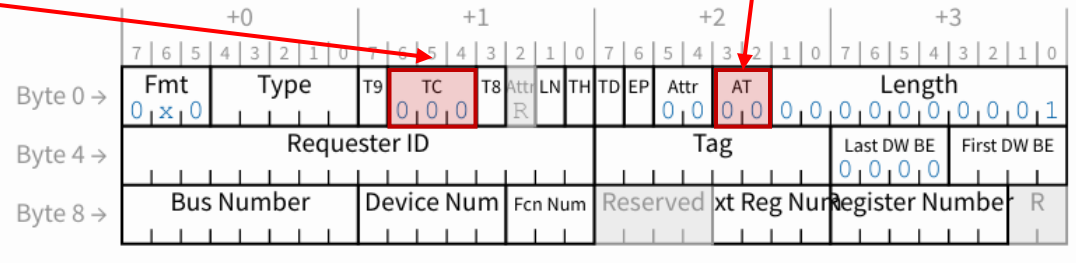


Figure 2-20 Request Header Format for Configuration Transactions

Option : Receiver가 이러한 Rule을 Check 하도록 설정할 수 있음
→ 만약 Violate 했다면, Malformed(잘못된) TLP로 처리됨.

TPH(TLP Processing Hints) Rule

- **TPH(TLP Processing Hints)란 ?**

  PCIe에서 트랜잭션을 특정 CPU 코어, 캐시 계층, 또는 DRAM으로 효율적으로 라우팅하여 캐시 오염 (Cache Pollution) 및 캐시 오버헤드를 줄이는 방법

※ TPH 미사용시, PCIe 데이트 흐름

1. Endpoint (NVMe, NIC) → DMA → DRAM
2. CPU가 해당 데이터를 읽을 때, DRAM → CPU Cache로 적재 (Cache Pollution 발생 가능)
3. 재사용되는 캐시 없이 계속, 기존 캐시 데이터 교체(Cache Eviction 발생) → 성능 저하 발생

→ **Cache Pollution과 Cache Eviction으로 인한 성능저하**

※ TPH 사용을 통한 PCIe 데이트 흐름 효율화

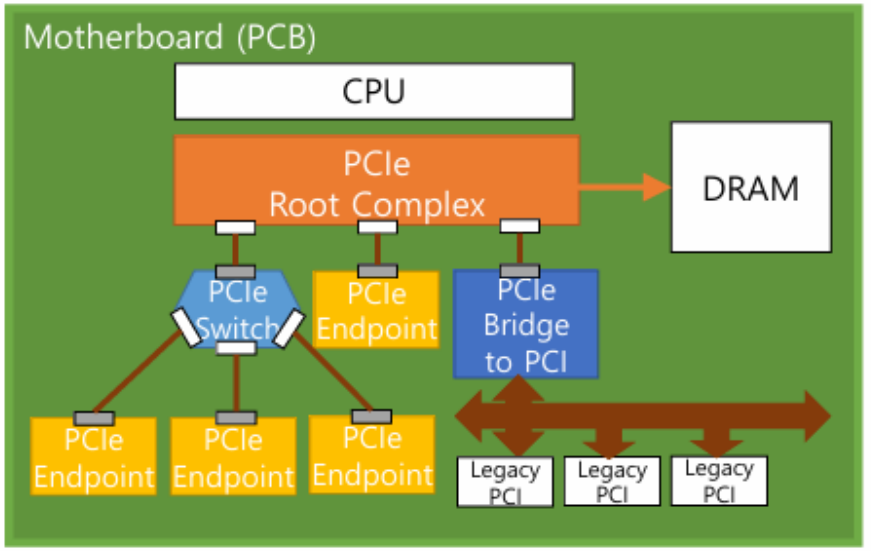
→ PCIe TLP를 통해, 특정 CPU 코어, L3 캐시, DRAM으로 효율적으로 라우팅 가능

<예시>

1) **NIC → DRAM 직접 저장**
   : CPU Cache로 데이터를 퍼나르는 것 없이, 곧바로 DRAM으로 저장(bypassing Cache)

2) **NVMe SSD → 특정 CPU Core에서 데이터 처리**
   : 모든 CPU Core에 캐시를 분산하면 성능이 저하될 수 있음.
   → 특정 CPU Core의 Cache에 데이터를 적재함으로써, 성능 저하 방지

3) **AI 가속기(GPU, NPU) → CPU 인터럽트 최적화**
   : 모든 CPU 코어에 인터럽트를 발생시키면, Context Switch로 인한 성능 저하 발생
   Interrupt 요청이 특정 CPU에서만 처리 → Context Switch 최소화



※ Cache Pollution?

: Cache를 재사용없이 사용하는 데이터가 많아져,
자주 참조하는 데이터가 캐시에서 밀려나
메모리 접근 속도가 느려지고 성능 저하 발생

→ Cache Hit Rate가 감소하는 결과

※ Context Switch?

: CPU의 이전 프로세스의 실행 상태 (Context)를 저장하고,
새로운 프로세스의 상태를 복원하는 작업

→ Context Switch의 오버헤드는 상당함

TPH(TLP Processing Hints) Rule

TPH는 TLP Prefix에 **Optionally**하게 추가할 수 있다.
→ 추가시, 더 효율적으로 트랜잭션이 처리 됨

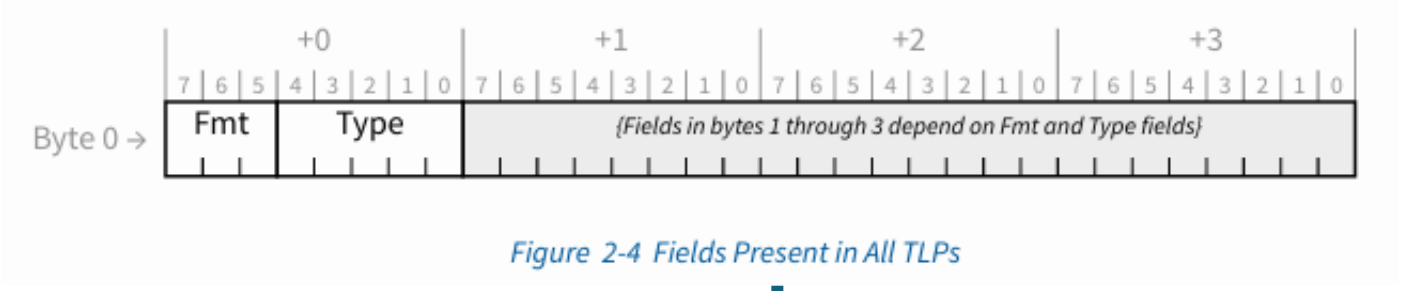


Figure 2-4 Fields Present in All TLPs

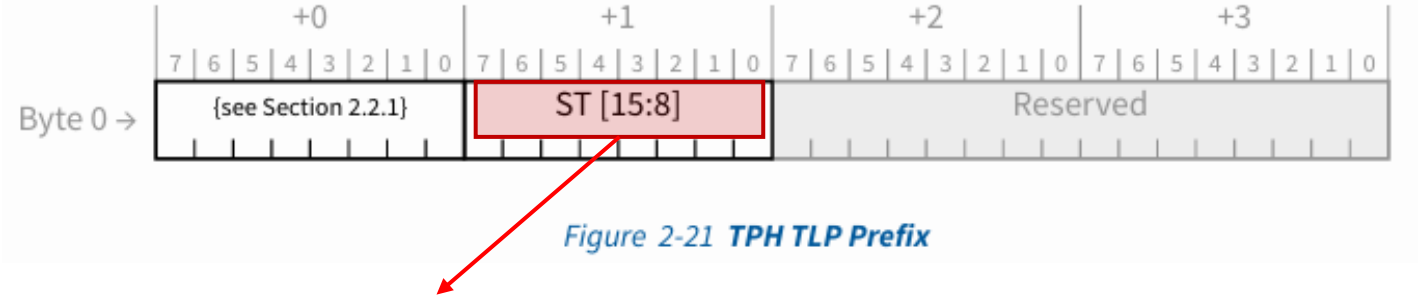


Figure 2-21 **TPH TLP Prefix**

선택적으로 TLP Prefix에서 ST[15:8]을 추가
※ 다만, 이렇게 추가할 시 TLP Header에서 TH Field를 Set해야함.

※ ST(Steering Tag) Field

: TPH에서 특정 데이터가 **어느 프로세싱 엔진(CPU코어, L3 캐시, 인터럽트 벡터 등)**에서 처리될지를 결정하는 태그
→ 워크로드를 최적화

1) **Direct Steering Mode** : ST 필드에 의해 직접 데이터 처리가 이루어짐
2) **Indirect Steering Mode** : OS또는 SW가 ST값을 기반으로 적절한 엔진을 결정

<ST Field가 지원하는 목적지>

1) **CPU 코어**
   : 특정 코어에서 데이터를 처리하도록 유도
2) **L3 캐시**
   : 데이터가 특정 캐시에 적재되도록 최적화
3) **인터럽트 벡터**
   : MSI-X 벡터 기반으로 라우팅
4) **DRAM**

TPH(TLP Processing Hints) Rule

TPH는 Memory, AtomicOp Request에서만 사용 가능

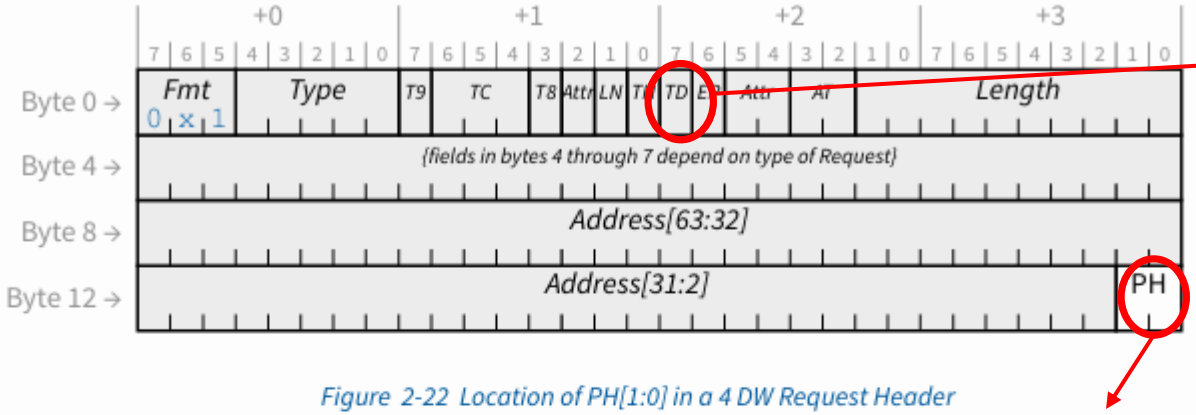


Figure 2-22 Location of PH[1:0] in a 4 DW Request Header

TPH 사용 시, **TH 필드가 Set(1)**되어야함.

Processing Hint Bit

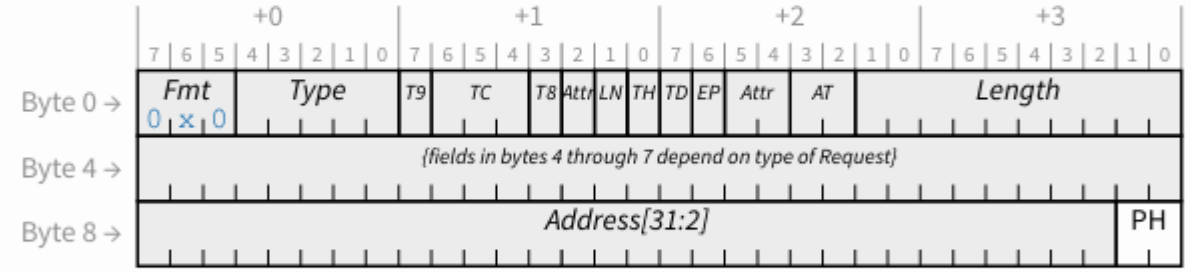


Figure 2-23 Location of PH[1:0] in a 3 DW Request Header

Table 2-14 Location of PH[1:0] in TLP Header

PH	32-bit Addressing	64-bit Addressing
1:0	Bits 1:0 of Byte 11	Bits 1:0 of Byte 15

PH	Processing Hint	설명	EX
00b	Bi-Directional Data Structure	호스트와 PCIe 장치가 데이터를 자주 주고받는 경우	NIC에서 패킷 송수신
01b	Requestor	PCIe 장치(ex, SSD, GPU)가 자주 데이터를 읽거나 쓰는 경우	GPU가 데이터를 직접 읽어서 처리
10b	Target	호스트가 데이터를 자주 읽거나 쓰는 경우	CPU가 NVMe SSD에서 데이터를 지속적으로 읽어오는 경우
11b	Target with Priority	CPU가 데이터를 자주 접근하며, 해당 데이터가 높은 시간적 지역성을 가짐	CPU가 주로 캐시 히트가 잘 나는 데이터를 다루는 경우

TPH(TLP Processing Hints) Rule

[TLP Header – Memory Write Request]

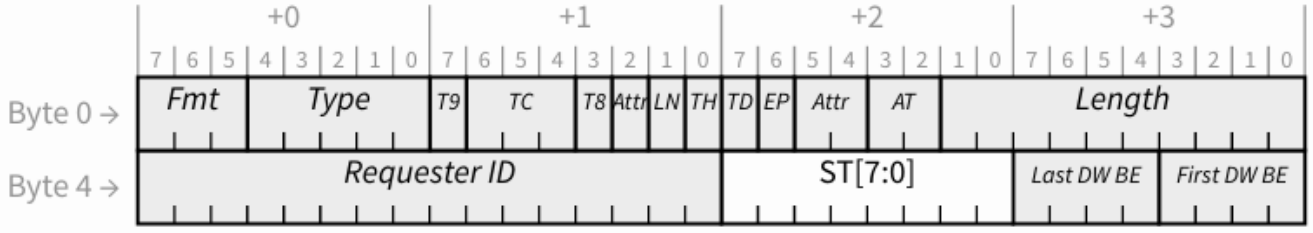


Figure 2-24 Location of ST[7:0] in the Memory Write Request Header

[TLP Header – Memory Read and AtomicOp Request]

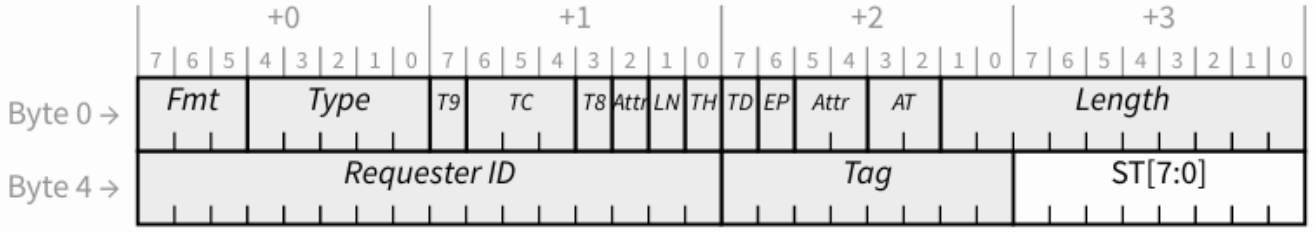


Figure 2-25 Location of ST[7:0] in Memory Read and AtomicOp Request Headers

Message Request Rule

[TLP Header]

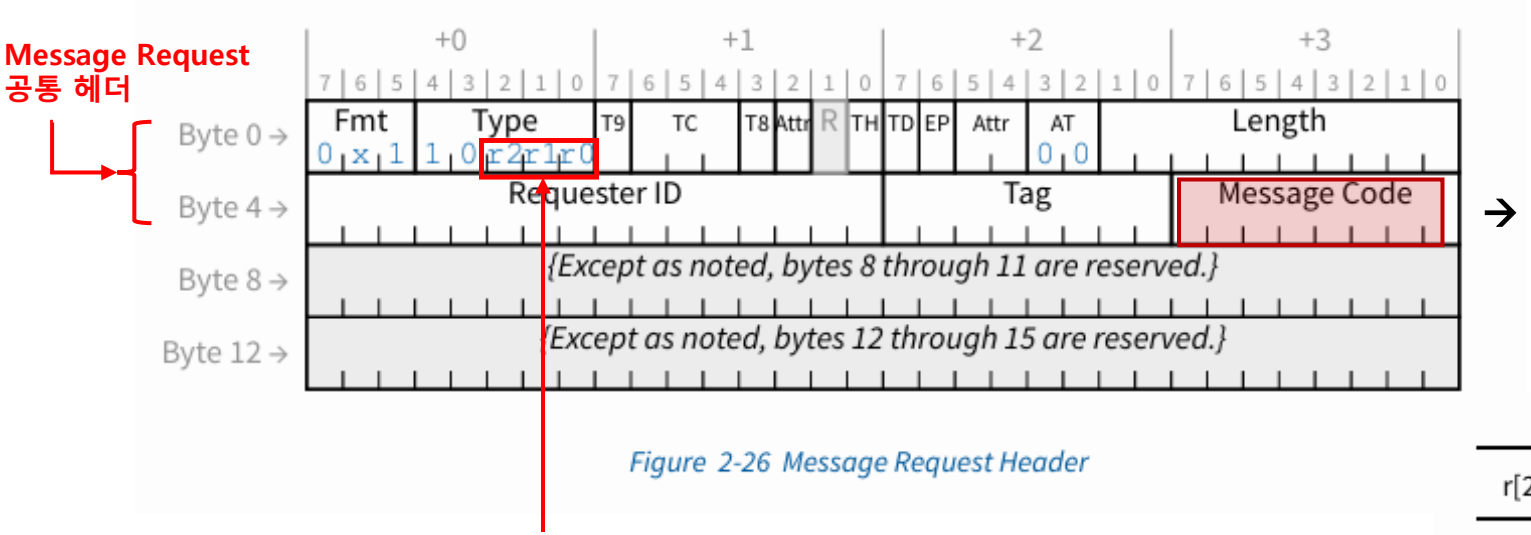


Figure 2-26 Message Request Header

→ Message Code[7:0] 필드로 Message Type이 정해짐

Type의 r[2:0] : Message Routing을 정의 (Implicit 라우팅 메커니즘)

※ Implict(암시적) 라우팅 메커니즘이란?
: 명확한 주소나 ID 없이 특정 대상에게
메시지를 보내는 것

Table 2-17 Message Routing

r[2:0] (b)	Description	Bytes 8 to 15[15]
000	Routed to Root Complex	Reserved
001	Routed by Address[16]	Address
010	Routed by ID	See Section 2.2.4.2
011	Broadcast from Root Complex	Reserved
100	Local - Terminate at Receiver	Reserved
101	Gathered and routed to Root Complex[17]	Reserved
110 to 111	Reserved - Terminate at Receiver	Reserved

Message Request Type

Message Type	설명	사용 예시
INTx Interrupt Siganling Messages	Legacy PCI 인터럽트 (INTA# ~ INTD#)를 PCIe 메시지로 변환	오래된 PCI 장치가 IRQ(Interrupt Request)를 보내면, PCIe 인터럽트 메시지로 변환되어 전달됨.
Power Management Messages	PCIe 장치의 전력 상태 변경(D0, D1, D2, D3) 관리	노트북에서 PCIe 기반 Wi-Fi 카드가 사용되지 않을 때, D3 상태(저전력 상태)로 전환됨.
Error Signaling Messages	PCIe 장치에서 오류를 감지하여 보고	데이터 전송 중 CRC 오류가 발생하면, PCIe 디바이스가 AER 메시지를 보내서 오류를 보고하고 복구 절차를 수행할 수 있음.
Locked Transactions Messages	데이터 일관성을 유지하기 위한 Atomic Transaction 지원	DMA 엔진이 공유 데이터 영역에서 **Fetch and Add** 연산을 수행할 때, 동시 접근을 방지하기 위해 Locked Transaction을 사용.
Slot Power Limit Messages	PCIe Slot이 제공할 수 있는 최대 전력 설정	서버 환경에서 PCIe 장치가 허용된 전력을 초과하는 요청을 하면, Slot Power Limit 메시지를 통해 전력 조절이 가능함.
Ignored Messages	PCIe 스펙에는 정해져있으나, 현재 사용하지 않는 메시지	PCIe 1.0 또는 2.0 시절에 정의되었으나, 최신 버전에서는 사용되지 않는 메시지.
Latency Tolerance Reporting(LTR) Messages	장치의 허용 가능한 지연 시간을 보고하여 전력 관리 최적화	NVMe SSD가 "나는 10마이크로초 동안 데이터 요청을 기다릴 수 있다"라는 LTR 메시지를 보내면, 시스템이 전력 최적화를 위해 PCIe 링크 속도를 낮출 수 있음.
Optimized Buffer Flush/Fill (OBFF) Messages	전력 효율성을 높이기 위해 Buffer Flush/Filling 최적화	NVMe SSD가 여러 개의 연속적인 요청을 하나의 버퍼에서 처리할 수 있도록 OS와 협력하여 OBFF 메시지를 전송함.
Precision Time Measurement (PTM) Messages	PCIe 장치 간 정밀한 시간 동기화 지원	서버 클러스터에서 여러 개의 PCIe SSD가 동일한 타임스탬프를 유지하기 위해 PTM 메시지를 사용함.
Vendor_Defined Messages	PCIe에서 벤더(제조사)가 독자적으로 정의한 메시지	
ㄴ **PCI-SIG-Defined VDMs**	Vender가 독자적으로 정의한 메시지 (Vender 전용 기능 수행	PCI-SIG에서 공식적으로 정의한 벤더 전용 메시지.
ㄴ **Locally Notification (LN) Messages**	특정 지역(Locality)에서 처리되어야 하는 트랜잭션을 알림	PCIe 스위치에서 특정 포트를 통해 **우선적으로 처리해야 하는 트랜잭션**을 LN 메시지를 통해 지정함.
ㄴ **Device Readiness Status (DRS) Message**	PCIe 장치가 초기화를 완료하고 준비됨을 보고	서버의 NVMe SSD가 "나는 정상적으로 동작할 준비가 되었음"을 알리는 DRS 메시지를 전송.
ㄴ **Function Readiness Status (FRS) Message**	PCIe 장치의 특정 기능이 준비됨을 보고	그래픽 카드(GPU)가 "나는 CUDA 기능을 사용할 준비가 되었습니다"라는 FRS메시지를 전송.
ㄴ **Hierarchy ID Messages**	PCIe 장치 및 링크의 계층 구조를 나타내는 메시지	PCIe 스위치 및 브리지가 장치 트리(Device Tree) 내에서 자신의 위치를 알리기 위해 Hierarchy ID 메시지를 전송.

Completion Rules

- 모든 Read, Non-Posted Write, AtomicOp 요청에는 Completion이 필요함
- Completion에는 Completion head가 포함되고, 유형에 따라 DW 데이터가 있을 수 있음.
- Completion은 ID에 의해 보내지고, 3 DW header를 사용함.
- Requester ID field는 개별 field가 아닌 하나의 ID field로 취급됨
- header field와 ID routing field 외에도 다음과 같은 추가 field를 포함함

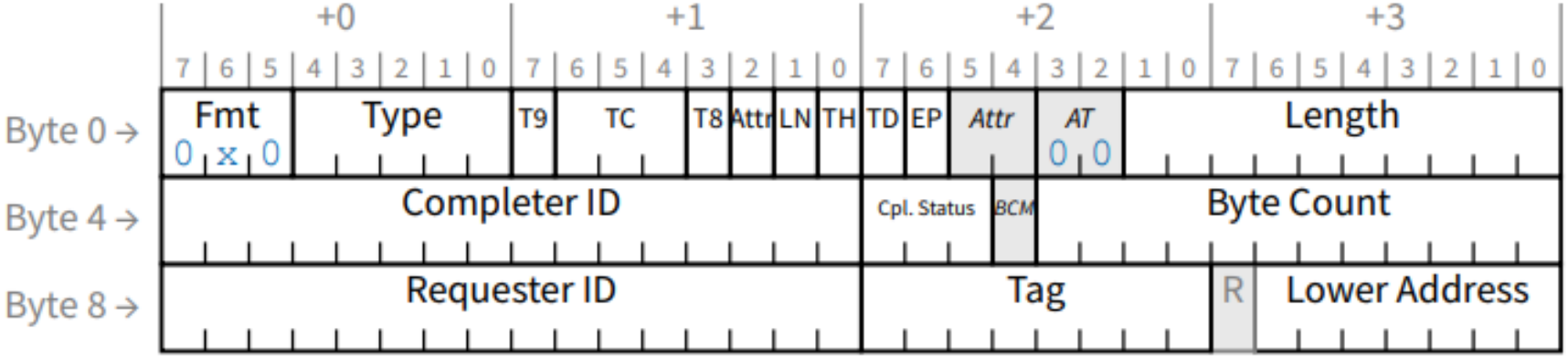


Figure 2-38 Completion Header Format

Completion Rules

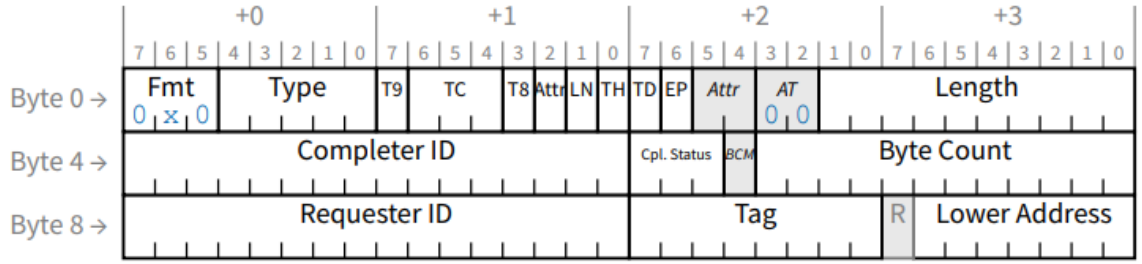


Figure 2-38 Completion Header Format

- Completer ID[15:0] - Completer를 식별
- Completion Status[2:0] - Completion의 상태를 나타냄 (Table 2-35)

Table 2-35 Completion Status Field Values

Cpl. Status[2:0] Field Value (b)	Completion Status
000	Successful Completion (SC)
001	Unsupported Request (UR)
010	Configuration Request Retry Status (CRS)
100	Completer Abort (CA)

Completion Rules

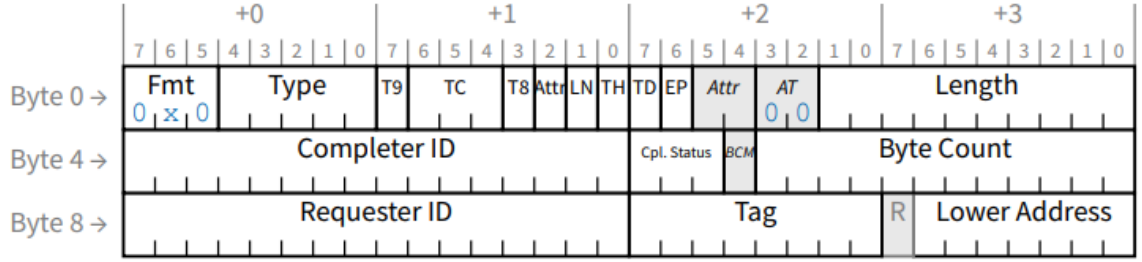


Figure 2-38 Completion Header Format

- BCM – 이 bit는 PCIe에서 설정할 수 없고, PCI-X에서만 설정이 가능함
- Byte Count[11:0] – 요청된 데이터의 남은 byte 수를 나타냄
- Tag[9:0] – Requester ID와 함께 Transaction ID를 식별하는 역할을 함
- Lower Address[6:0] – Completion이 시작되는 Byte를 나타냄

TLP Prefix Rules

- TLP Prefix는 PCIe의 확장 기능을 지원하는 메커니즘임
- 모든 TLP에서, 첫 번째 byte의 Fmt[2:0] field 값이 100b이면 TLP Prefix가 존재함을 의미함
- Type[4] bit 값에 따라 Local TLP Prefix 또는 End-End TLP Prefix로 구분됨
- Type[4] = 0b -> Local TLP Prefix
- Type[4] = 1b -> End-End TLP Prefix
- TLP는 여러 개의 TLP Prefix를 포함할 수 있고, 각 TLP Prefix의 크기는 1DW이며, 여러 개를 연속적으로 사용해 추가 데이터를 포함할 수 있음

TLP Prefix Rules

Local TLP Prefix는 PCIe 링크 내에서만 사용됨

Local TLP Prefix의 유형은 Type field의 L[3:0] sub-field를 사용해 결정함

Table 2-36 Local TLP Prefix Types

Local TLP Prefix Type	L[3:0] (b)	Description
MR-IOV	0000	**MR-IOV TLP Prefix** - Refer to [MR-IOV] specification for details.
VendPrefixL0	1110	**Vendor Defined Local TLP Prefix** - Refer to Section 2.2.10.1.1 for further details.
VendPrefixL1	1111	**Vendor Defined Local TLP Prefix** - Refer to Section 2.2.10.1.1 for further details.
		All other encodings are Reserved.

- Local TLP Prefix 유형에 따라 size, routing, flow control방법이 다름

TLP Prefix Rules

- End-End TLP Prefix는 Endpoints간 전송되는 추가 정보를 포함
- End-End TLP Prefix의 유형은 Type field의 E[3:0] sub-field를 사용해 결정함

Table 2-37 End-End TLP Prefix Types

End-End TLP Prefix Type	E[3:0] (b)	Description
TPH	0000	**TPH** - Refer to Section 2.2.7.1 and Section 6.17 for further details.
PASID	0001	**PASID** - Refer to Section 6.20 for further details.
VendPrefixE0	1110	**Vendor Defined End-End TLP Prefix** - Refer to Section 2.2.10.2.1 for further details.
VendPrefixE1	1111	**Vendor Defined End-End TLP Prefix** - Refer to Section 2.2.10.2.1 for further details.
		All other encodings are Reserved.

Handling of Received TLPs

- 수신된 TLP는 다음 Flowchart와 같이 처리됨
- TLP가 formation rules을 따랐는지 확인하고 그렇지 않았다면 해당 TLP를 삭제하고 잘못된 Packet이라 보고함
- Type field의 값이 정의 되어 있는지
- TLP가 Request인지 Completion인지

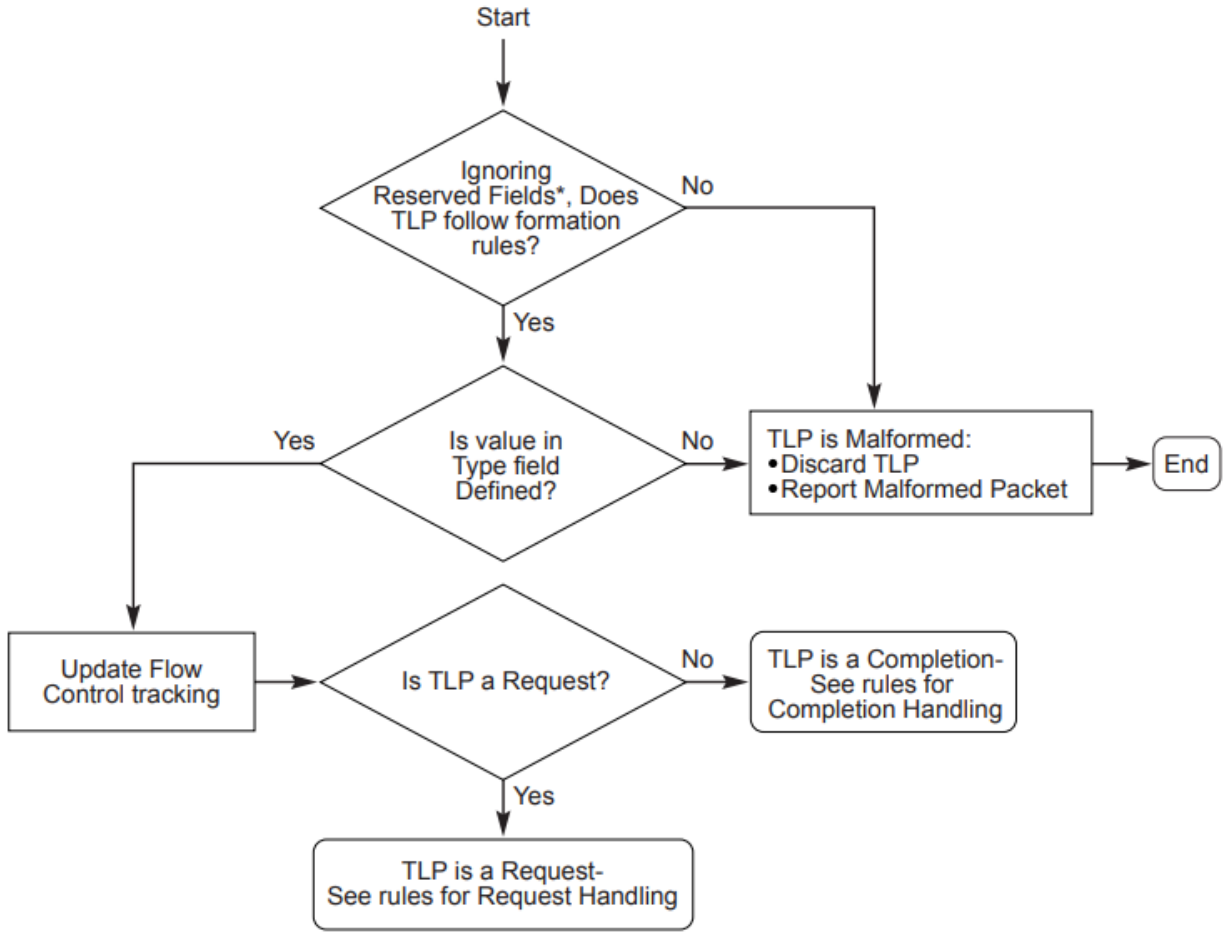


Figure 2-41 Flowchart for Handling of Received TLPs

Handling of Received TLPs

Request Handling Rules

1. Request가 지원되지 않을 경우
   - Device에서 지원하지 않는 Request type인 경우 해당 요청은 UR(Unsupported Request)로 처리됨
2. Request Type이 Msg인 경우

   2-1. Message Code field가 잘못된 경우
   - UR 상태로 보고됨

   2-2. Message Code field의 값이 정의된 경우
   - 해당 Message를 처리함

Handling of Received TLPs

3. Request가 programming model을 위반 하는 경우
   - Request가 Completion을 필요로 하는 경우 Completion Status 값을 CA(Completer Abort)로 설정하여 Completion Packet을 반환해야 함
4. Request가 정상적으로 처리되는 경우
   - Request가 유효하고 Device에서 처리 가능한 경우 요청에 대한 적절한 작업을 수행, Completion을 필요로 하는 경우 Completion Packet을 생성해 반환함

Handling of Received TLPs

- Request Handling Rules을 flowchart로 나타내면 다음과 같음

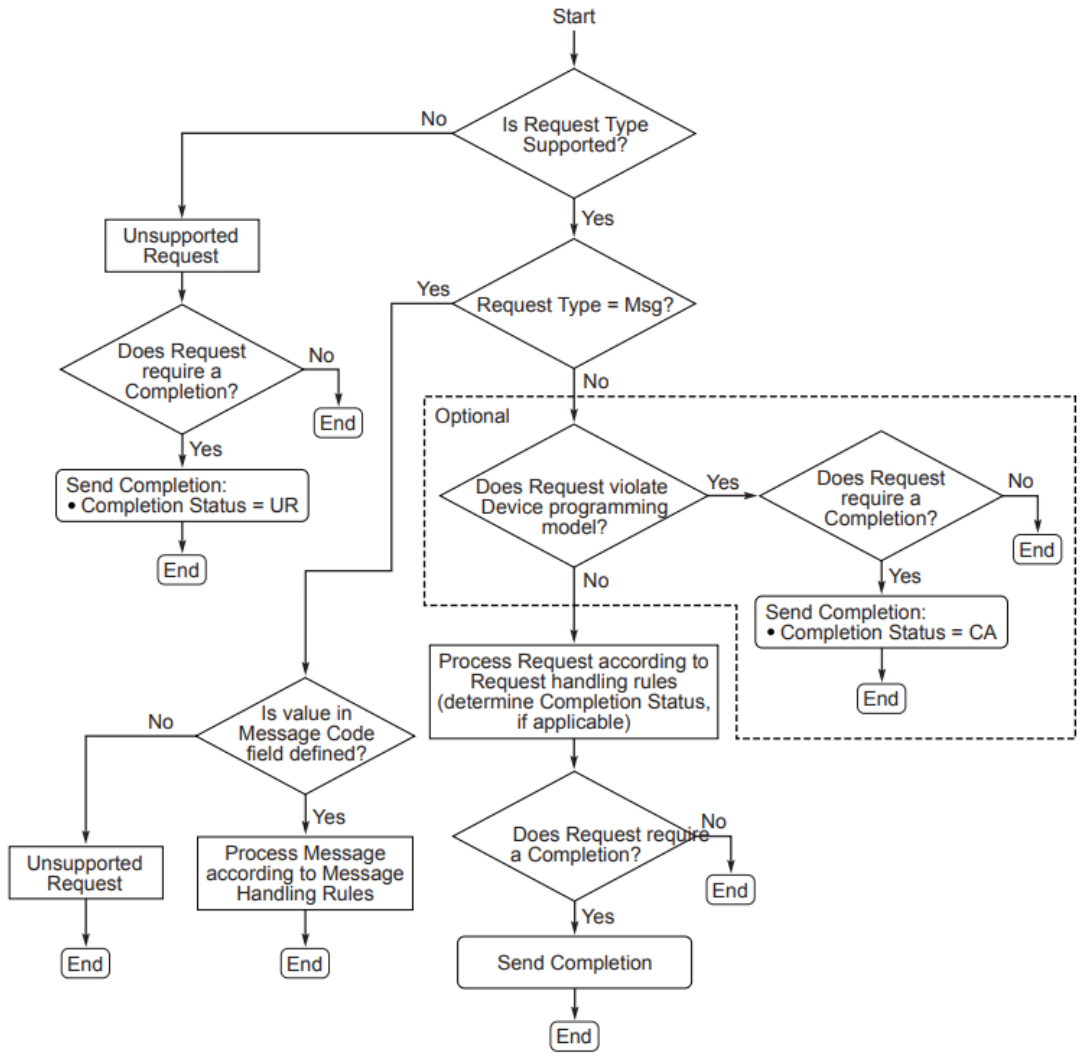


Figure 2-43 Flowchart for Handling of Received Request

Handling of Received TLPs

Completion Handling Rules

1. Unexpected Completion
   - Device가 요청한 적 없는 Transaction ID를 가진 Completion이 도착하면 이를 Unexpected Completion이라고 함
   - Unexpected Completion이 수신되면 해당 Packet을 폐기하고 오류를 보고함
2. Completion Status 값이 비정상적인 경우
   - Completion Status가 SC(Successful Completion)가 아닌 경우
   - Device는 해당 Request와 관련된 내부 자원을 해제해야 함
   - Requester는 자체적으로 오류를 처리해야 함

Handling of Received TLPs

3. Reserved Completion Status 처리
   - 정의되지 않은 Completion Status 값이 수신된 경우, 해당 Completion을 UR(Unsupported Request)로 처리해야 함
   - UR 및 CA(Completer Abort) 상태의 Completion은 PCI 방식으로 오류를 보고해야 함
4. Read Completion이나 AtomicOp Completion이 실패한 경우
   - Completion에 데이터가 포함되지 않음
   - CplD 대신 Cpl을 사용해야 함
   - 해당 Completion을 Request 의 마지막 Completion으로 간주되며, 추가적인 Completion이 전송되지 않음

Transaction Ordering

- Transaction Ordering Rule은 Transaction이 처리되는 순서를 설명함
- 이 규칙은 Deadlock방지, 시스템 성능 최적화, Producer/Consumer 모델 지원을 위해 필수적임
- PCIe에서 Transaction Ordering Rule은 TC(Traffic Class) 내에서만 적용됨
- 서로 다른 TC 간에는 순서 보장이 제공되지 않음

Transaction Ordering

- Transaction의 유형은 다음과 같음

Table 2-40 Ordering Rules Summary

Row Pass Column?		Posted Request (Col 2)	Non-Posted Request		Completion (Col 5)
			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	Read Request (Row B)	a) No b) Y/N	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

- Posted Request: Memory Write 또는 Message Request
- Read Request: Configuration Read, I/O Read, Memory Read Request
- NPR with Data: Configuration Write, I/O Write, AtomicOp Request

Transaction Ordering

Transaction 간 순서 관계

1. Posted Request 간의 순서 관계
   - 기본적으로 Posted Request간에는 순서를 유지해야 함
   - RO(Relaxed Ordering)이 설정된 경우 일부 예외가 허용됨
   - 이 경우 후속 Posted Request가 이전의 Posted Request보다 먼저 도착할 수 있음

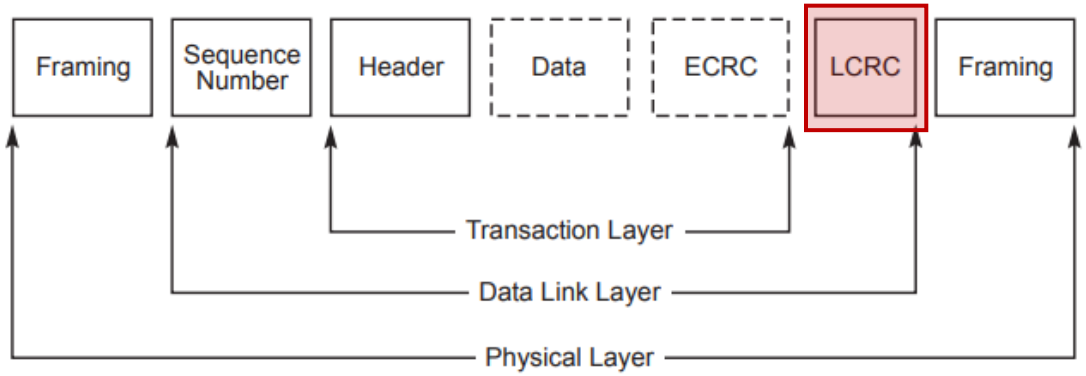
Transaction Ordering

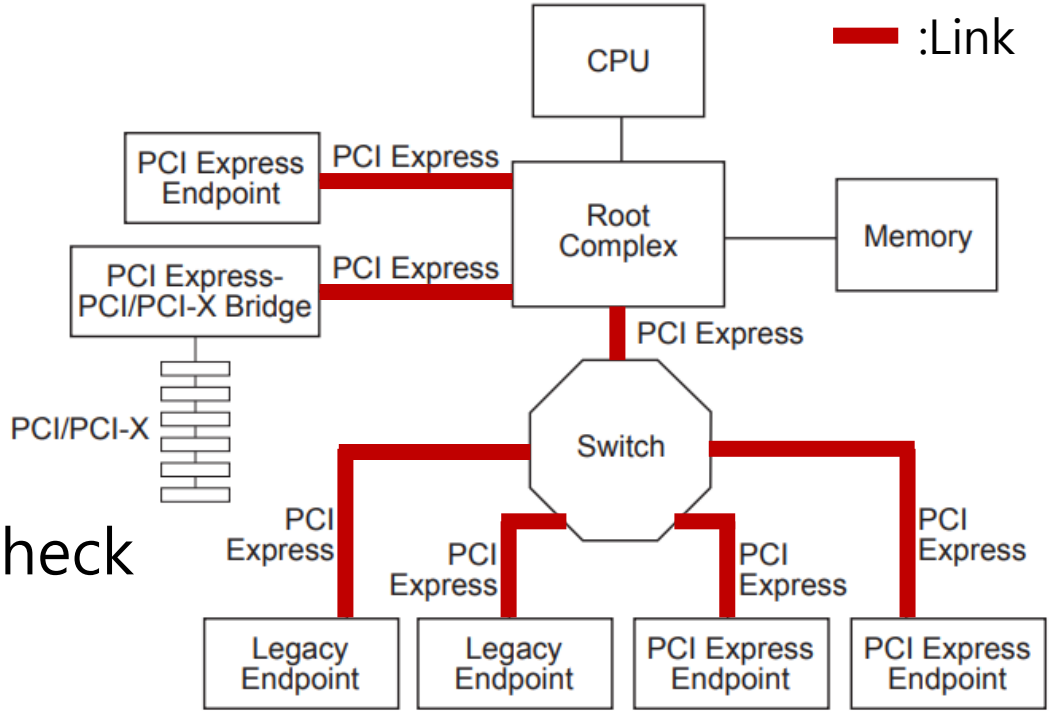
2. Posted Request VS Non-Posted Request
   - Posted Request는 반드시 Non-Posted Request보다 우선적으로 전송될 수 있어야 함
   - 즉 Memory Write 같은 Posted Request는 Read Request와 같은 Non-Posted Request 보다 먼저 처리될 수 있음
3. Posted Request VS Completion
   - Posted Request는 Completion보다 우선적으로 전송될 수 있음

Transaction Ordering

4. Read Request VS 다른 Transaction
   - Read Request는 기본적으로 이전에 발생한 Posted Request를 우회할 수 없음
5. Non-Posted Request VS 다른 Transaction
   - Non-Posted Request는 다른 Non-Posted Request를 우회할 수 있음
   - Completion은 Non-Posted Request보다 우선적으로 전송될 수 있음
6. Completion 간 순서 관계
   - Completion Packet들이 동일한 Transaction ID를 공유하는 경우, 순서가 유지되어야 함
   - 서로 다른 Transaction ID를 가진 Completion들은 순서를 바꿔도 무방함

Data Integrity

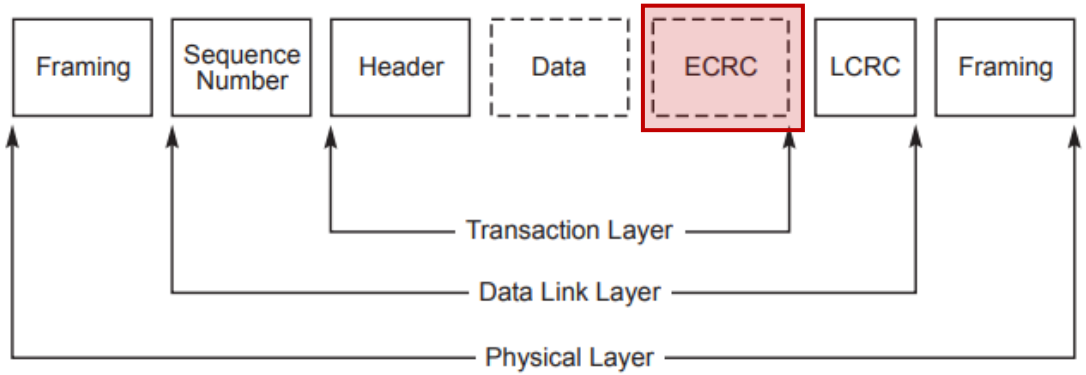




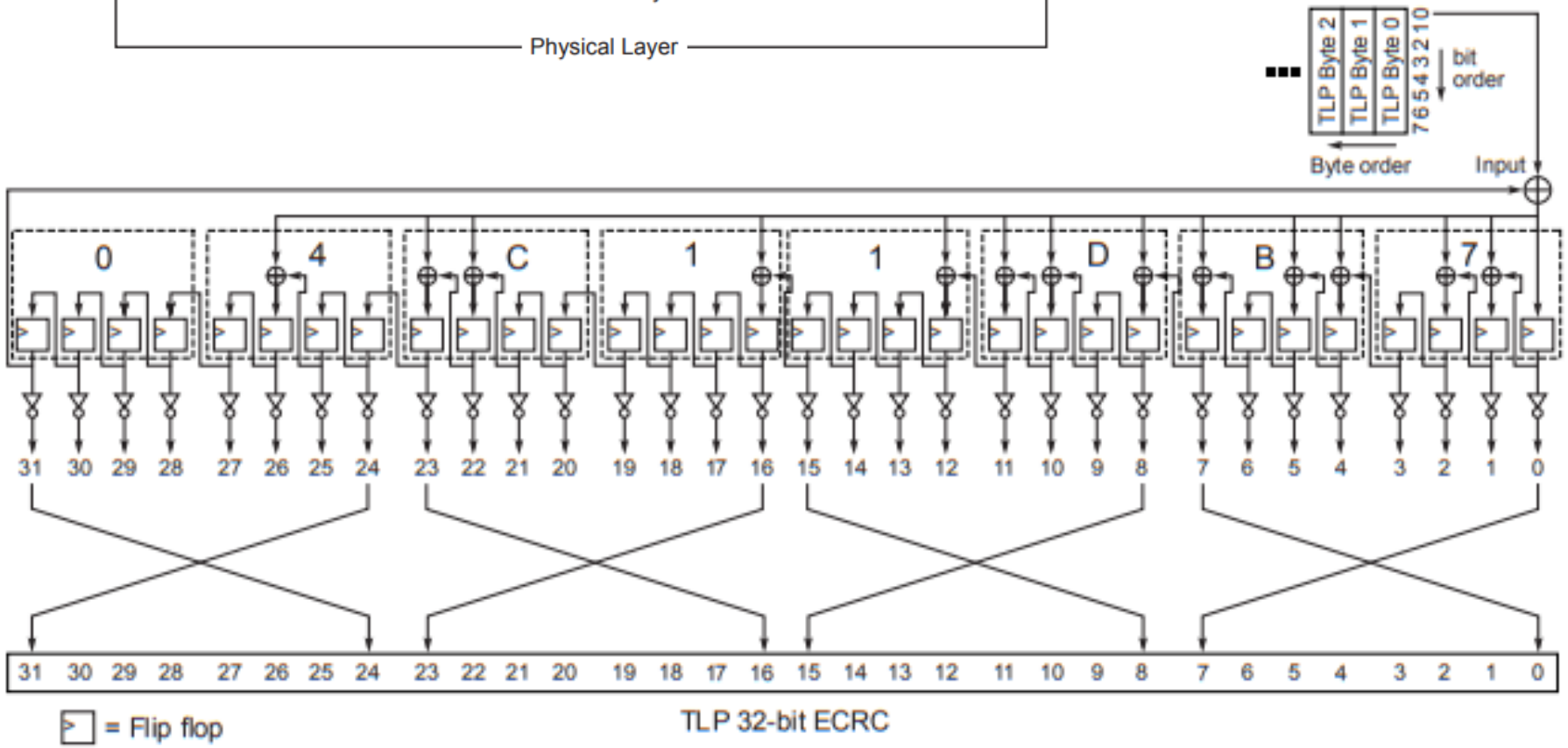
- LCRC: Link-to-link Cyclic Redundancy Check
  - Link 단위로 TLP의 오류를 감지
  - Switch에서 LCRC를 확인하고 새로 계산
  - Switch 내부에서 데이터가 손상되어도, LCRC가 새롭게 갱신되면서 오류가 숨겨짐

➡ LCRC만으로는 전체적인 데이터 무결성을 보장할 수 없음

Data Integrity



- ECRC: End-to-end CRC
  - 최종 목적지에서 TLP의 오류를 감지
  - Switch는 ECRC를 변경하지 않고 유지해야 함

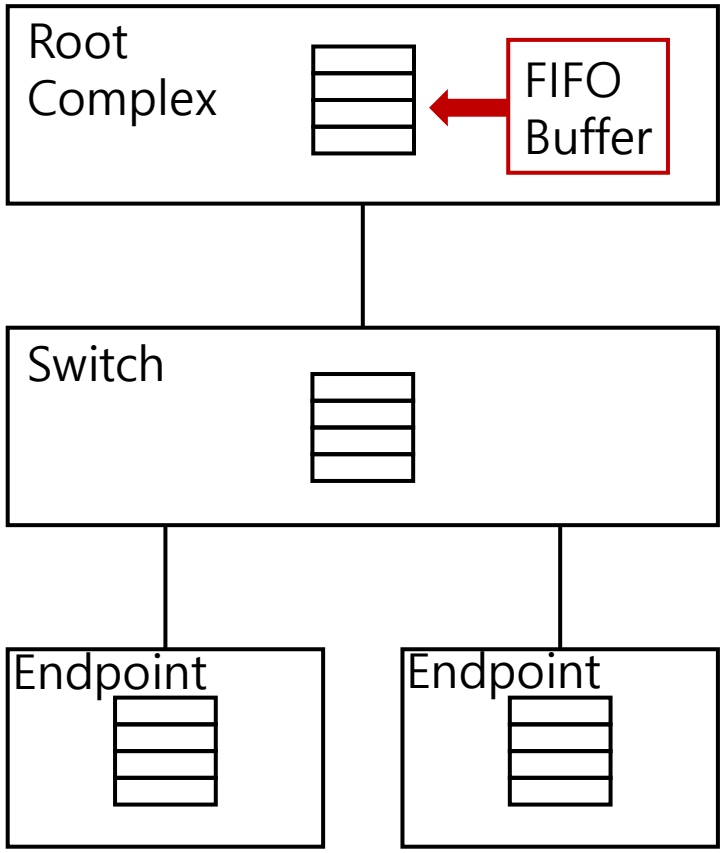


- ECRC 계산방식
  - TLP Header와 Data Payload의 모든 bit에 대해 순차대로 XOR 및 Shifting 연산 수행
  - 수신 시 동일한 방식으로 계산 후 비교하여 오류 검출

Flow Control

- Credit 기반의 Flow Control
- 다음 노드의 Credit이 부족하면 패킷 전송을 중단

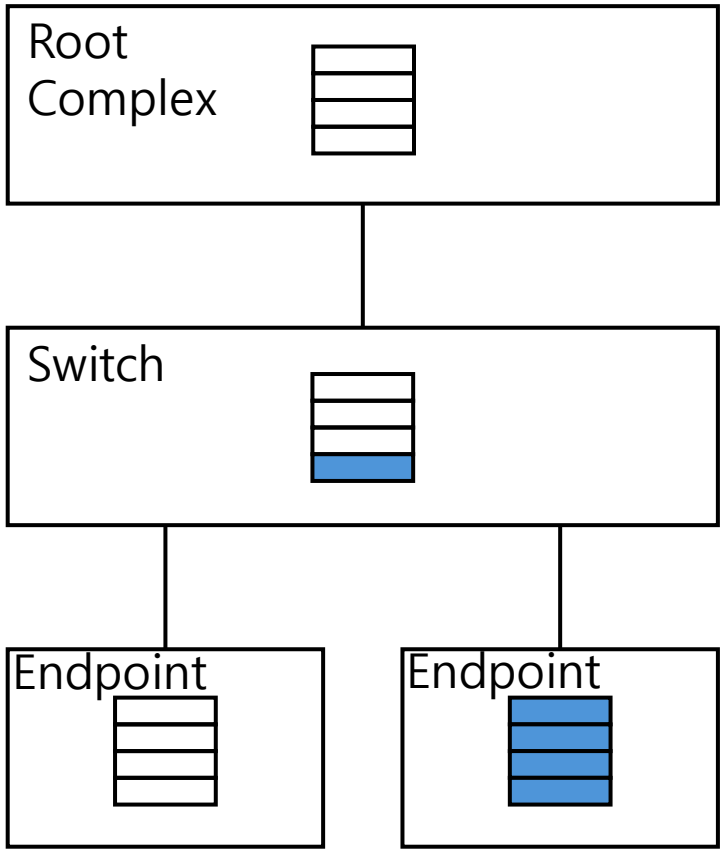
➡ 오버 플로우 방지



Flow Control

- 하나의 Buffer만 존재한다면
  Head-of-Line Blocking 문제 발생

  1. 오른쪽 Endpoint에 파란색 패킷 5개를 보냈는데 Endpoint의 Buffer가 가득 차서 패킷 1개는 Switch의 Buffer에 머무르고 있음

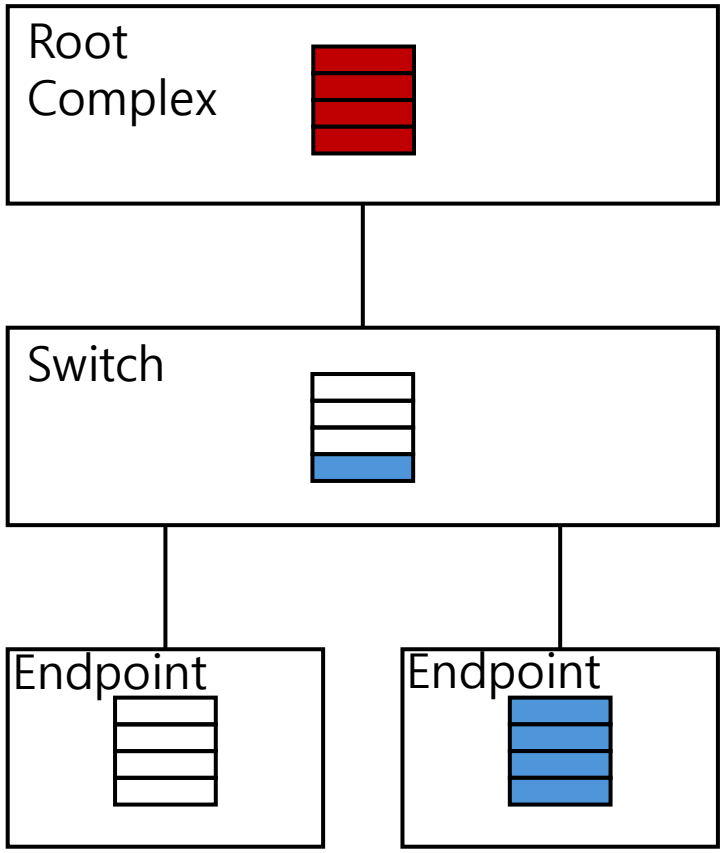


Flow Control

- 하나의 Buffer만 존재한다면
  Head-of-Line Blocking 문제 발생

  1. 오른쪽 Endpoint에 파란색 패킷 5개를 보냈는데 Endpoint의 Buffer가 가득 차서 패킷 1개는 Switch의 Buffer에 머무르고 있음

  2. 이때 왼쪽 Endpoint에 빨간색 패킷 4개를 전송하려 하면 Buffer가 FIFO이기 때문에 막혀버림

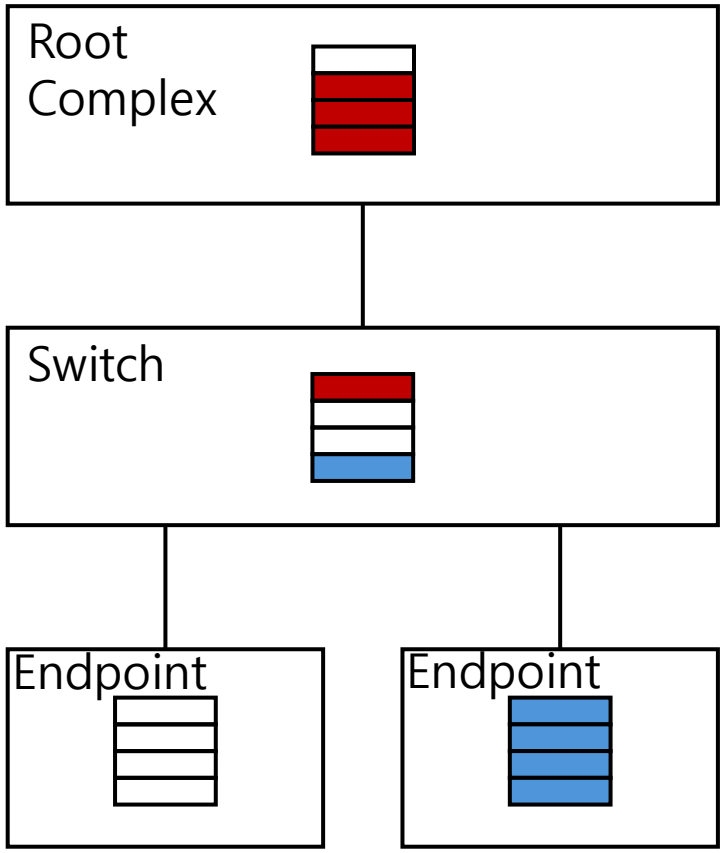


Flow Control

- 하나의 Buffer만 존재한다면
  Head-of-Line Blocking 문제 발생

  1. 오른쪽 Endpoint에 파란색 패킷 5개를 보냈는데 Endpoint의 Buffer가 가득 차서 패킷 1개는 Switch의 Buffer에 머무르고 있음

  2. 이때 왼쪽 Endpoint에 빨간색 패킷 4개를 전송하려 하면 Buffer가 FIFO이기 때문에 막혀버림

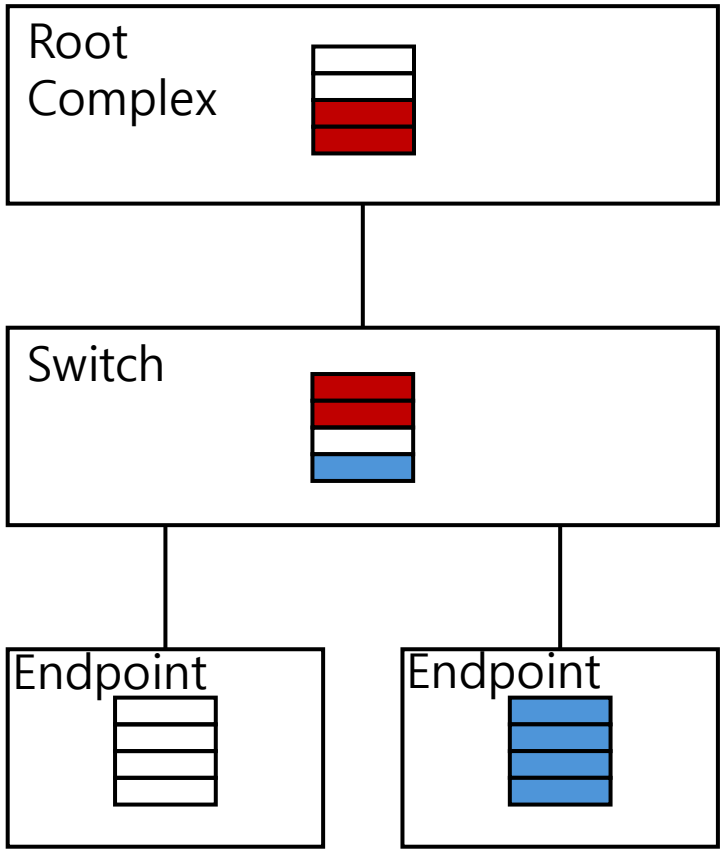


Flow Control

- 하나의 Buffer만 존재한다면
  Head-of-Line Blocking 문제 발생

  1. 오른쪽 Endpoint에 파란색 패킷 5개를 보냈는데 Endpoint의 Buffer가 가득 차서 패킷 1개는 Switch의 Buffer에 머무르고 있음

  2. 이때 왼쪽 Endpoint에 빨간색 패킷 4개를 전송하려 하면 Buffer가 FIFO이기 때문에 막혀버림



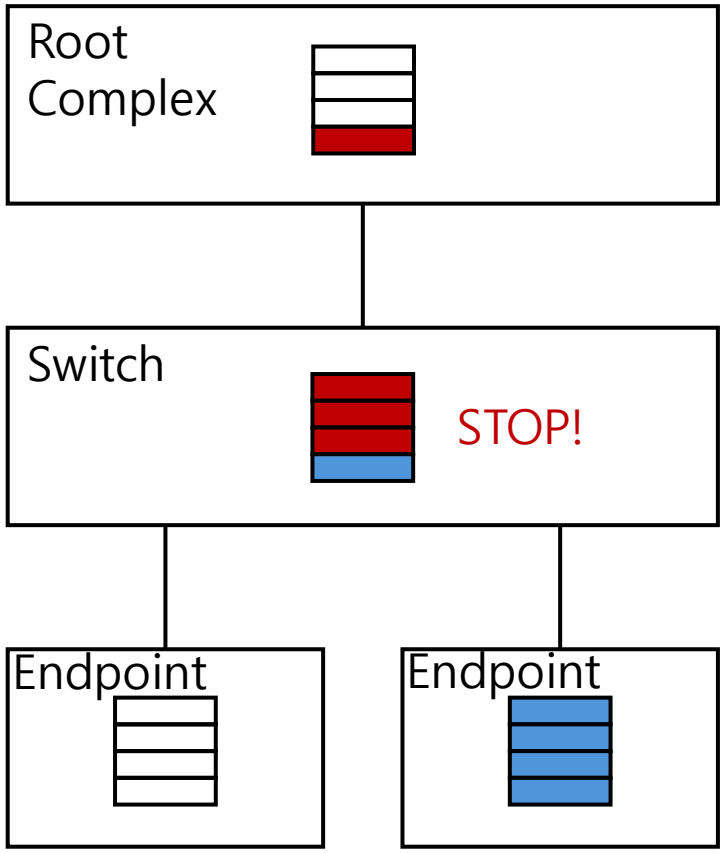
Flow Control

- 하나의 Buffer만 존재한다면
  Head-of-Line Blocking 문제 발생

  1. 오른쪽 Endpoint에 파란색 패킷 5개를 보냈는데 Endpoint의 Buffer가 가득 차서 패킷 1개는 Switch의 Buffer에 머무르고 있음

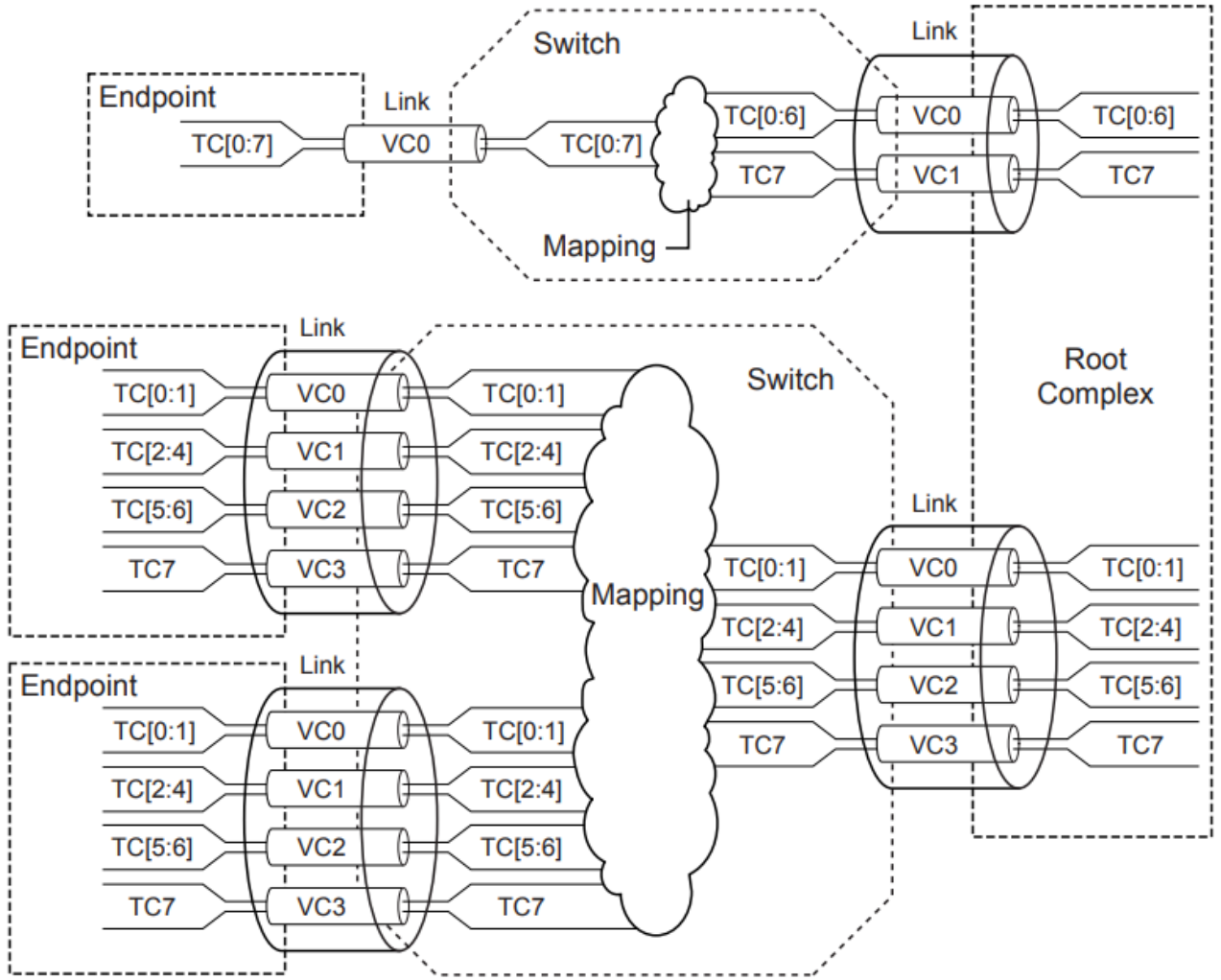
  2. 이때 왼쪽 Endpoint에 빨간색 패킷 4개를 전송하려 하면 Buffer가 FIFO이기 때문에 막혀버림

  ➡ Virtual Channel 필요



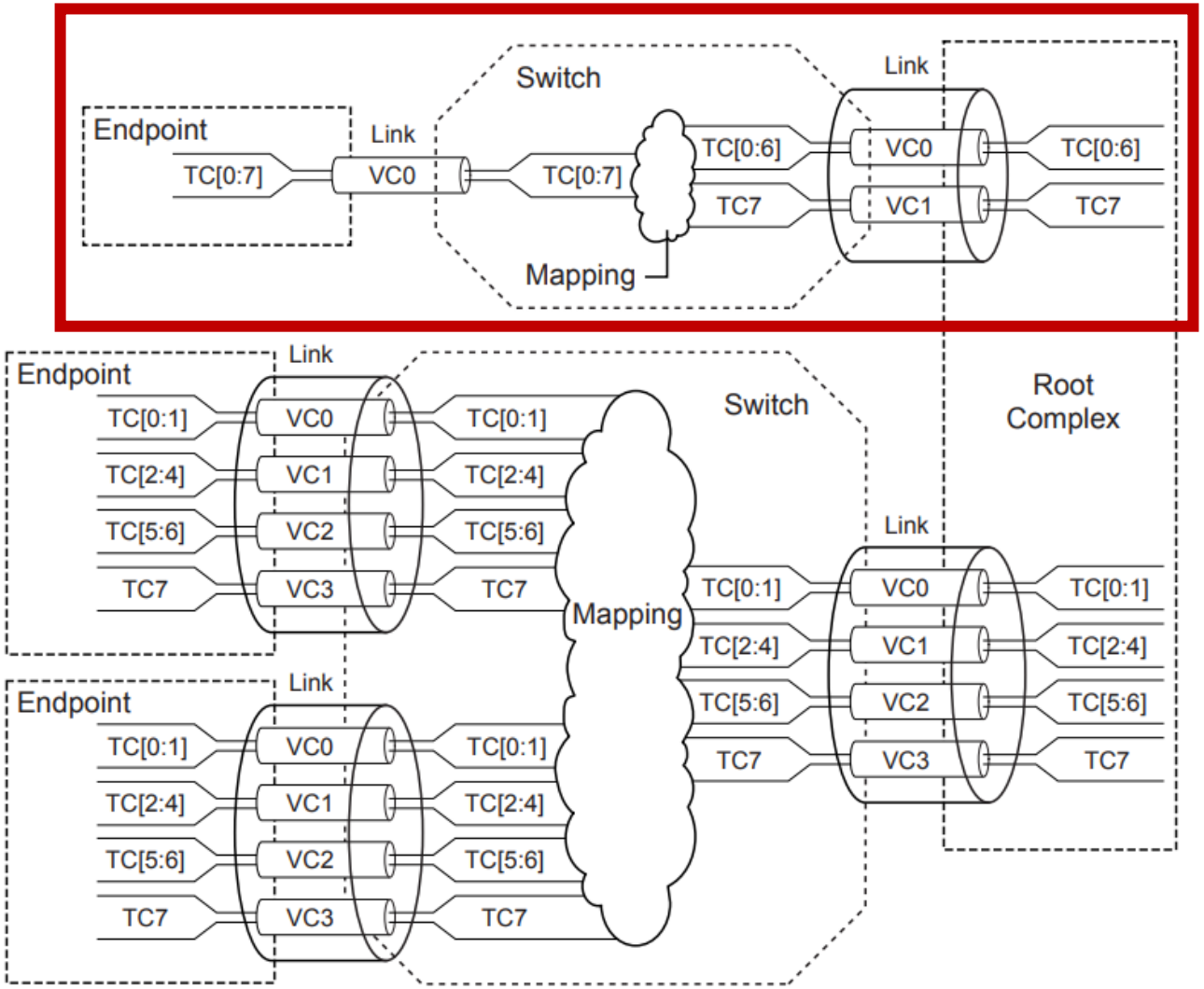
Virtual Channel

- TC(Traffic Class): 패킷 전송의 우선 순위를 결정
- 0~7까지의 TC값에 따라 VC(Virtual Channel)이 정해짐



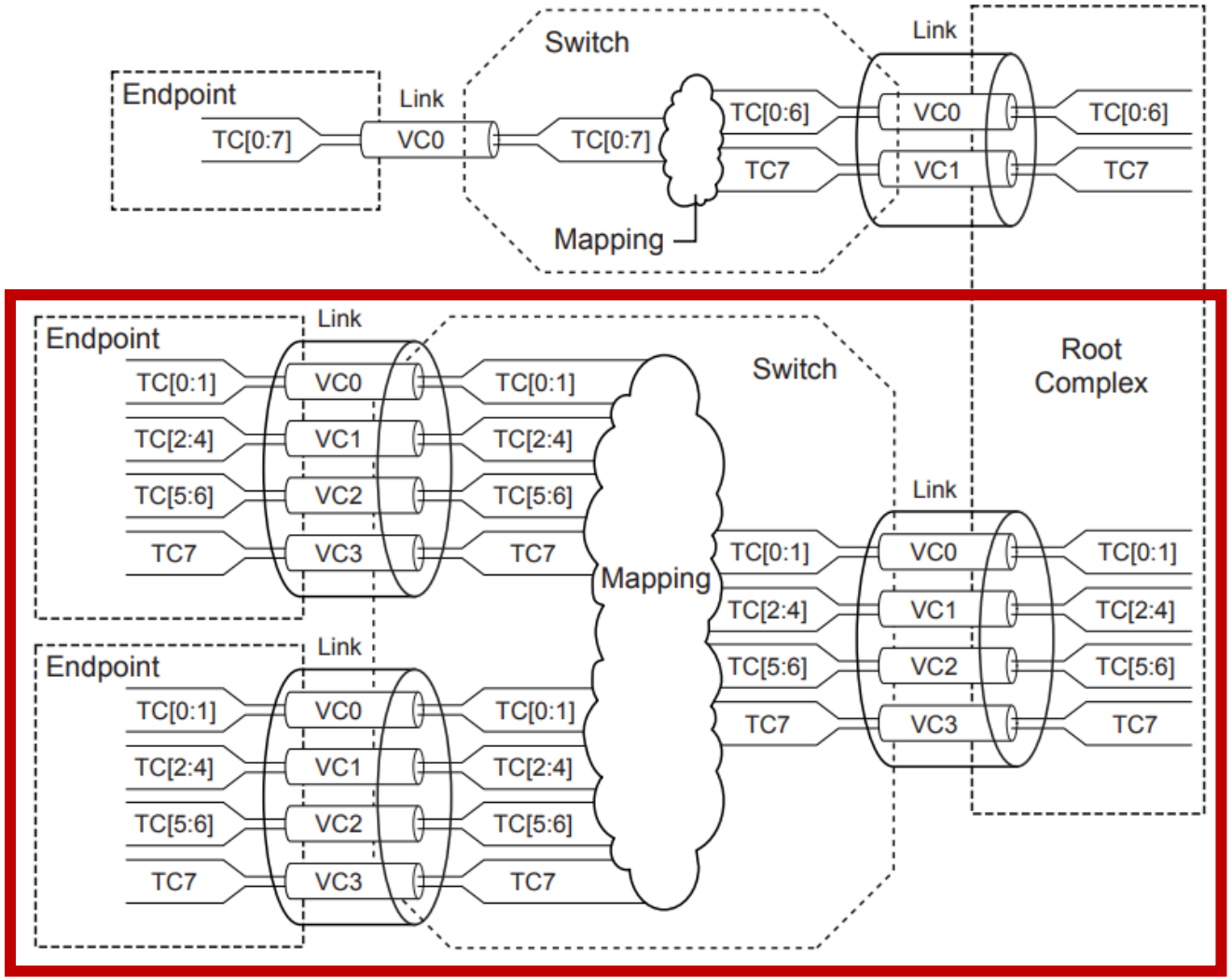
Virtual Channel

- TC(Traffic Class): 패킷 전송의 우선 순위를 결정
- 0~7까지의 TC값에 따라 VC(Virtual Channel)이 정해짐



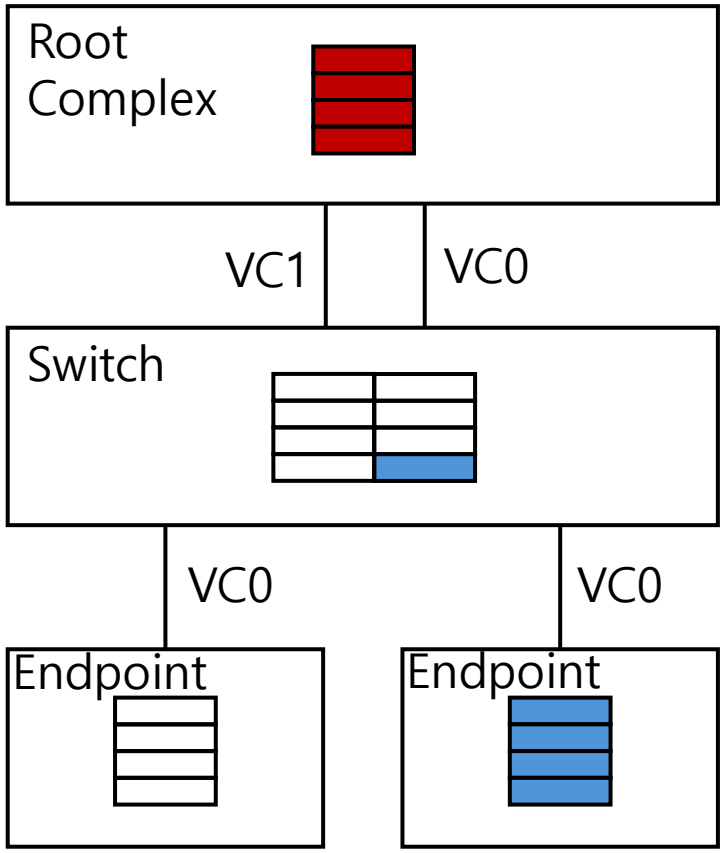
Virtual Channel

- TC(Traffic Class): 패킷 전송의 우선 순위를 결정
- 0~7까지의 TC값에 따라 VC(Virtual Channel)이 정해짐



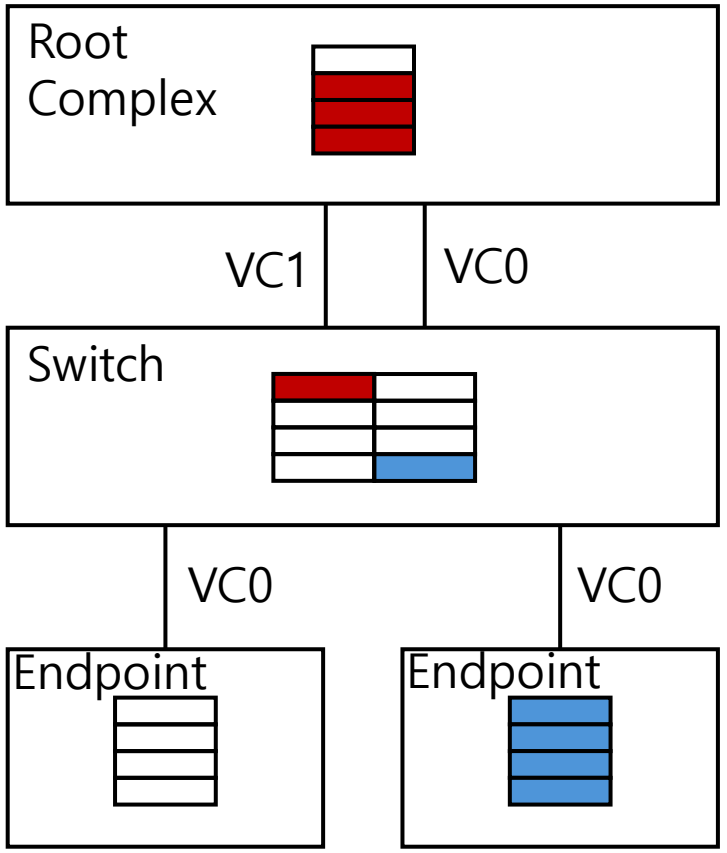
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



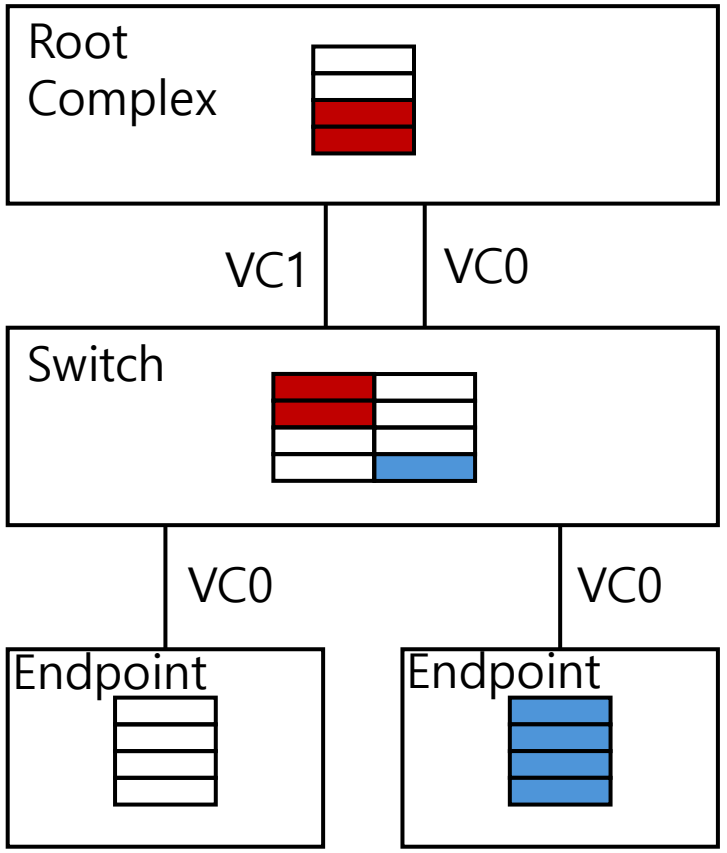
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



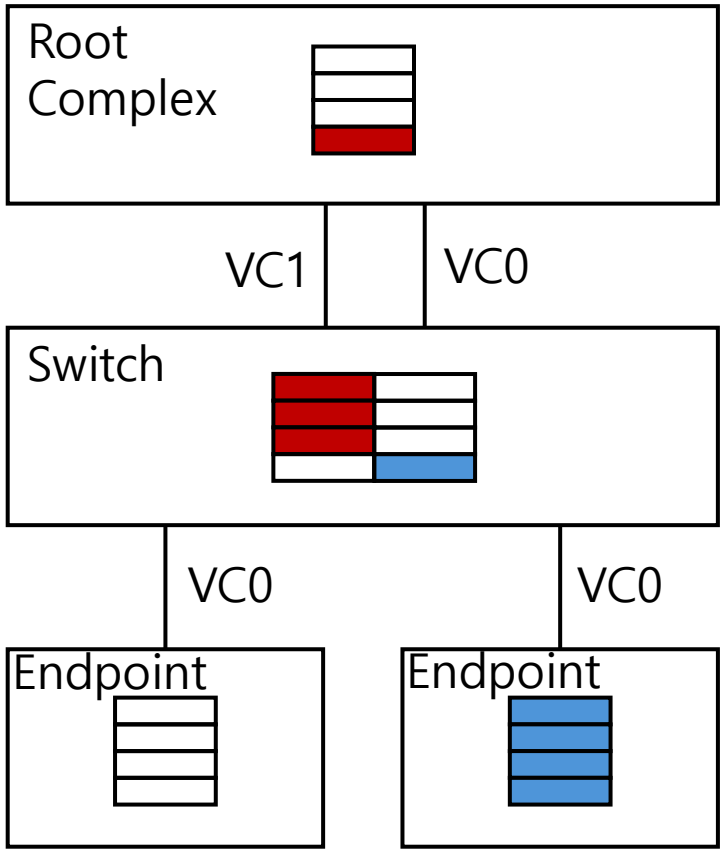
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



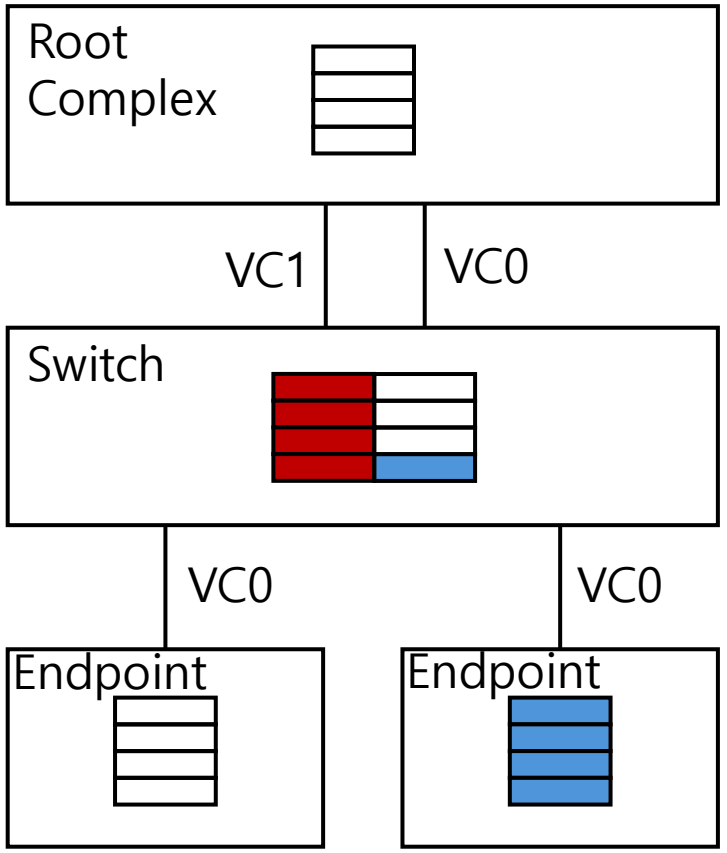
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



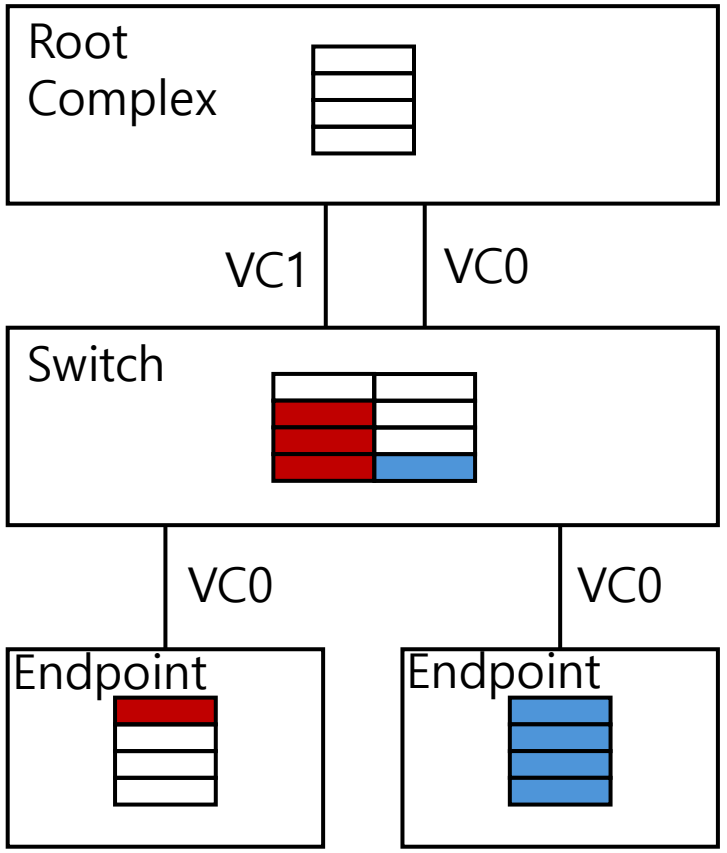
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



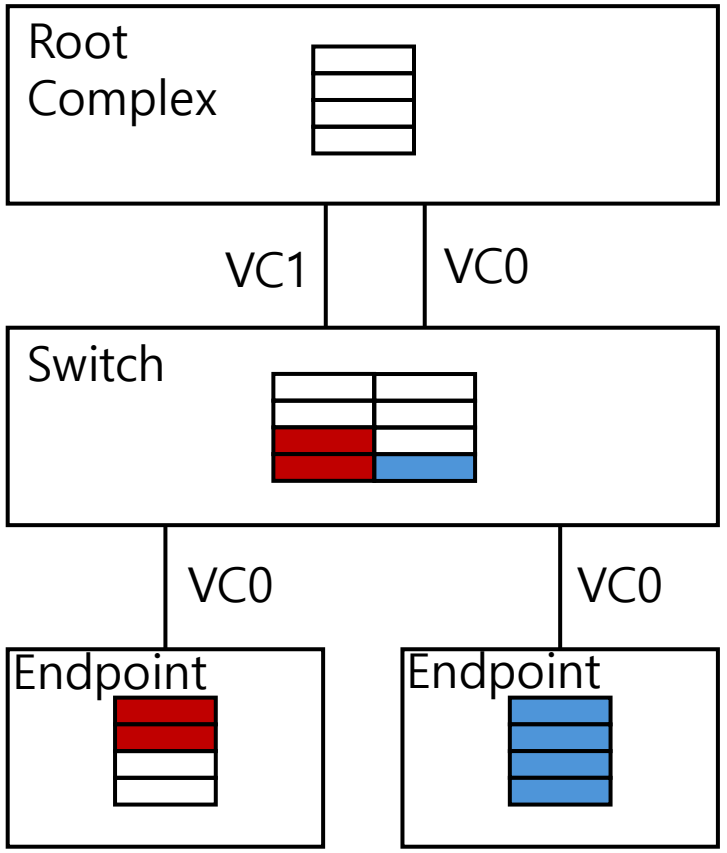
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



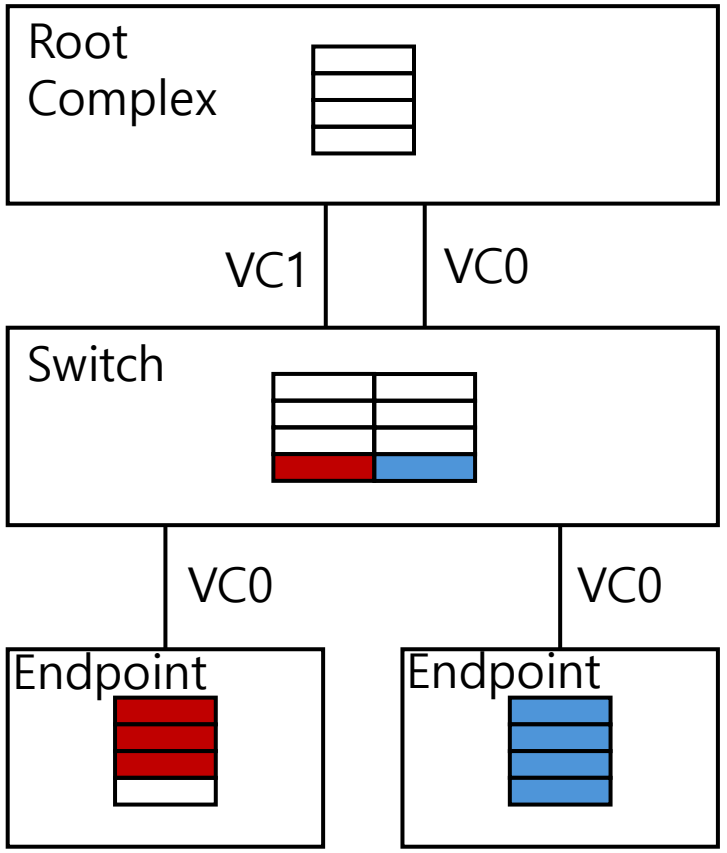
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



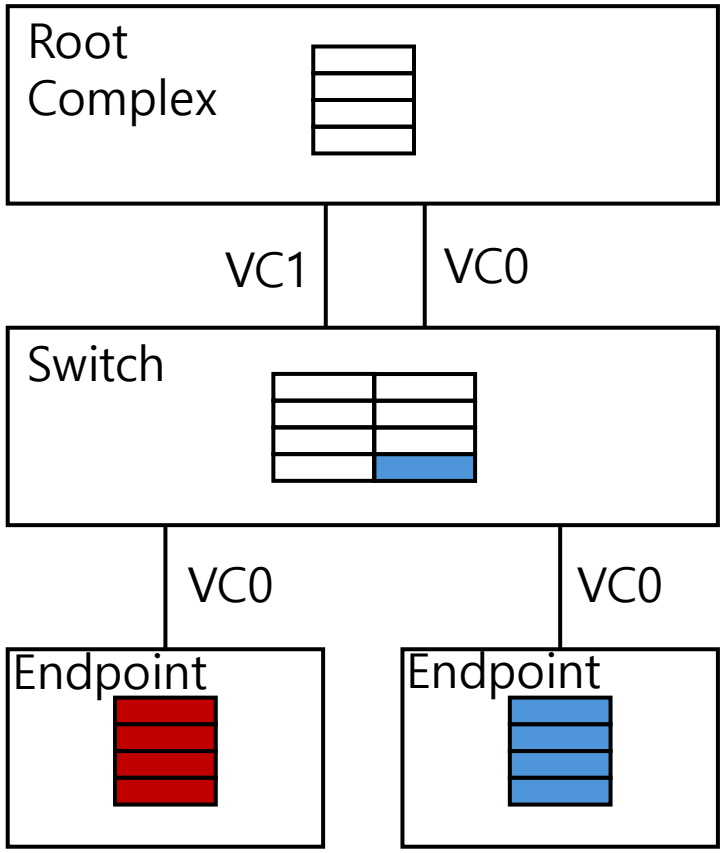
Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



Virtual Channel

- Head-of-line Blocking 문제 해결
  - 각 VC별로 독립적인 버퍼를 가짐
  - 한 VC의 패킷이 정체되어 있어도 다른 VC의 패킷은 정상적으로 전송될 수 있음



Scalable
Archi
Lab.

Thank You