SmartNICs SUMMIT
April 26-28, 2022
DoubleTree by Hilton San Jose
SmartNICsSummit.com

PANIC: A High-Performance Programmable NIC for Multi-tenant Networks

Jiaxin Lin[1], Kiran Patel, Brent E. Stephens[2],
Anirudh Sivaraman[3] and Aditya Akella[1]

TEXAS
The University of Texas at Austin
1

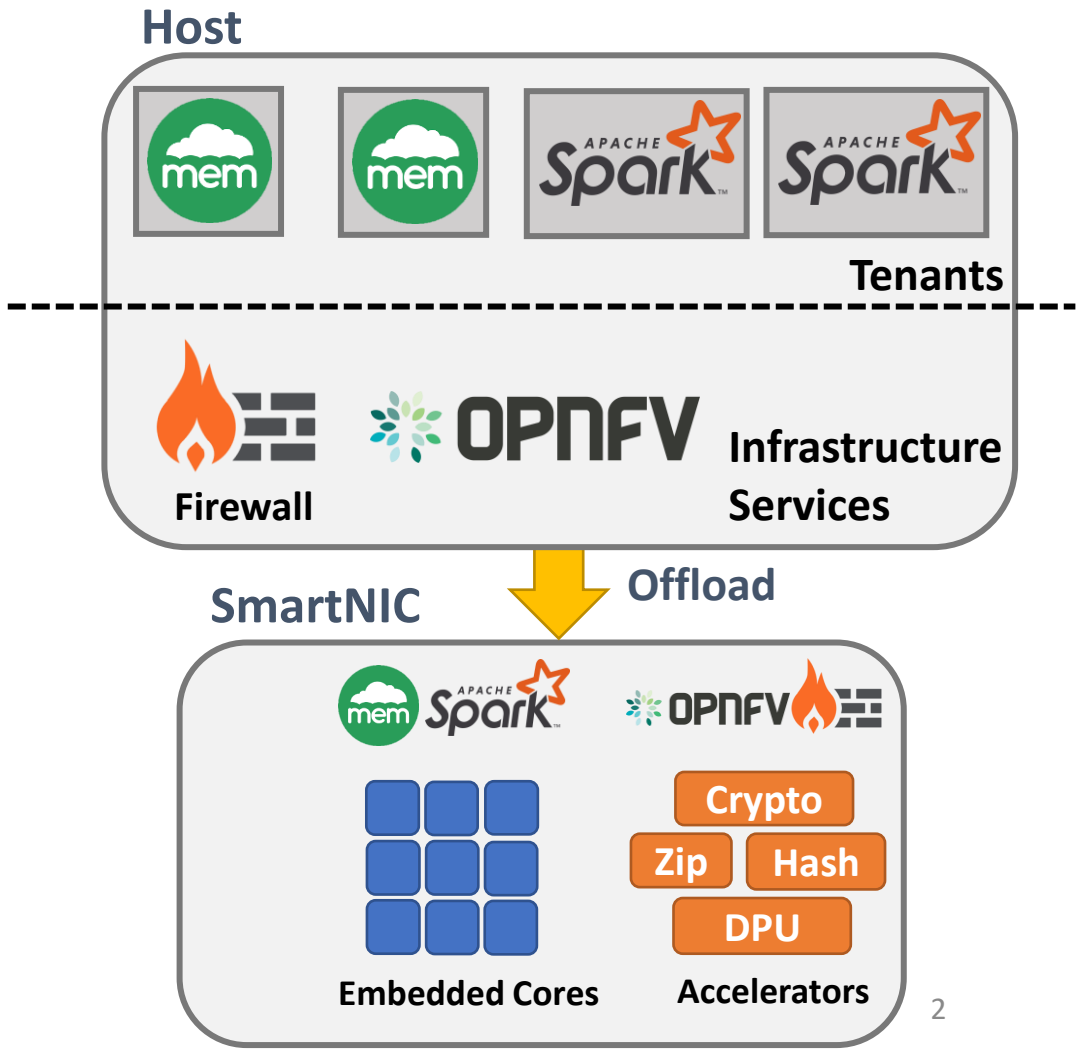
THE UNIVERSITY OF UTAH
2

NYU
3

SmartNIC and Multi Tenancy

- SmartNICs can help drive increasing network line-rates (100Gbps+) by offloading applications or cloud services

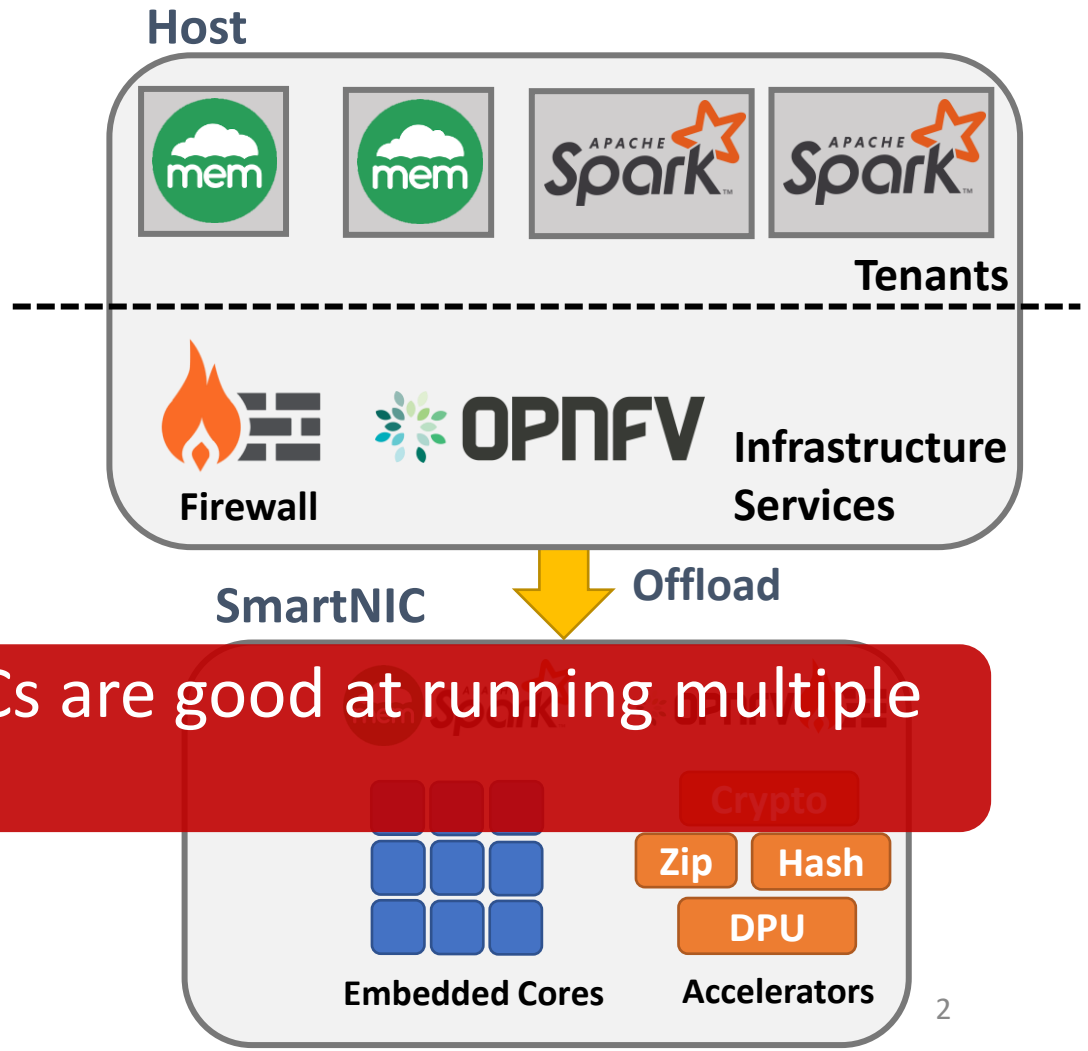
- In the multi-tenant environment, to get benefits from the SmartNIC, servers may want to run multiple offloads on the SmartNIC.



SmartNIC and Multi Tenancy

- SmartNICs can help drive increasing network line-rates (100Gbps+) by offloading applications or cloud services

- In the multi-tenant environment, to get benefits from the SmartNIC, servers may want to run multiple offloads on the SmartNIC.

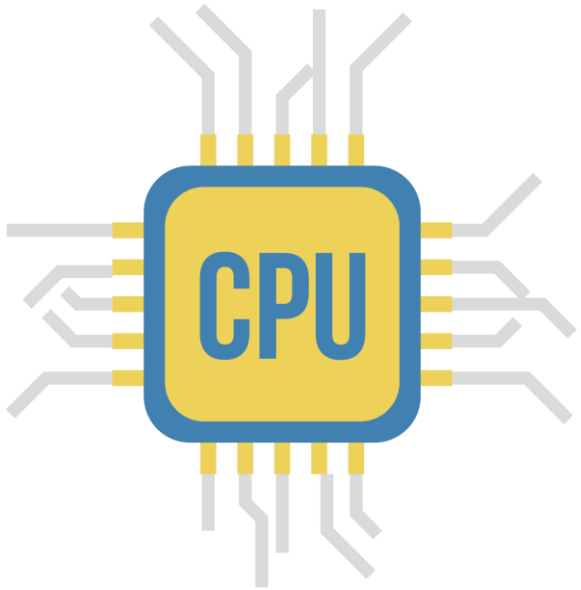


Problem: None of the current SmartNICs are good at running multiple tenants' offloads at the same time.

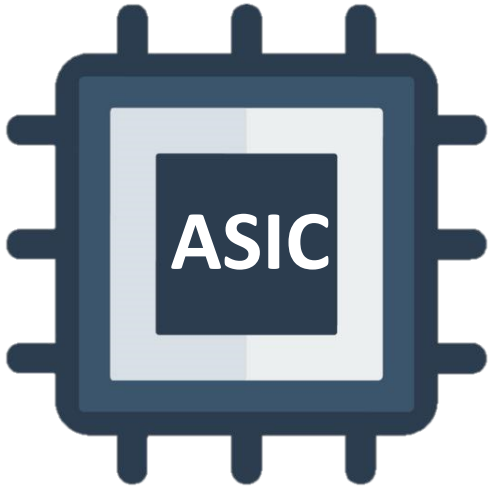
What are the requirements for a SmartNIC in a multi-tenant environment?

Requirements # 1 Generality

- **Generality**: Different tenants on the host may requires different types of offloads.
  - Both ASIC offload and CPU core should be supported
  - Offload may have below line rate/variable performance

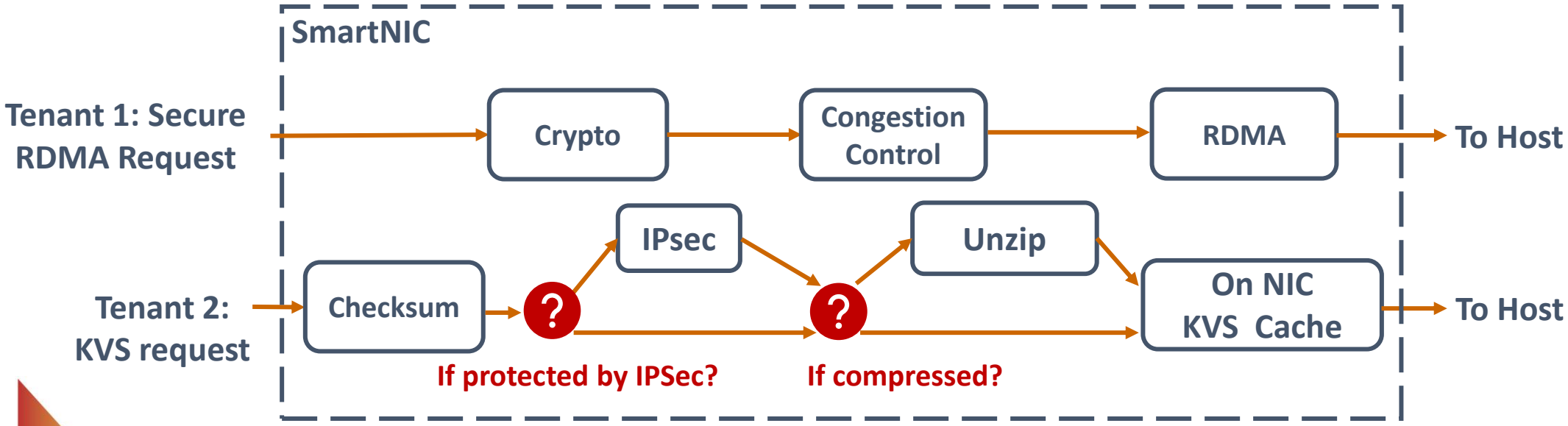


OR?



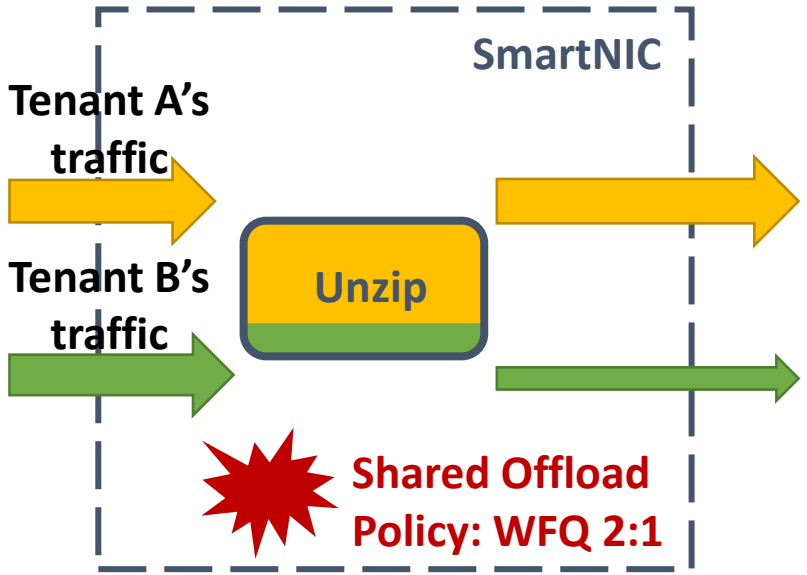
Requirements # 2 Flexible Chaining

- **Flexible Chaining:**
  - Different tenants will specify their own chains of offloads.
  - NIC should support sending packets through offloads in any order.



Requirements # 3 Isolation
4 Performance

- **Isolation:**
  - SmartNIC should provide performance isolation between competing tenants.
- **Performance:**
  - SmartNIC should provide high throughput for line-rate offloads.
  - SmartNIC should not incur additional latency for low latency offload.



Motivation: Build a programmable NIC that meets all these requirements!

Outline

Motivation

PANIC Design

Implementation & Evaluation

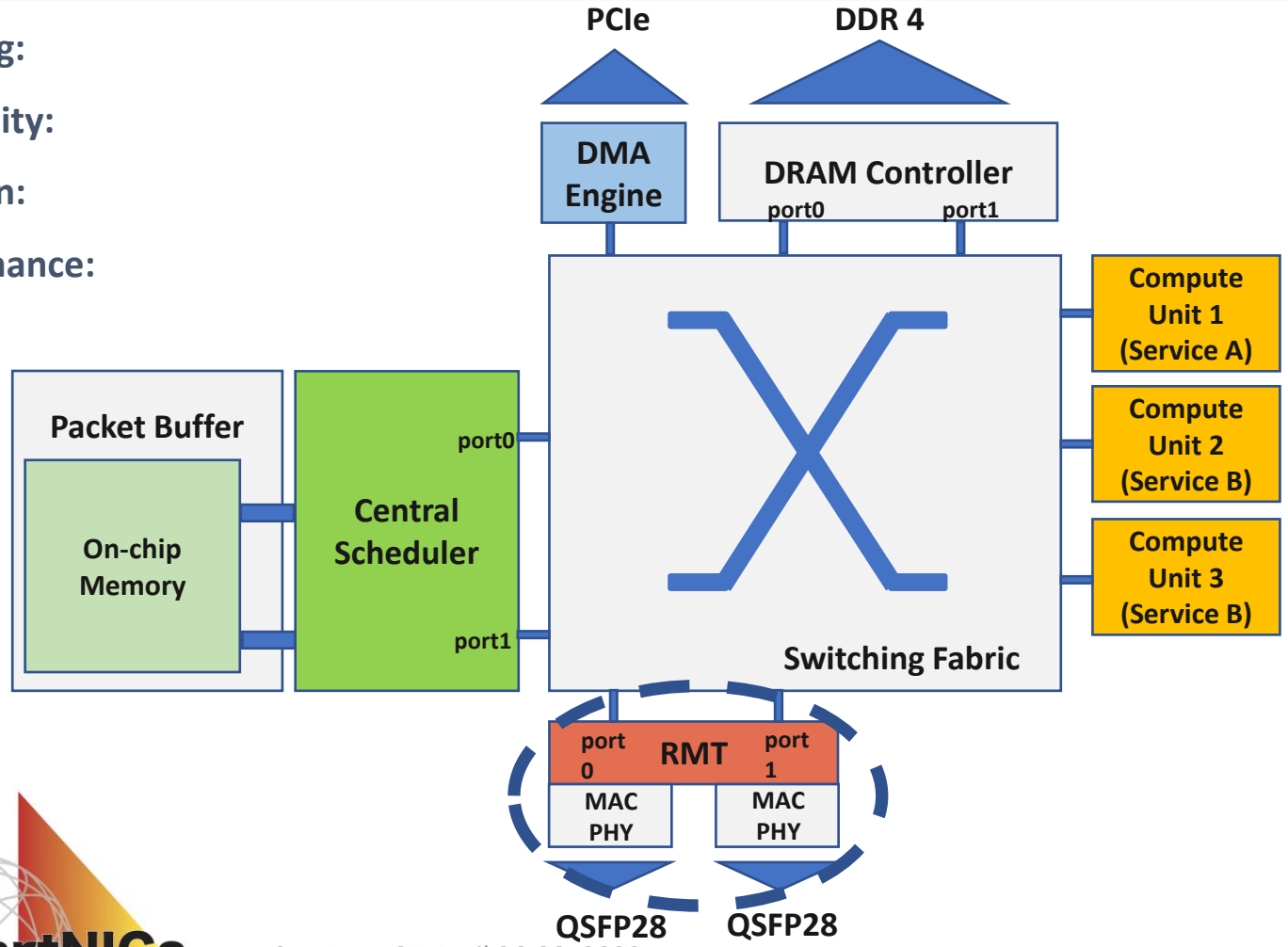
PANIC Design Overview

Chaining:

Generality:

Isolation:

Performance:



PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain

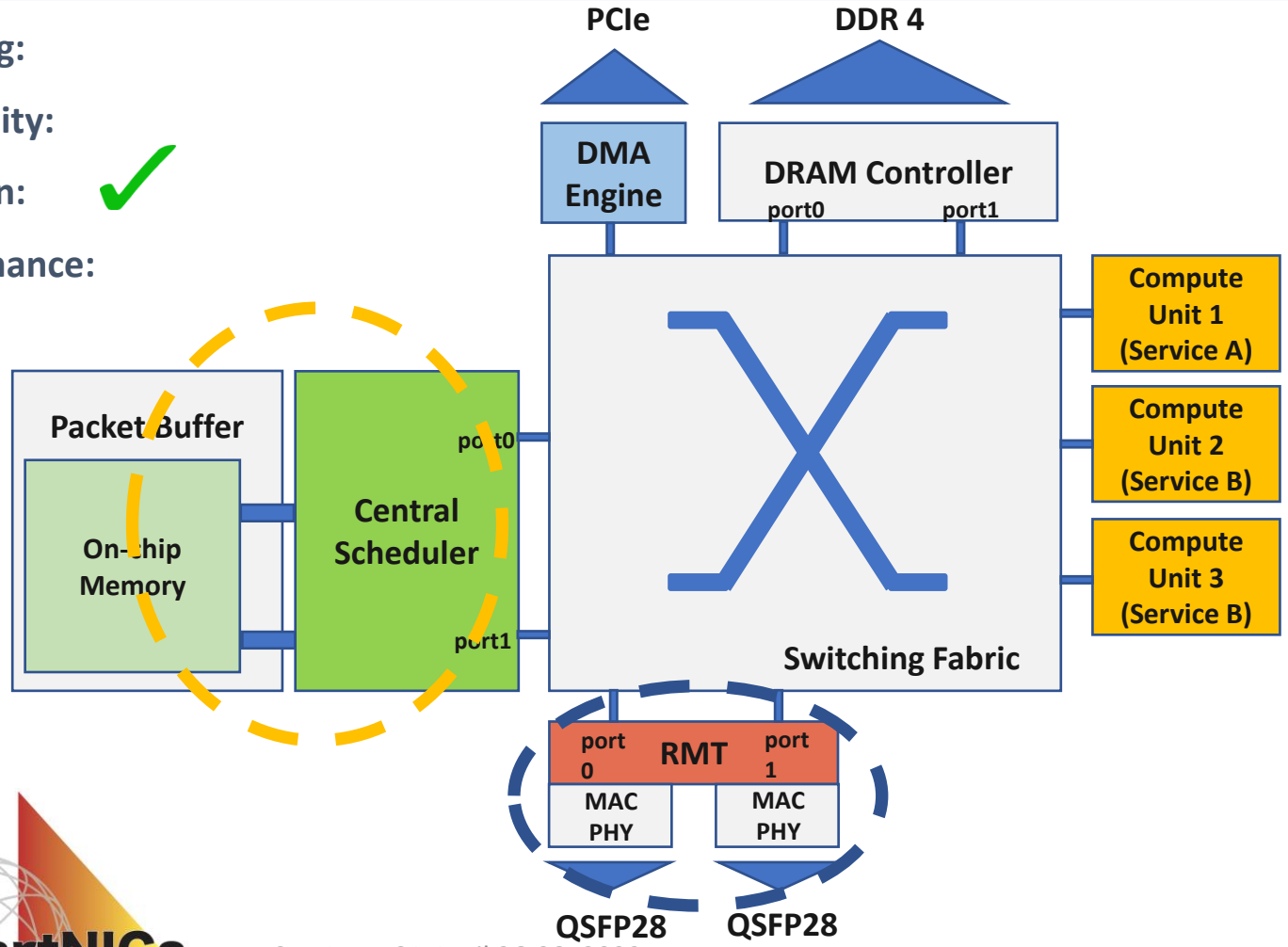
PANIC Design Overview

Chaining:

Generality:

Isolation: ✓

Performance:



PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain
2. **Central Scheduler:** enforce isolation policies and schedule packets

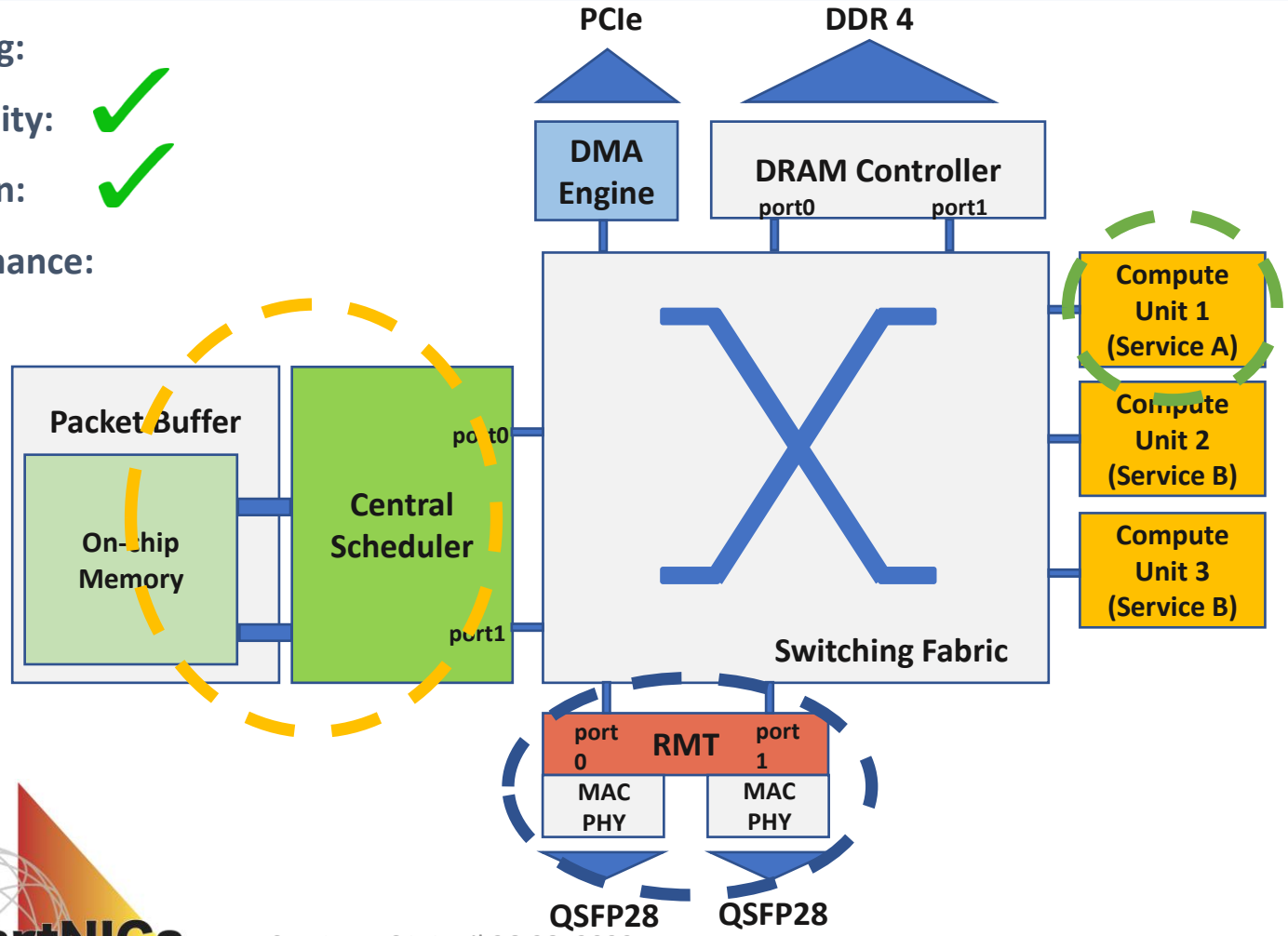
PANIC Design Overview

Chaining:

Generality: ✓

Isolation: ✓

Performance:



PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain
2. **Central Scheduler:** enforce isolation policies and schedule packets
3. **Independent Compute Unit:** Support hardware accelerator or CPU core

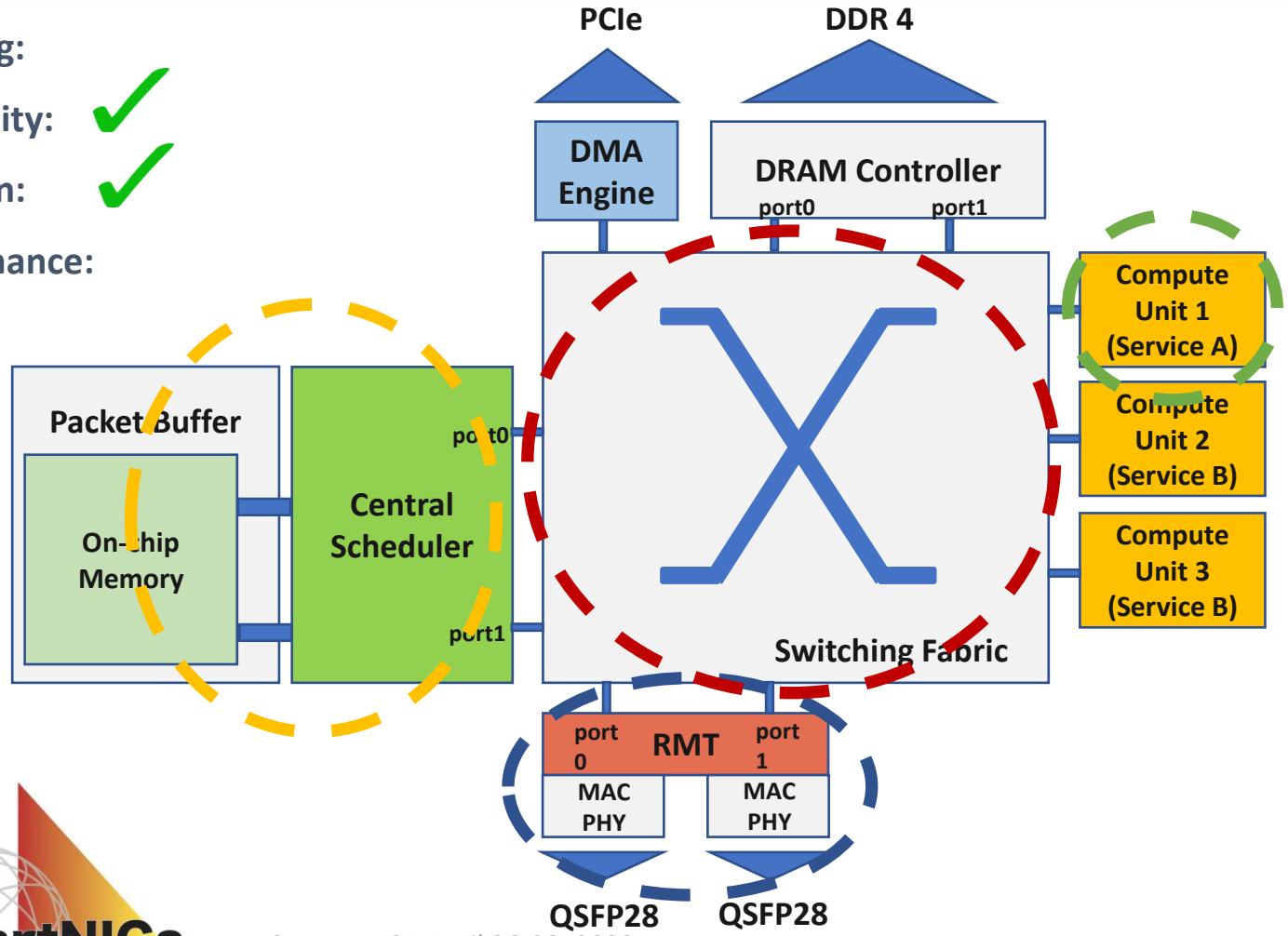
PANIC Design Overview

Chaining:

Generality: ✓

Isolation: ✓

Performance:



PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain
2. **Central Scheduler:** enforce isolation policies and schedule packets
3. **Independent Compute Unit:** Support hardware accelerator or CPU core
4. **High-throughput Switching Fabric:** Interconnects different hardware resources.

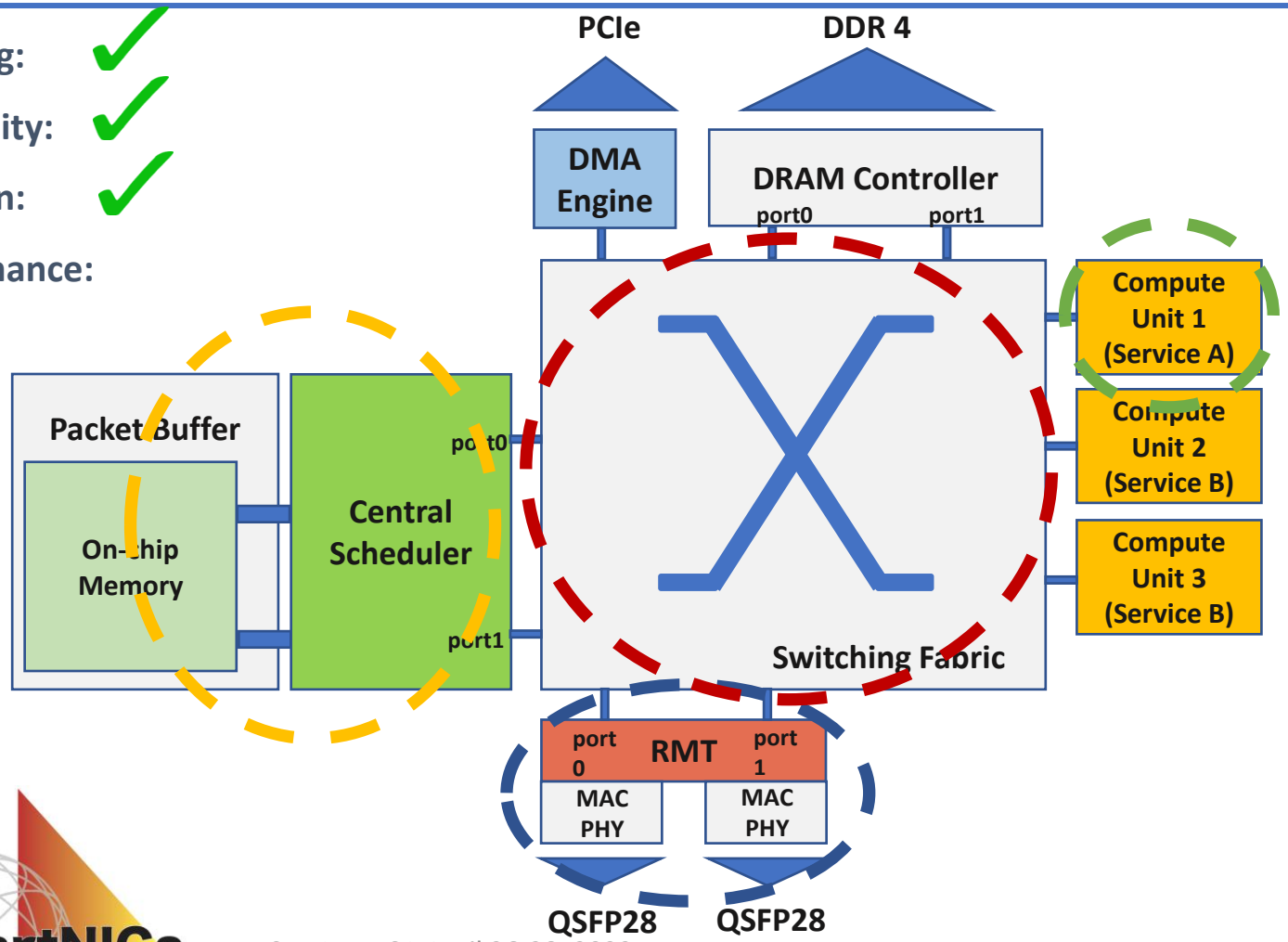
PANIC Design Overview

Chaining: ✓

Generality: ✓

Isolation: ✓

Performance:

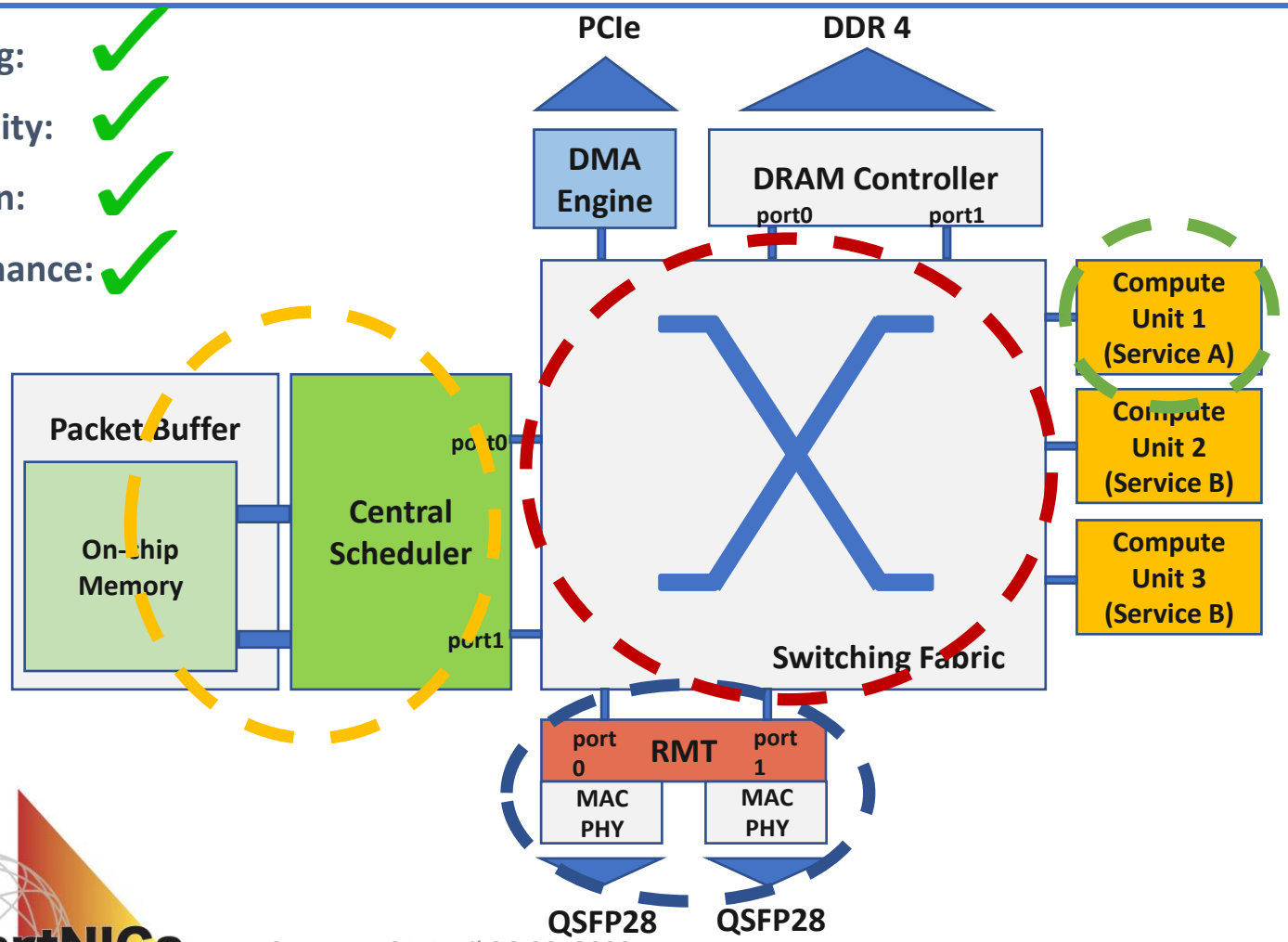


PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain
2. **Central Scheduler:** enforce isolation policies and schedule packets
3. **Independent Compute Unit:** Support hardware accelerator or CPU core
4. **High-throughput Switching Fabric:** Interconnects different hardware resources.

PANIC Design Overview

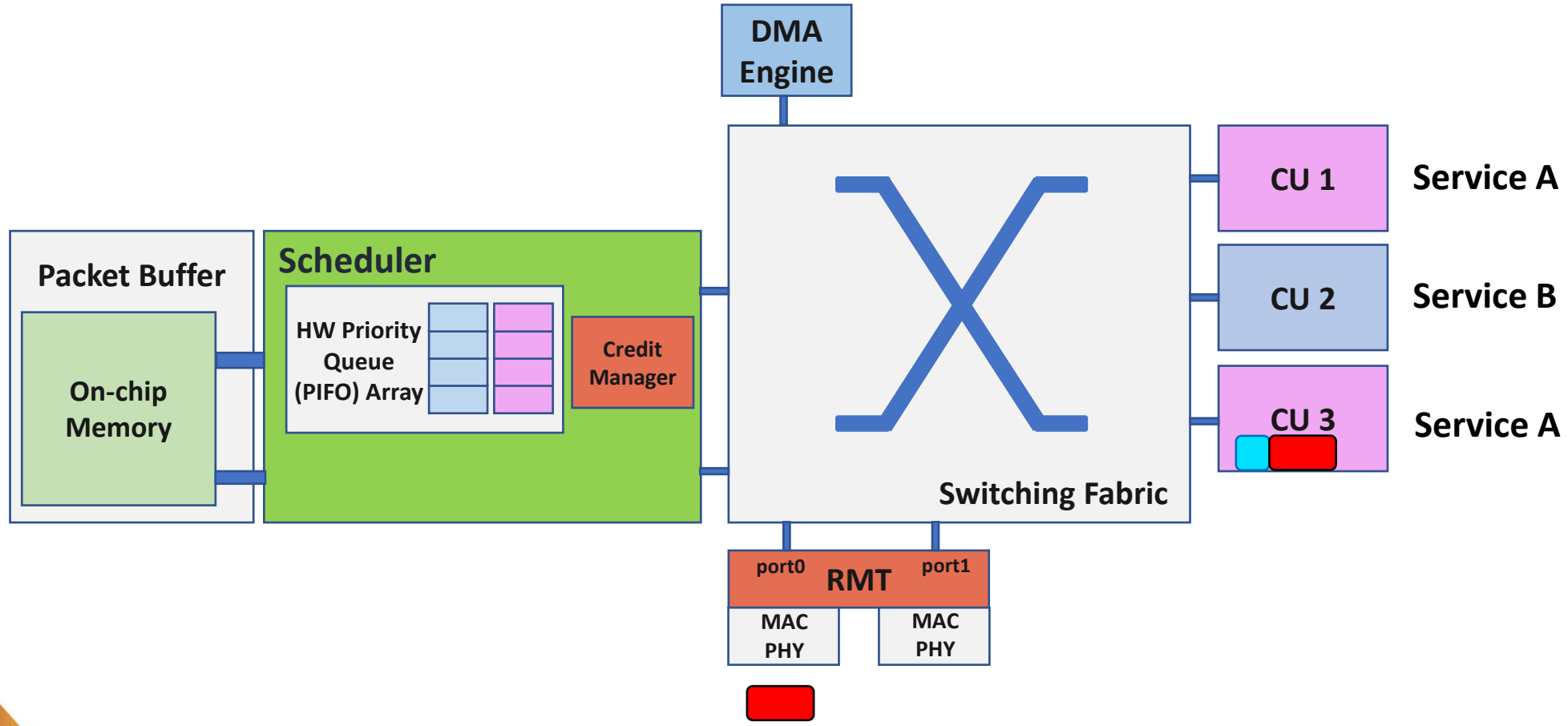
Chaining: ✓
Generality: ✓
Isolation: ✓
Performance: ✓



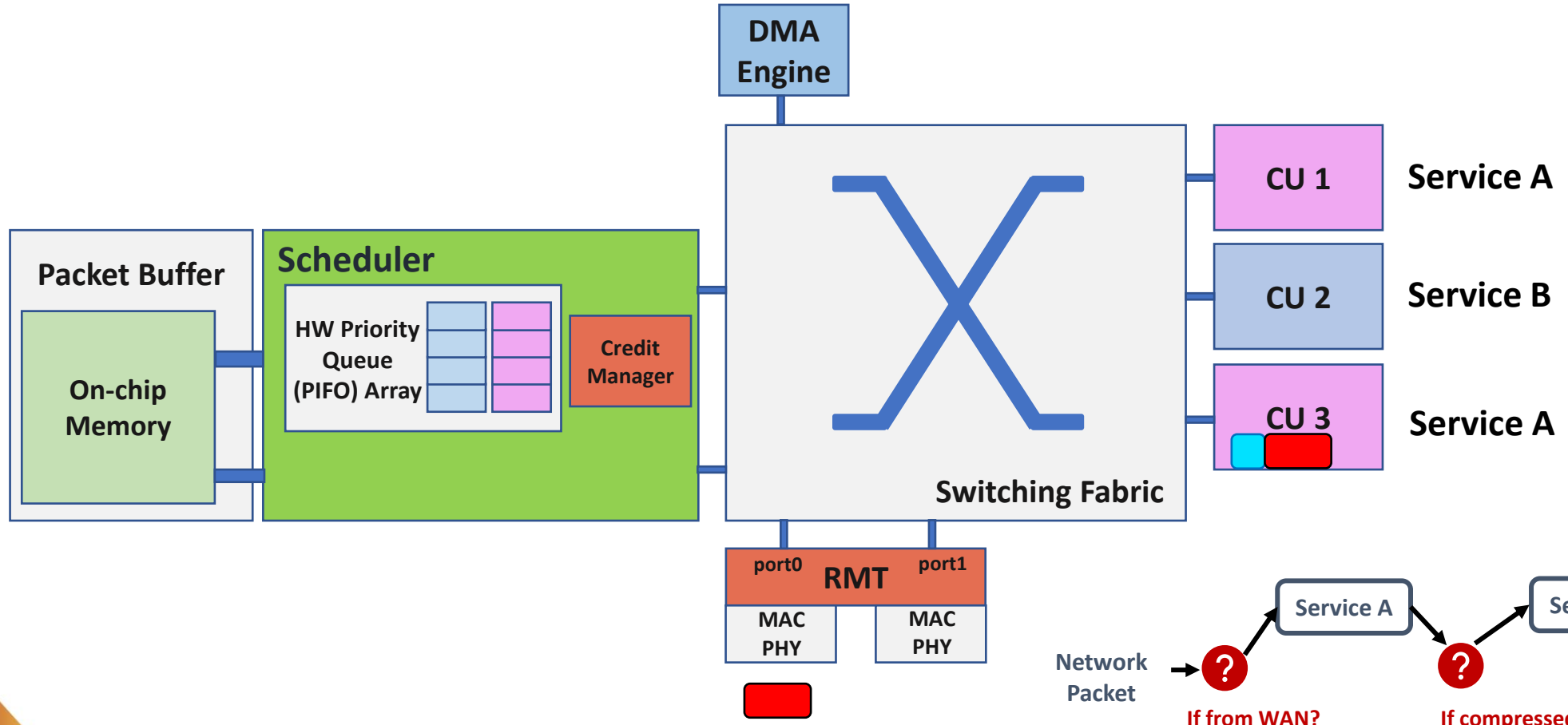
PANIC Components:

1. **Reconfigurable-Match-Action Pipeline**: Parse packets and determine offload chain
2. **Central Scheduler:** enforce isolation policies and schedule packets
3. **Independent Compute Unit:** Support hardware accelerator or CPU core
4. **High-throughput Switching Fabric:** Interconnects different hardware resources.

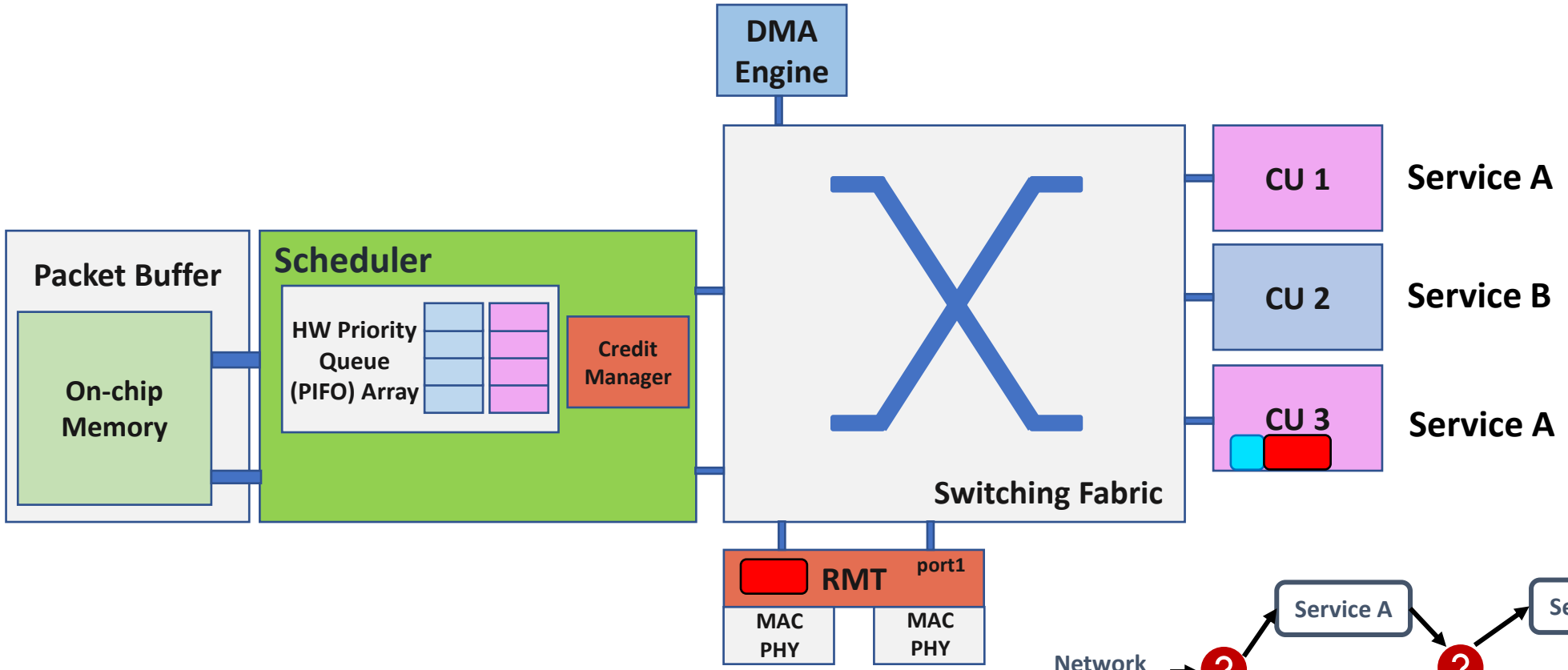
Life-Cycle of a Packet in PANIC

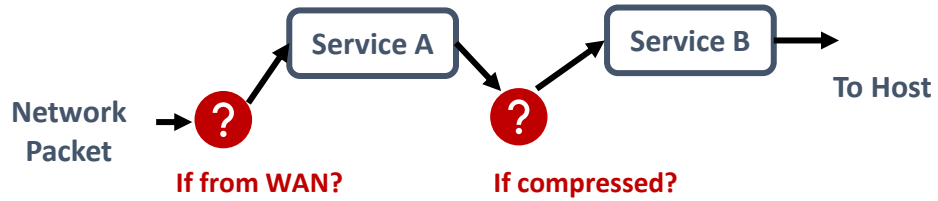


Life-Cycle of a Packet in PANIC

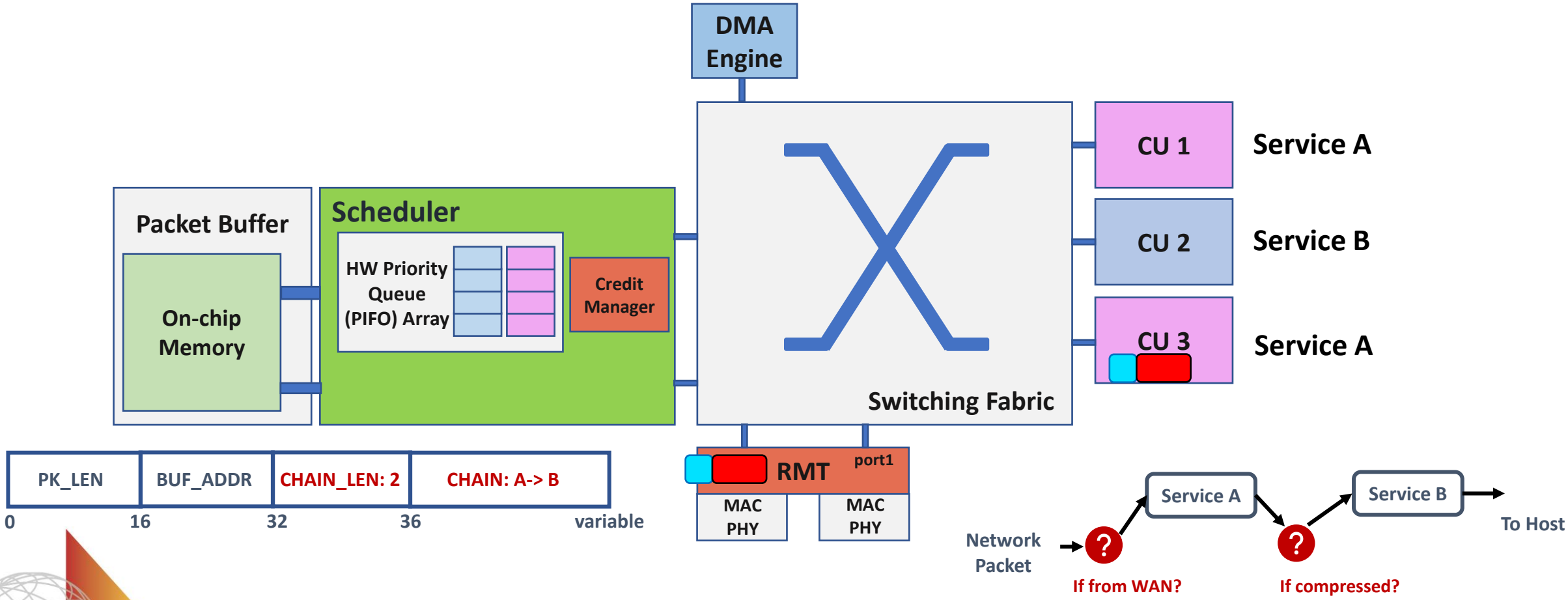


Life-Cycle of a Packet in PANIC

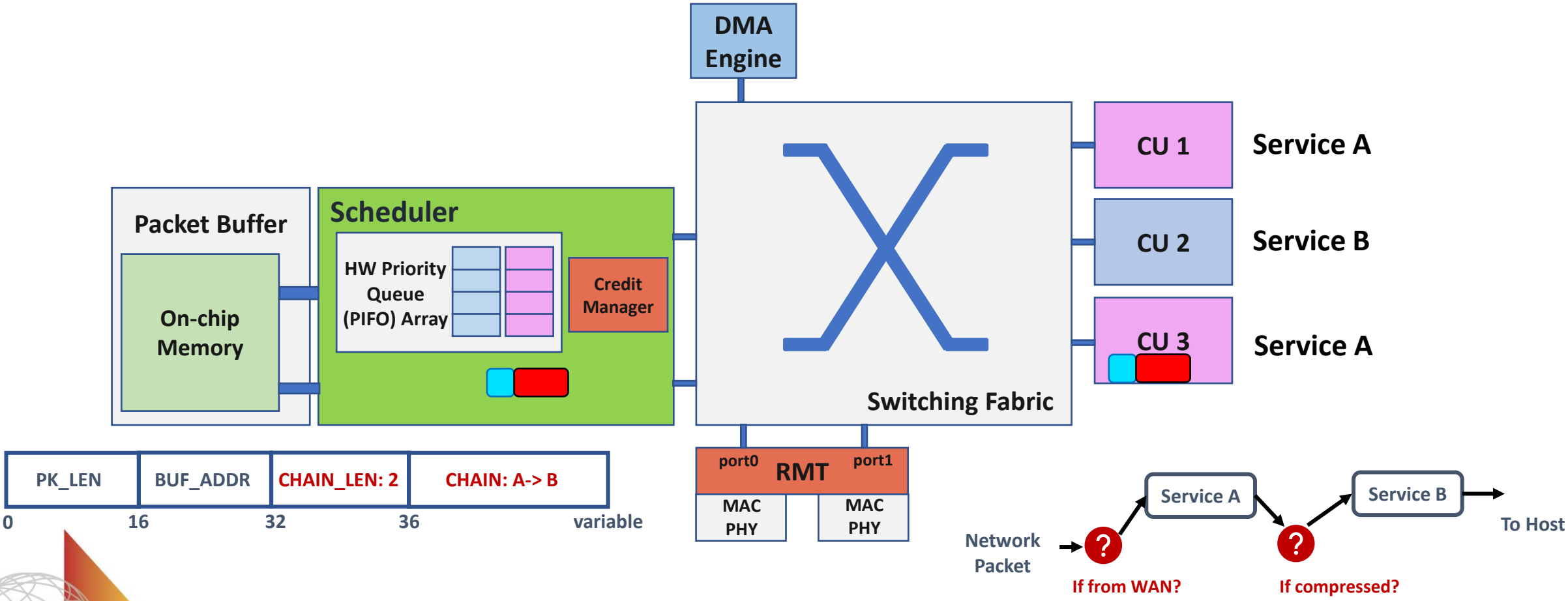




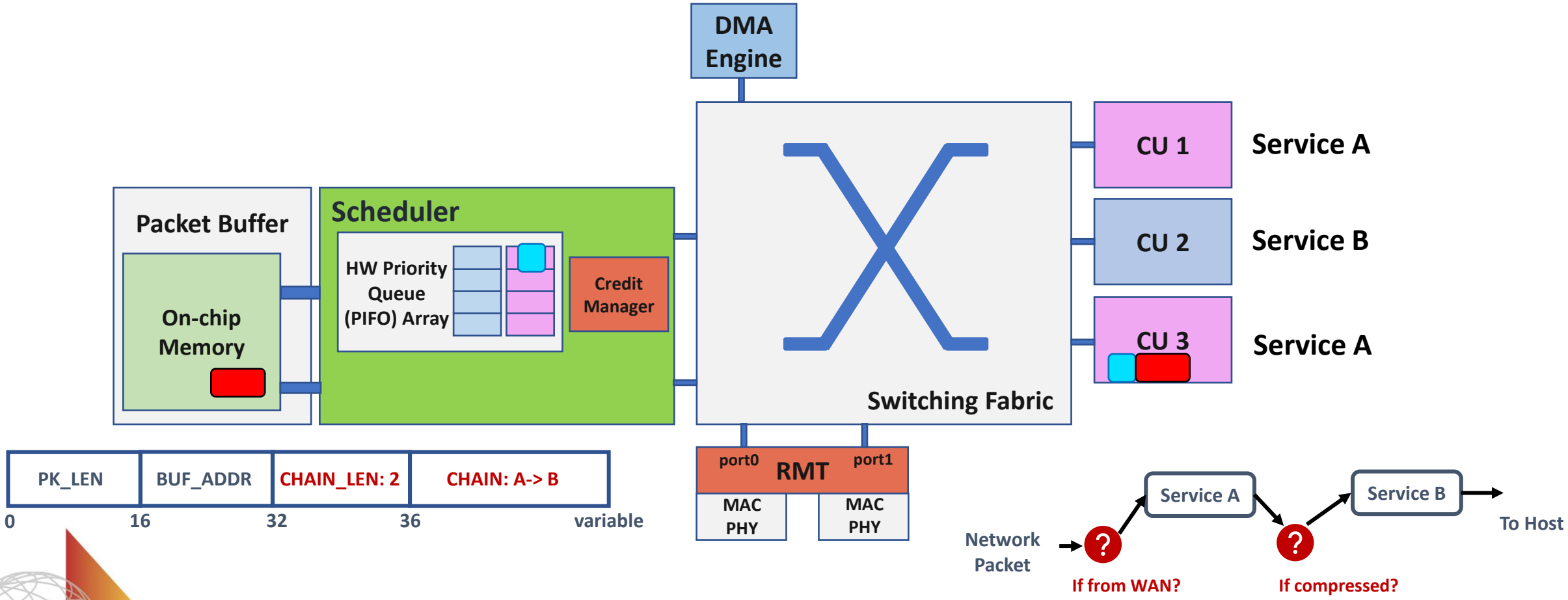
Life-Cycle of a Packet in PANIC



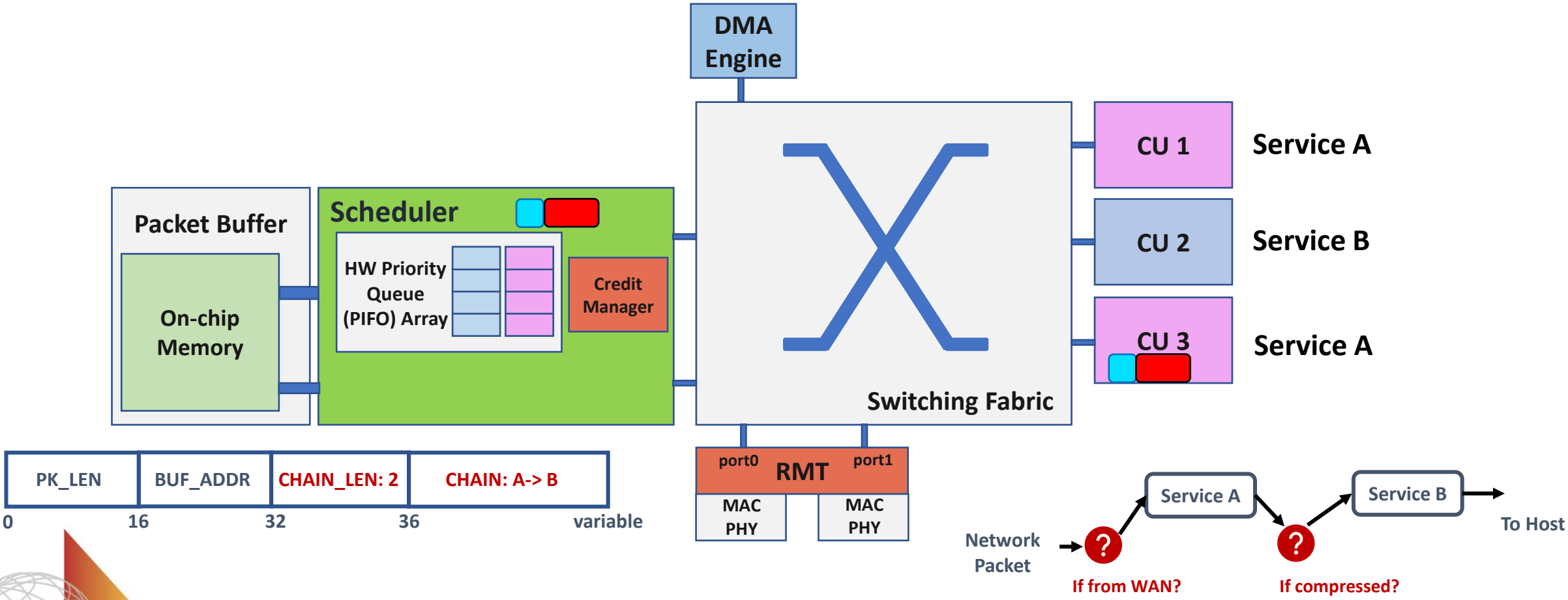
Life-Cycle of a Packet in PANIC



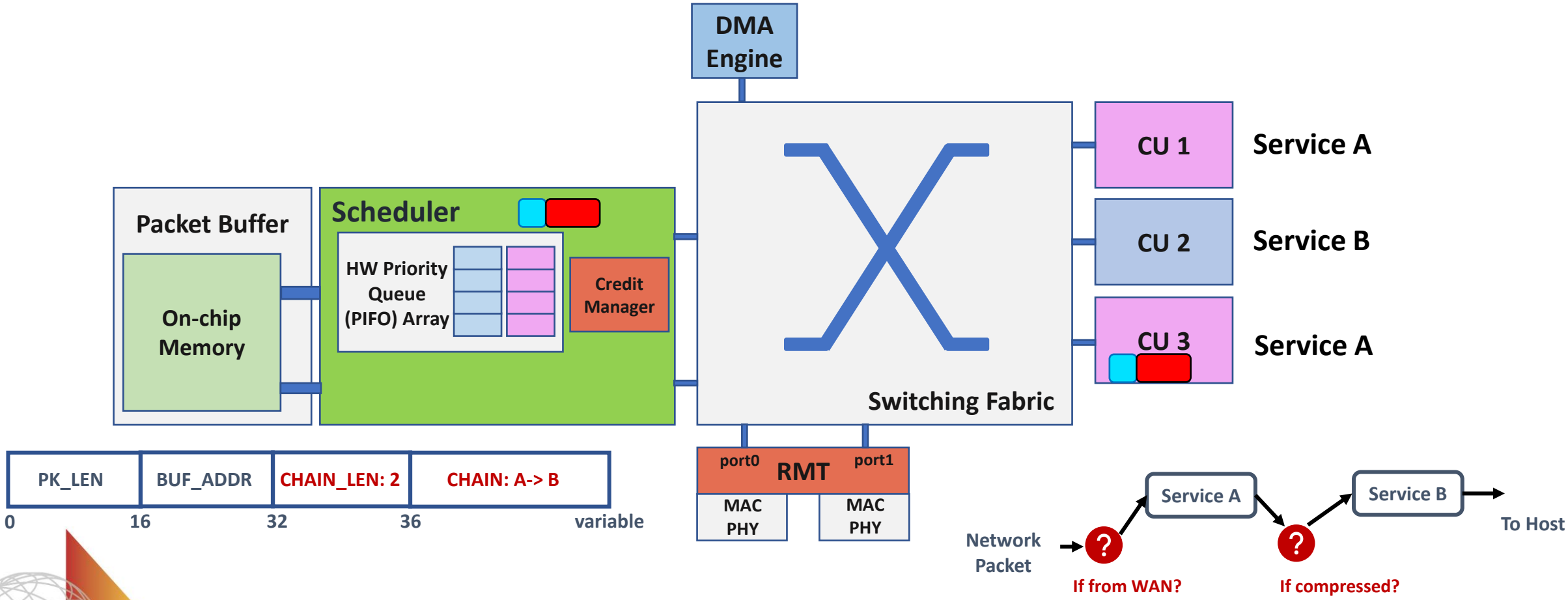
Life-Cycle of a Packet in PANIC



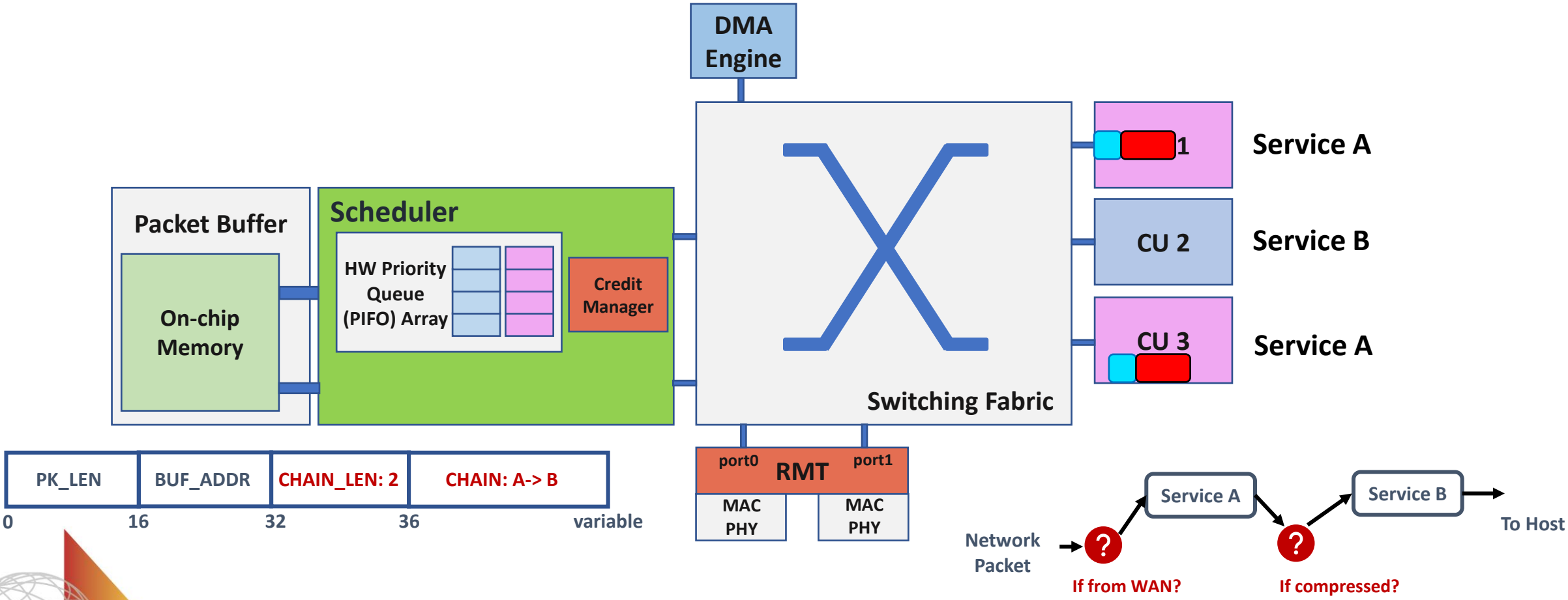
Life-Cycle of a Packet in PANIC



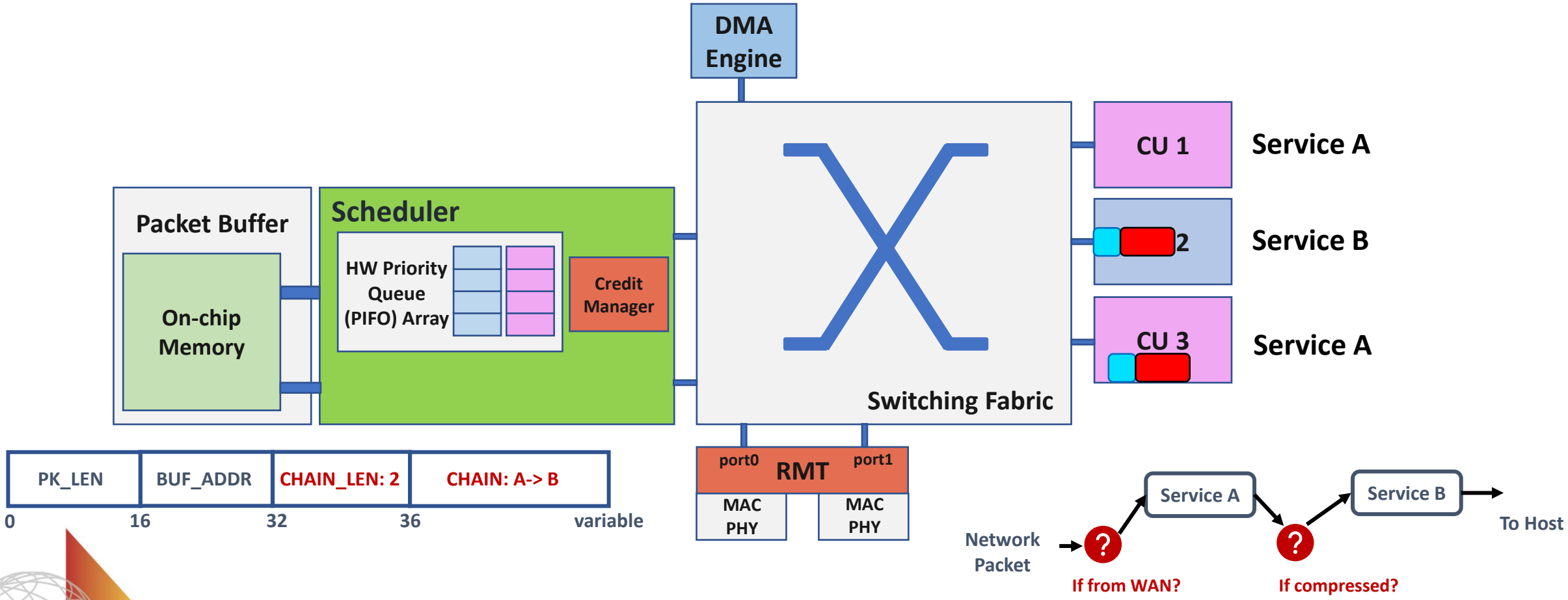
Life-Cycle of a Packet in PANIC



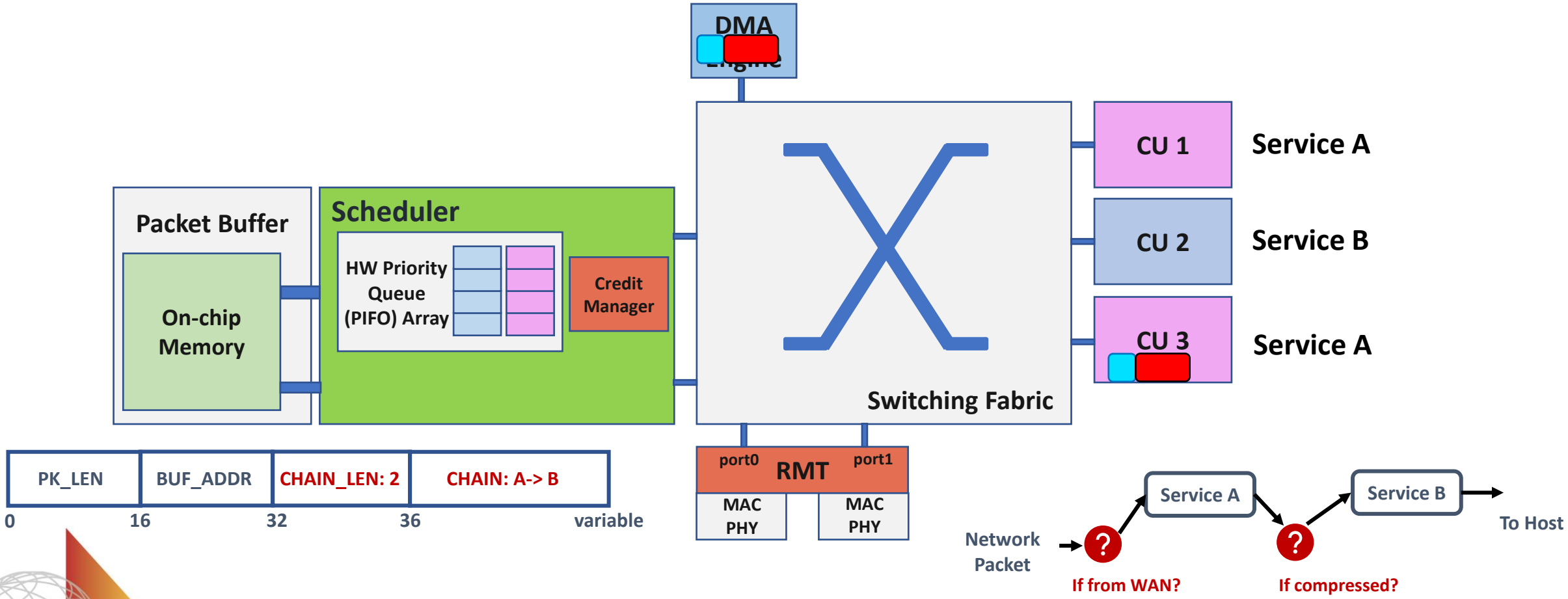
Life-Cycle of a Packet in PANIC



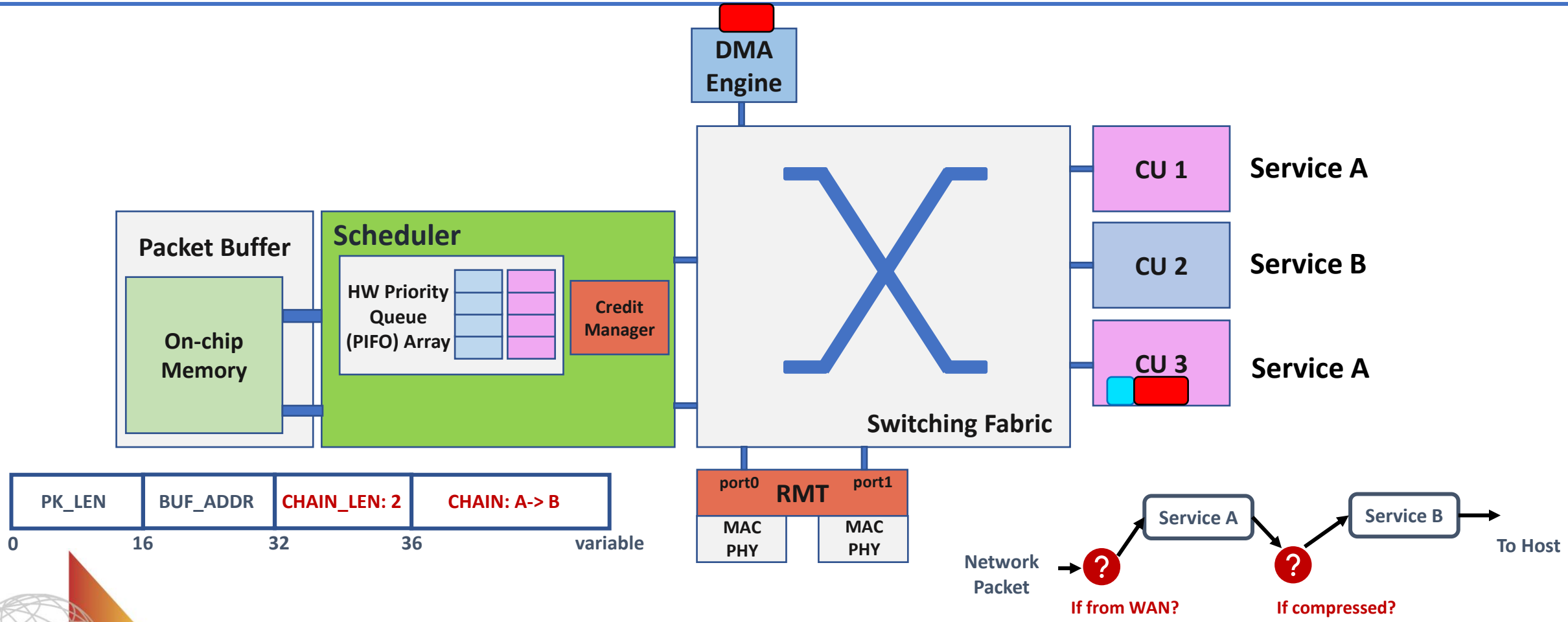
Life-Cycle of a Packet in PANIC



Life-Cycle of a Packet in PANIC



Life-Cycle of a Packet in PANIC



RMT Pipeline & Switching Fabric

RMT pipeline:

- Users program the match action tables in RMT.
  - **The service chain** for each tenant's traffic.
  - **The isolation policy and priority number** for each tenant's traffic.
- Action stage, RMT generates a PANIC descriptor for every packet

Switching Fabric:

- Providing offload chaining for an arbitrary chain.
- Each interconnect port should send and receive at full line-rate (> 100Gbps)

PANIC Scheduler:

Goal #1: Achieve high-performance chaining

Goal #2: Load-balance packets across parallel compute units in a service

Goal #3: Performance isolation across tenants

Goal #4: Buffer isolation across tenants

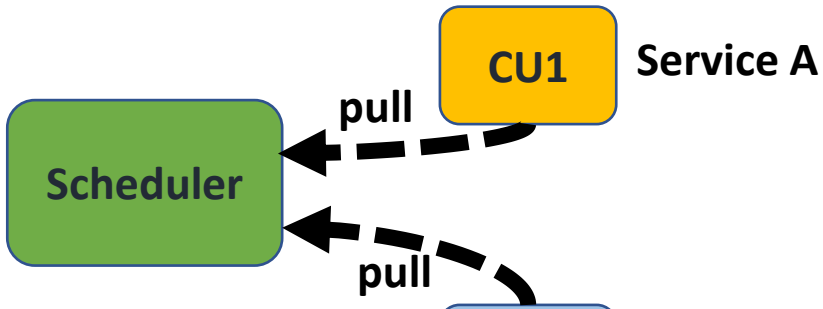
Problem: Chaining and Load Balancing

Goal #1: Achieve high-performance chaining

Goal #2: Load-balance packets across parallel compute units in a service.

- Pull-based scheduling

✓ **Achieve Load balancing!**



Every hop goes back to scheduler; cannot achieve high performance chaining!

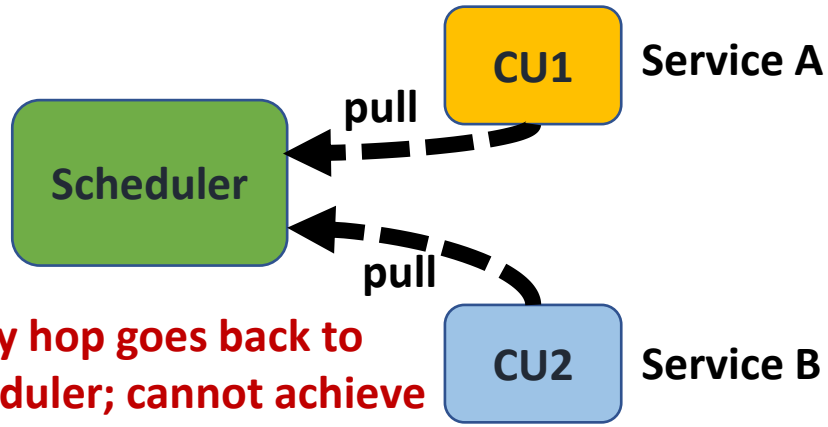
Problem: Chaining and Load Balancing

Goal #1: Achieve high-performance chaining

Goal #2: Load-balance packets across parallel compute units in a service.

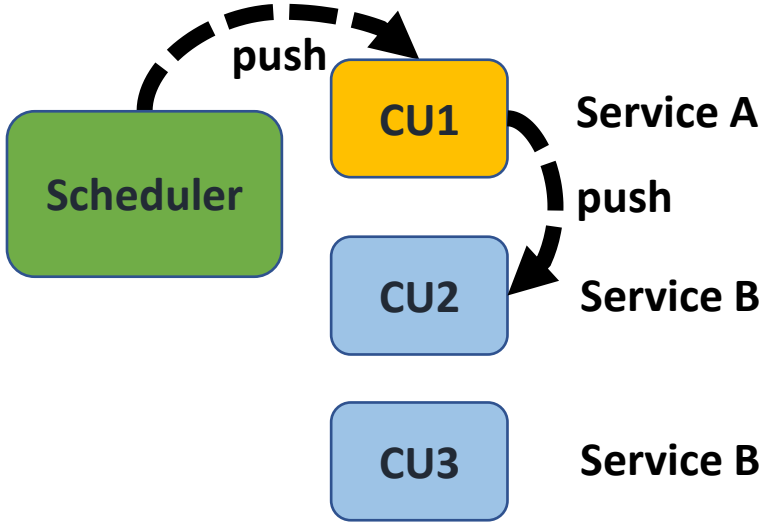
- Pull-based scheduling

✓ **Achieve Load balancing!**



Every hop goes back to scheduler; cannot achieve high performance chaining!

- Push-based scheduling



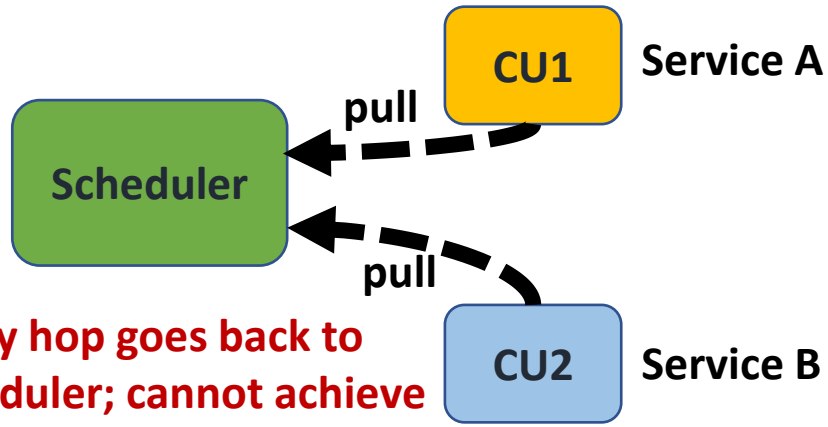
Problem: Chaining and Load Balancing

Goal #1: Achieve high-performance chaining

Goal #2: Load-balance packets across parallel compute units in a service.

- Pull-based scheduling

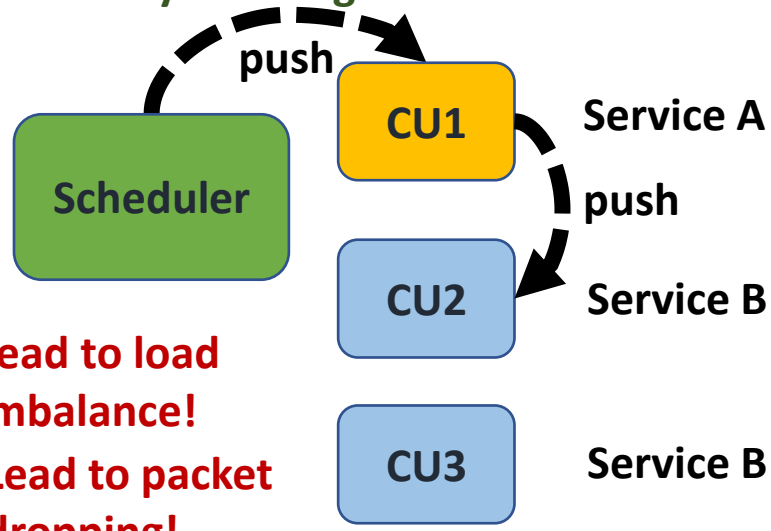
✓ **Achieve Load balancing!**



Every hop goes back to scheduler; cannot achieve high performance chaining!

- Push-based scheduling

✓ **Low latency chaining!**



Lead to load imbalance!

Lead to packet dropping!

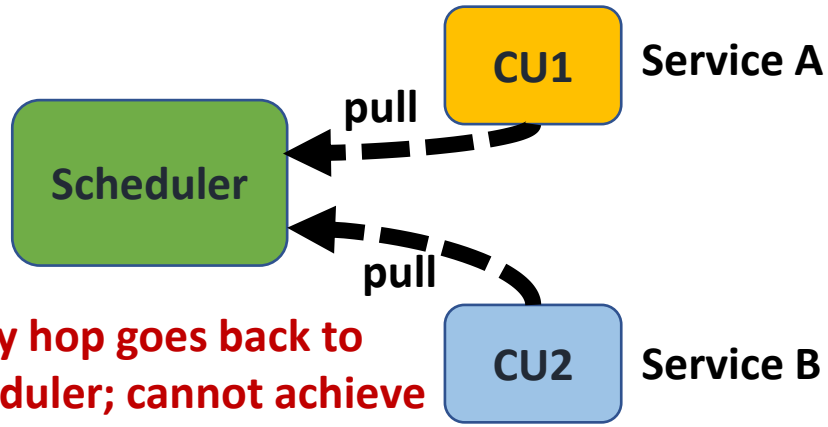
Problem: Chaining and Load Balancing

Goal #1: Achieve high-performance chaining

Goal #2: Load-balance packets across parallel compute units in a service.

- Pull-based scheduling

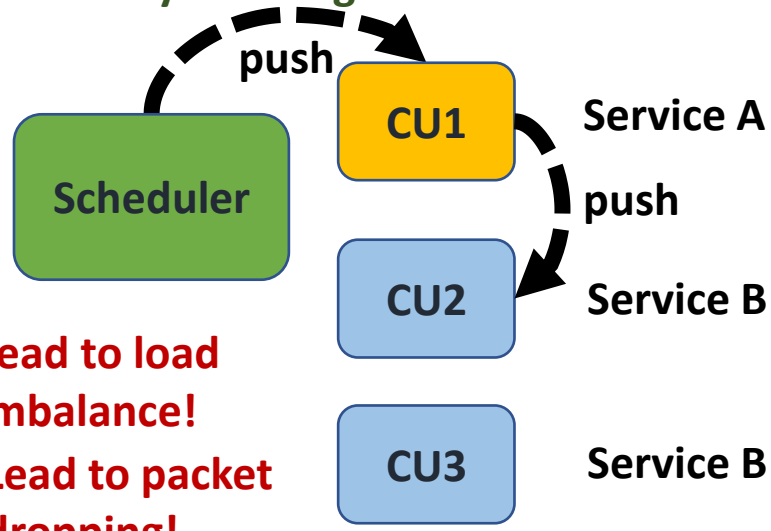
✓ **Achieve Load balancing!**



Every hop goes back to scheduler; cannot achieve high performance chaining!

- Push-based scheduling

✓ **Low latency chaining!**



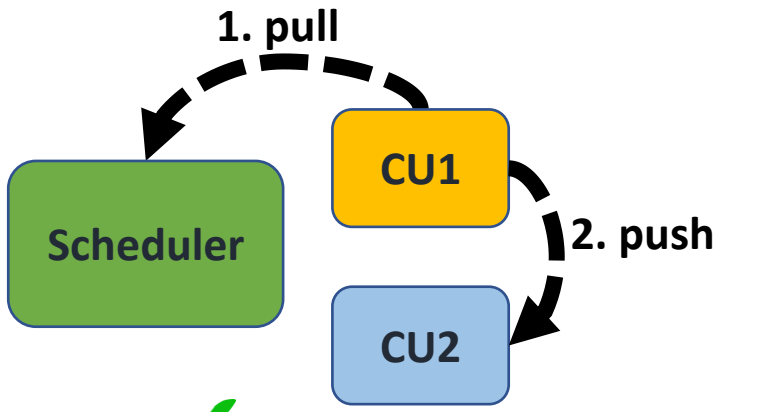
Lead to load imbalance!

Lead to packet dropping!

Solution: Hybrid push-pull scheduling

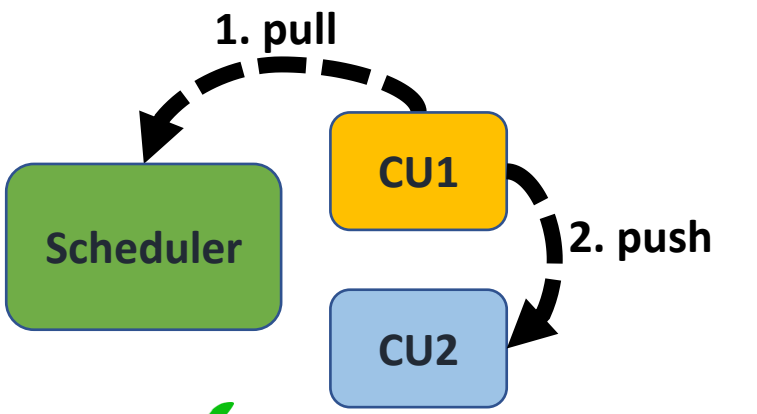
PANIC Scheduler: Hybrid Push Pull Scheduling

- **Hybrid Push Pull scheduling:**
  - CU can either **pull** packet from the scheduler or **accept the pushed** packet from other units.
  - According to CUs' load, switches between push pull scheduling.
    - **During Low Load:** the packet is pushed to all the units in a chain.

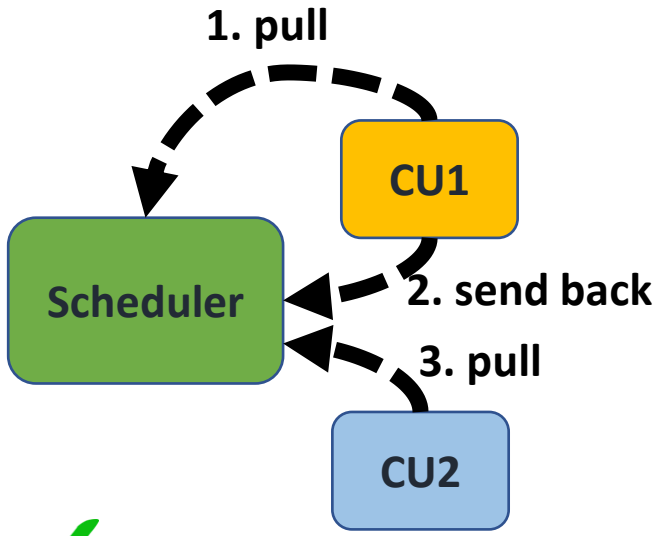


PANIC Scheduler: Hybrid Push Pull Scheduling

- **Hybrid Push Pull scheduling:**
  - CU can either **pull** packet from the scheduler or **accept the pushed** packet from other units.
  - According to CUs' load, switches between push pull scheduling.
    - **During Low Load:** the packet is pushed to all the units in a chain.
    - **During High Load:** the packet is sent back to the scheduler, until it can be pulled by an idle CU.



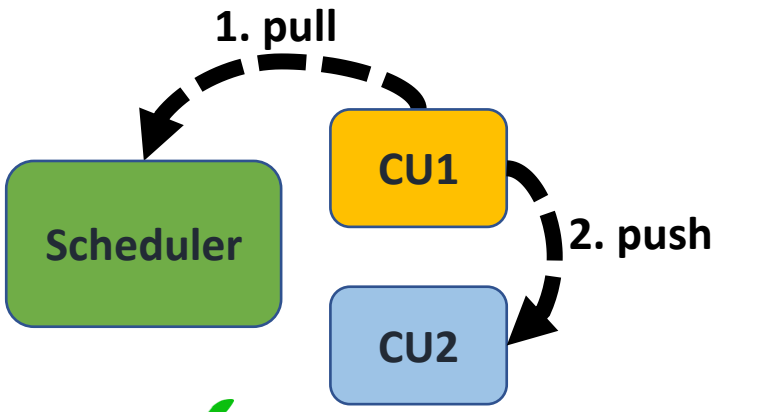
✓ Achieve High Performance Chaining!



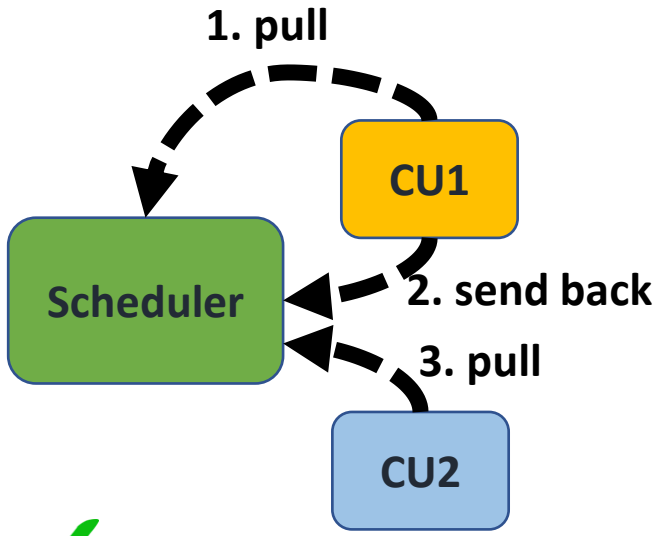
✓ Achieve Load balancing!

PANIC Scheduler: Hybrid Push Pull Scheduling

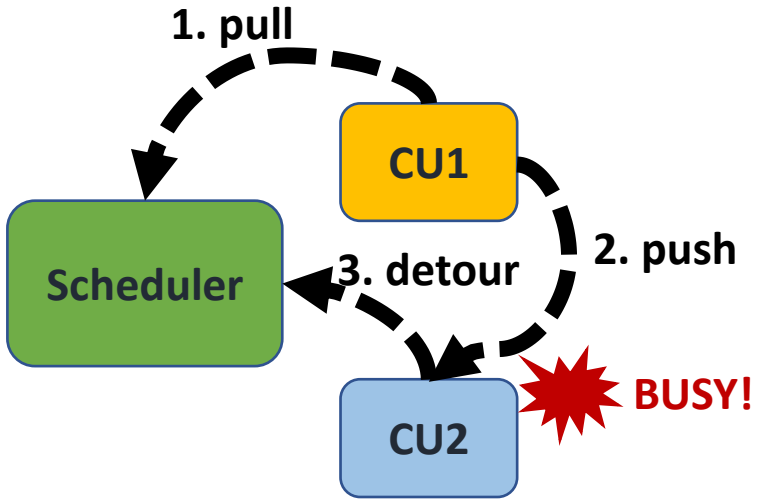
- **Hybrid Push Pull scheduling:**
  - CU can either **pull** packet from the scheduler or **accept the pushed** packet from other units.
  - According to CUs' load, switches between push pull scheduling.
    - **During Low Load:** the packet is pushed to all the units in a chain.
    - **During High Load:** the packet is sent back to the scheduler, until it can be pulled by an idle CU.
    - **Detour Routing:** In push scheduling, if the downstream is busy due to a burst.



✓ Achieve High Performance Chaining!



✓ Achieve Load balancing!

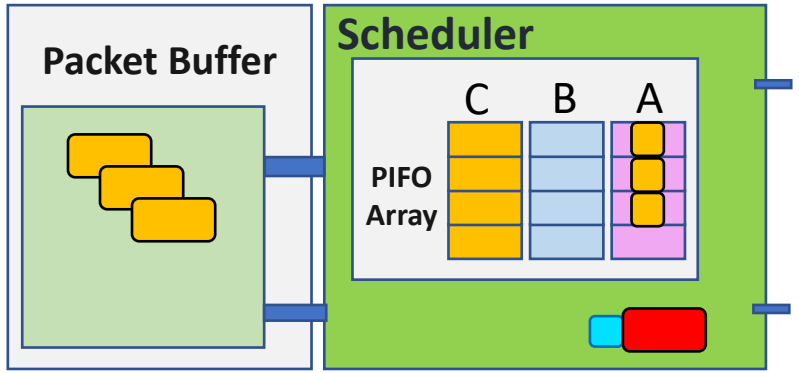


✓ Avoid Packet Dropping

PANIC Scheduler: Performance Isolation

Goal #3: Priority scheduling and performance isolation

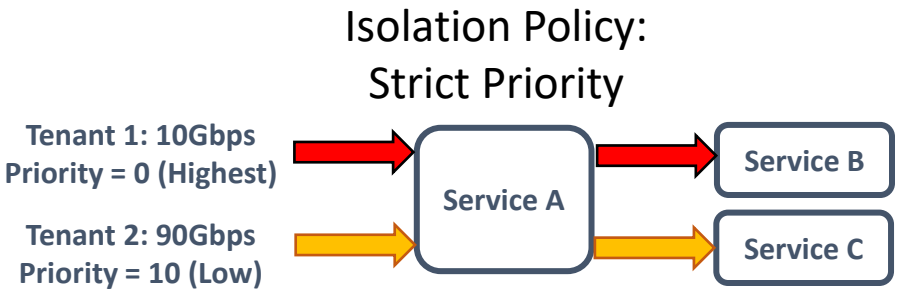
- **PIFO array for performance Isolation:**
  - PIFO (*PUSH IN, FIRST OUT Queue*) runs like the hardware priority queue. One service has one logic PIFO queue.

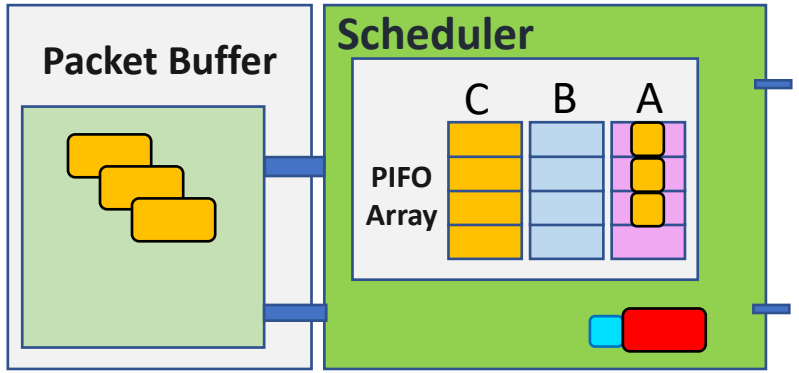


PANIC Scheduler: Performance Isolation

Goal #3: Priority scheduling and performance isolation

- **PIFO array for performance Isolation:**
  - PIFO (*PUSH IN, FIRST OUT Queue*) runs like the hardware priority queue. One service has one logic PIFO queue.
  - Packet descriptors is sorted according to the packet rank in per-service PIFO.

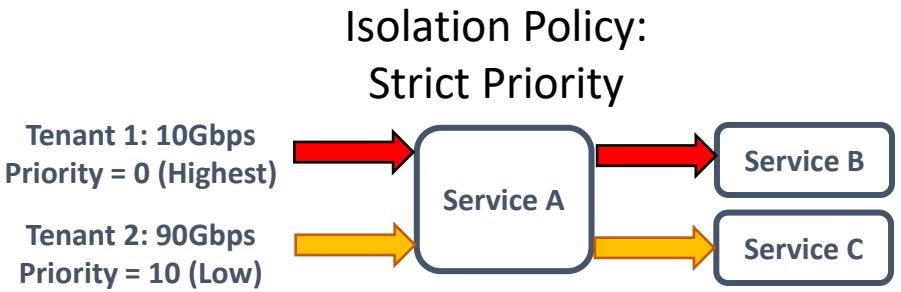




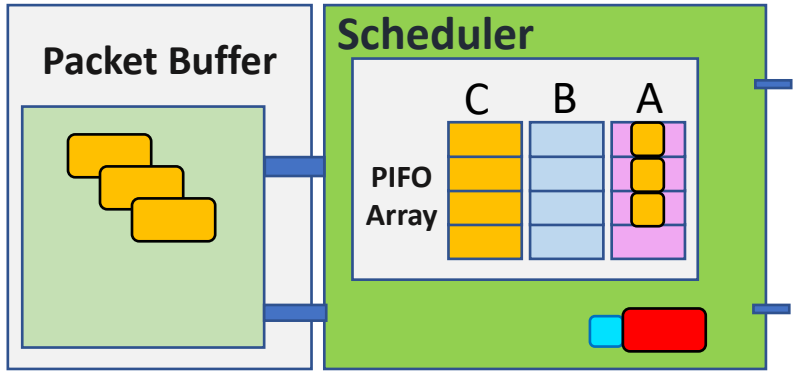
PANIC Scheduler: Performance Isolation

Goal #3: Priority scheduling and performance isolation

- **PIFO array for performance Isolation:**
  - PIFO (*PUSH IN, FIRST OUT Queue*) runs like the hardware priority queue. One service has one logic PIFO queue.
  - Packet descriptors is sorted according to the packet rank in per-service PIFO.



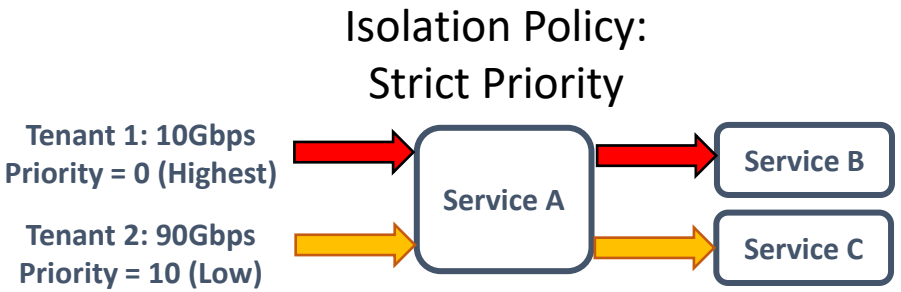
PK_LEN	BUF_ADDR	CHAIN_LEN: 2	CHAIN: A-> B	SCHE_METADATA: <Strict, High>



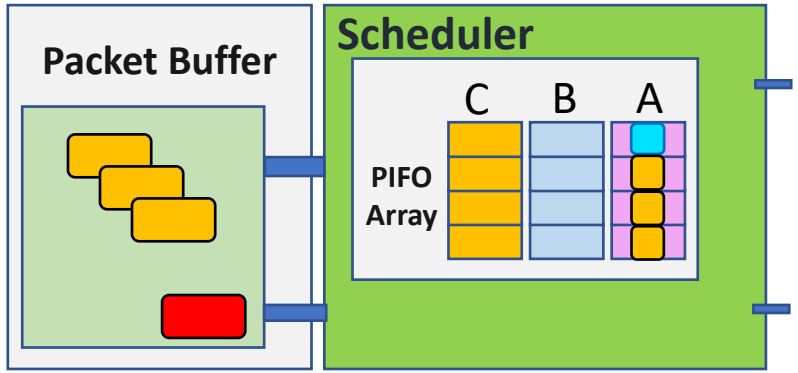
PANIC Scheduler: Performance Isolation

Goal #3: Priority scheduling and performance isolation

- **PIFO array for performance Isolation:**
  - PIFO (*PUSH IN, FIRST OUT Queue*) runs like the hardware priority queue. One service has one logic PIFO queue.
  - Packet descriptors is sorted according to the packet rank in per-service PIFO.



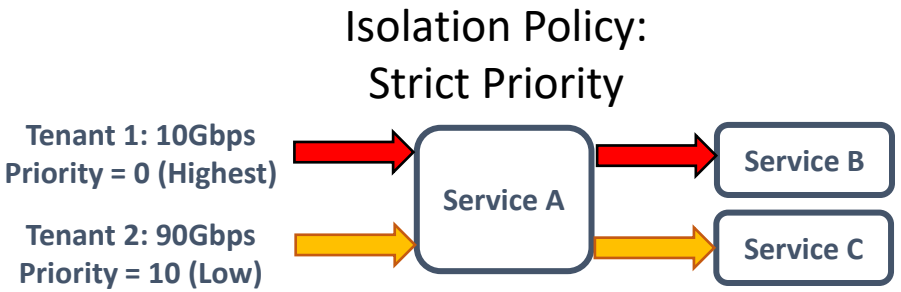
PK_LEN	BUF_ADDR	CHAIN_LEN: 2	CHAIN: A-> B	SCHE_METADATA: <Strict, High>



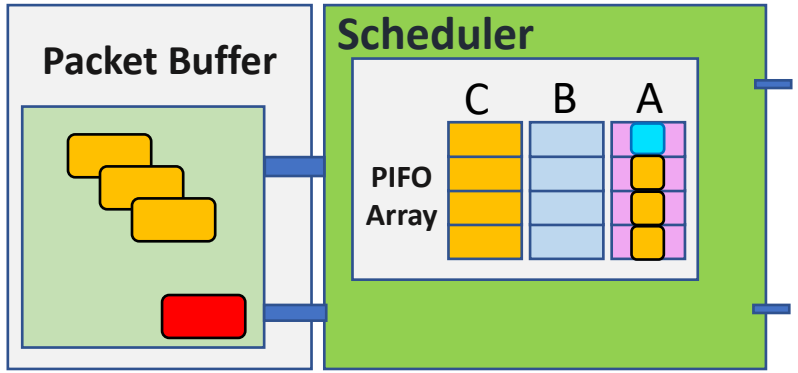
PANIC Scheduler: Performance Isolation

Goal #3: Priority scheduling and performance isolation

- **PIFO array for performance Isolation:**
  - PIFO (*PUSH IN, FIRST OUT Queue*) runs like the hardware priority queue. One service has one logic PIFO queue.
  - Packet descriptors is sorted according to the packet rank in per-service PIFO.
  - Support different isolation policy
    - (WFQ, LSTF, Rate Limiting…)



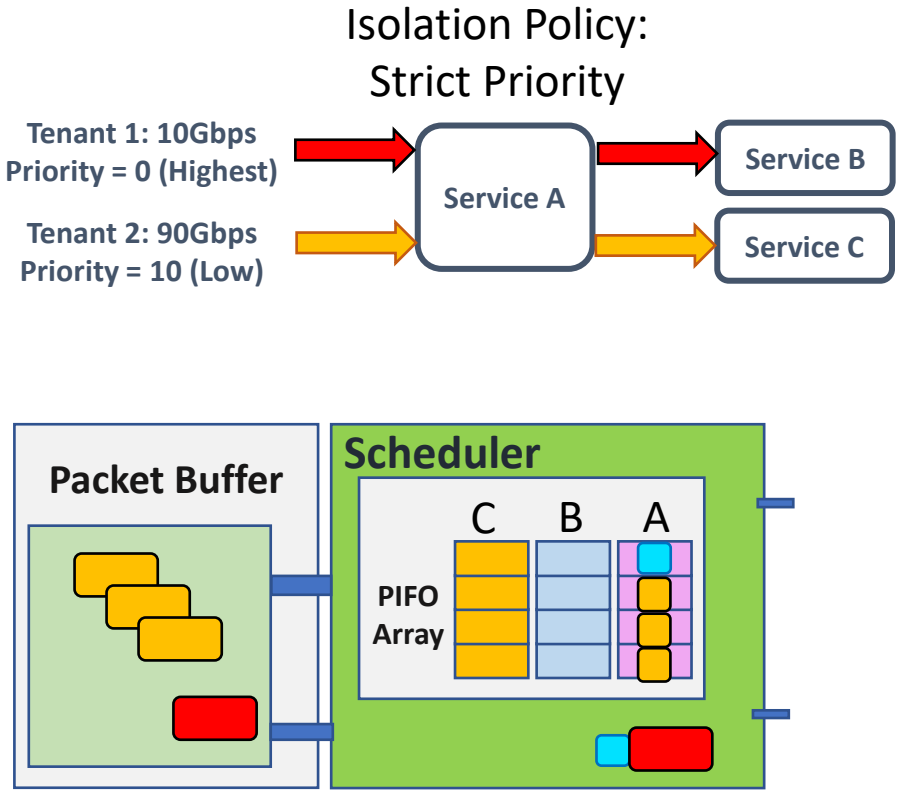
PK_LEN	BUF_ADDR	CHAIN_LEN: 2	CHAIN: A-> B	SCHE_METADATA: <Strict, High>



PANIC Scheduler: Prioritized Dropping

Goal #4: Ensure buffer isolation across tenants

- **Naïve dropping method:** drop the newest income packet when the buffer is full:
  No Isolation!

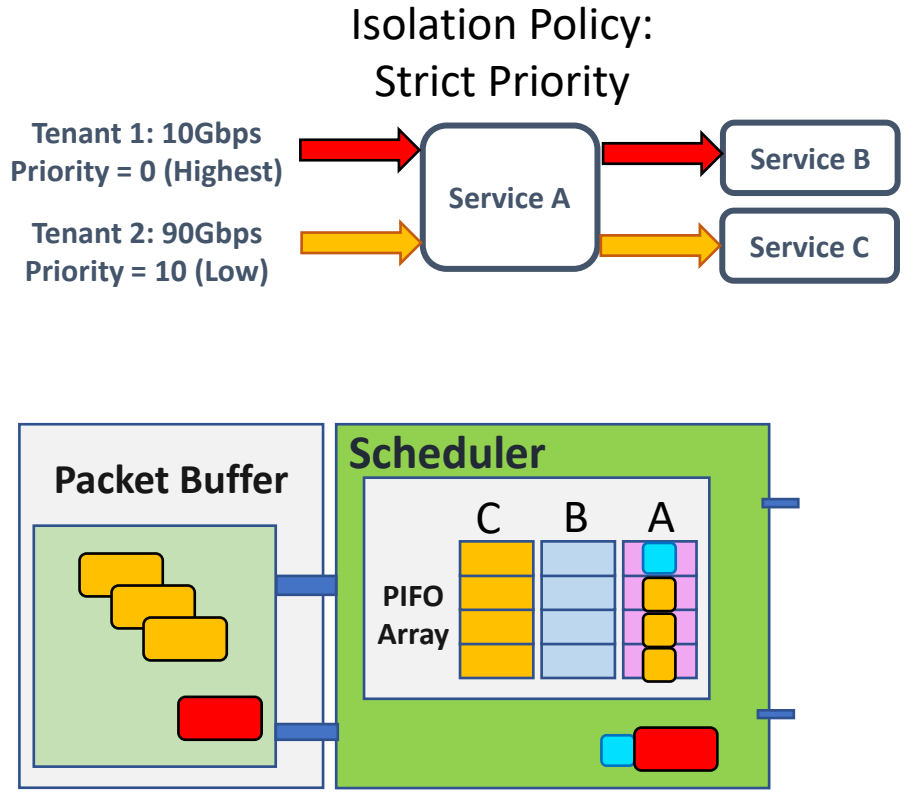


PANIC Scheduler: Prioritized Dropping

Goal #4: Ensure buffer isolation across tenants

- **Naïve dropping method:** drop the newest income packet when the buffer is full:
  No Isolation!

- **Prioritized Dropping:** drop the lowest rank packet when the buffer is almost full.
  - Extend PIFO's interface to allow it to support *PUSH IN, FIRST OUT, **REMOVE LAST***
  - **Isolation!** PANIC can ensure the high priority packet enters buffer and receive service.

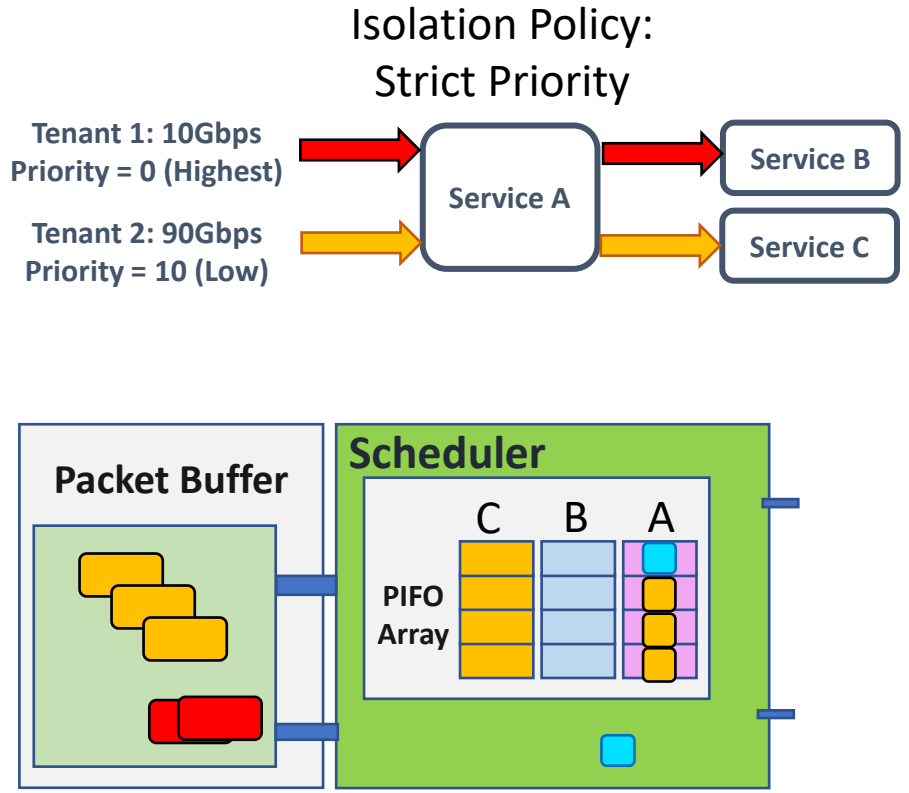


PANIC Scheduler: Prioritized Dropping

Goal #4: Ensure buffer isolation across tenants

- **Naïve dropping method:** drop the newest income packet when the buffer is full:
  No Isolation!

- **Prioritized Dropping:** drop the lowest rank packet when the buffer is almost full.
  - Extend PIFO's interface to allow it to support *PUSH IN, FIRST OUT, **REMOVE LAST***
  - **Isolation!** PANIC can ensure the high priority packet enters buffer and receive service.

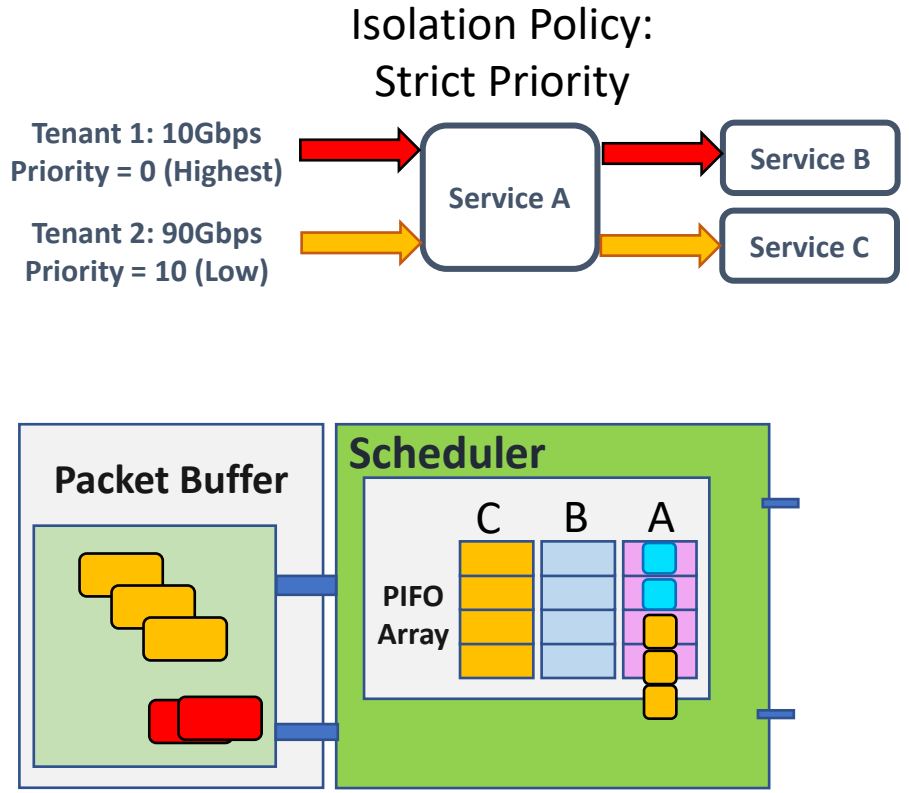


PANIC Scheduler: Prioritized Dropping

Goal #4: Ensure buffer isolation across tenants

- **Naïve dropping method:** drop the newest income packet when the buffer is full:
  No Isolation!

- **Prioritized Dropping:** drop the lowest rank packet when the buffer is almost full.
  - Extend PIFO's interface to allow it to support *PUSH IN, FIRST OUT, **REMOVE LAST***
  - **Isolation!** PANIC can ensure the high priority packet enters buffer and receive service.

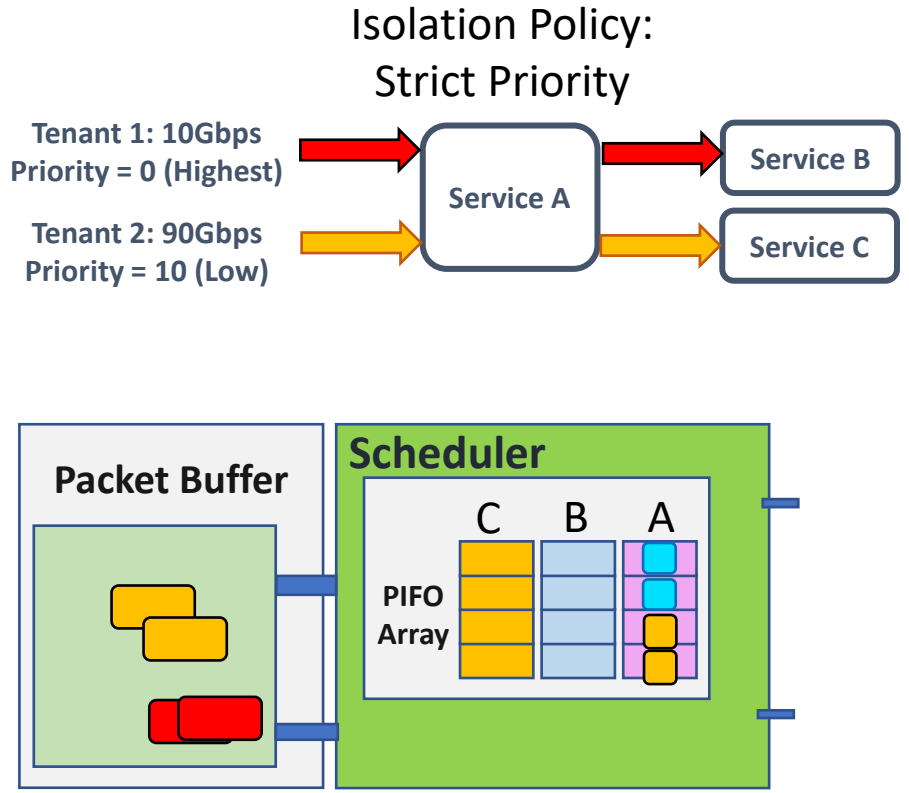


PANIC Scheduler: Prioritized Dropping

Goal #4: Ensure buffer isolation across tenants

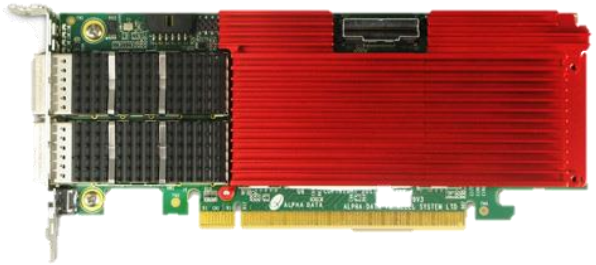
- **Naïve dropping method:** drop the newest income packet when the buffer is full:
  No Isolation!

- **Prioritized Dropping:** drop the lowest rank packet when the buffer is almost full.
  - Extend PIFO's interface to allow it to support *PUSH IN, FIRST OUT, **REMOVE LAST***
  - **Isolation!** PANIC can ensure the high priority packet enters buffer and receive service.



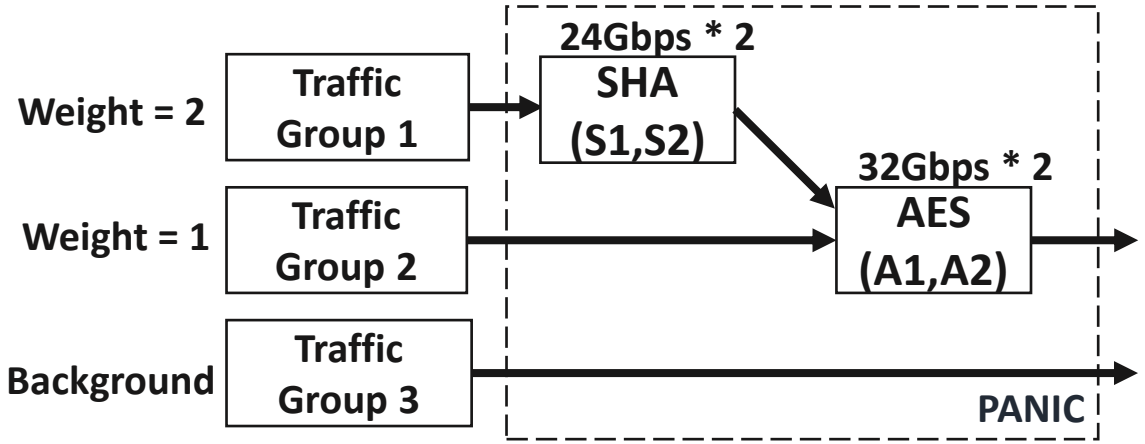
PANIC Implementation

- 100G FPGA prototype in ADM-PCIE-9V3 accelerator
- ~6K lines of Verilog code
- Prototype Components:
  - A lightweight RMT pipeline
  - 8 * 8 full connected crossbar (512 bit width @ 250MHz)
  - Dual-port central scheduler (512 bit width @ 250MHz)
    - PIFO block @ 125MHz
  - Compute Units
    - **AES-256-CTR** encryption unit (24Gbps @ 250Mhz)
    - **SHA-3-512** hash unit (32Gbps @ 150Mhz)
    - **An RISC-V core** unit (5-stage pipeline @ 250MHz)



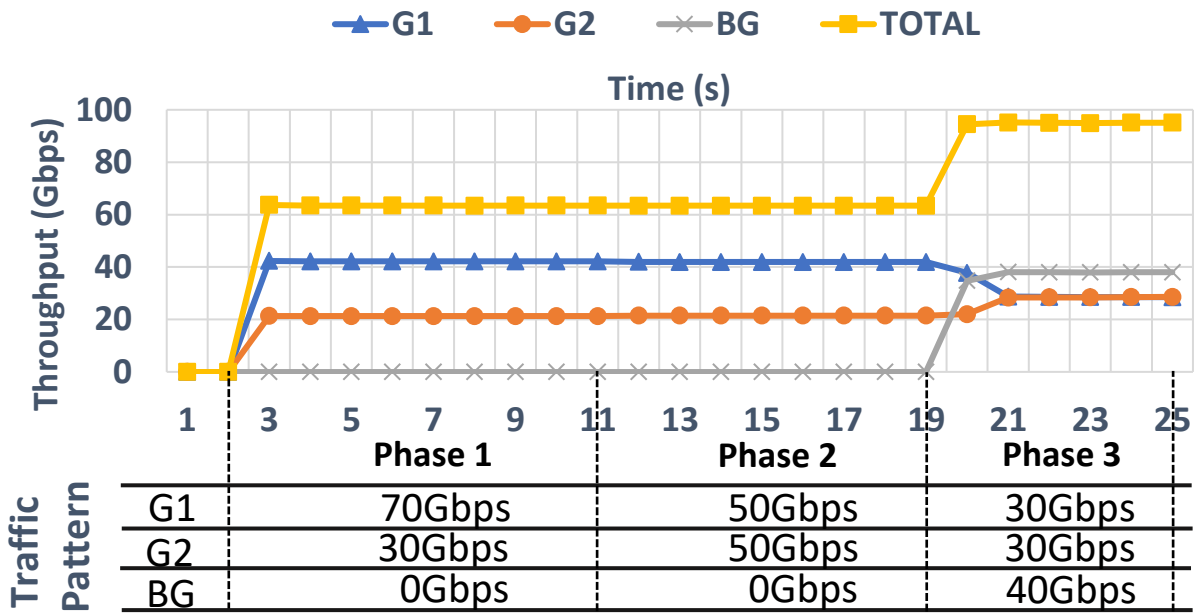
https://bitbucket.org/uw-madison-networking-research/panic_osdi20_artifact/

PANIC Evaluation



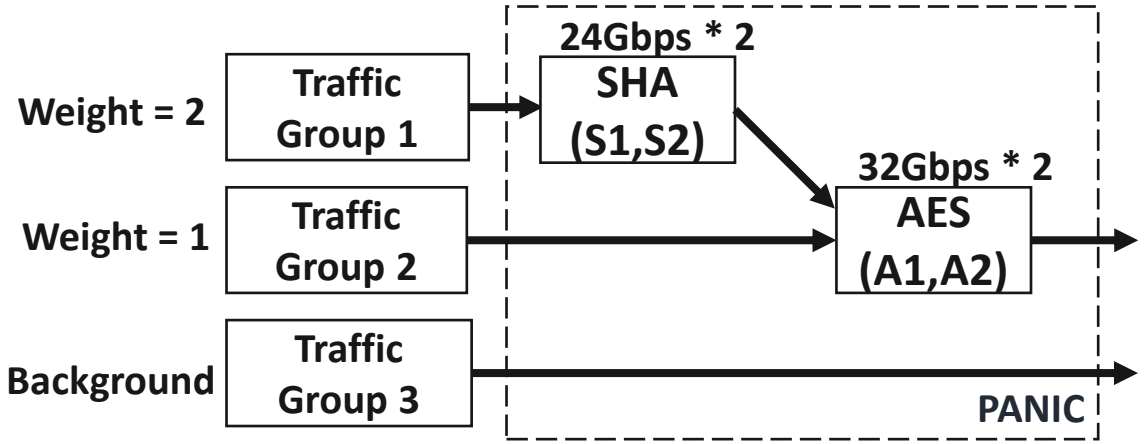
Setup:

- The sender server generate network traffic using different traffic pattern.
- 2 SHA engines and 2 AES engines are attached.
- Isolation Policy: weighted fair queuing.



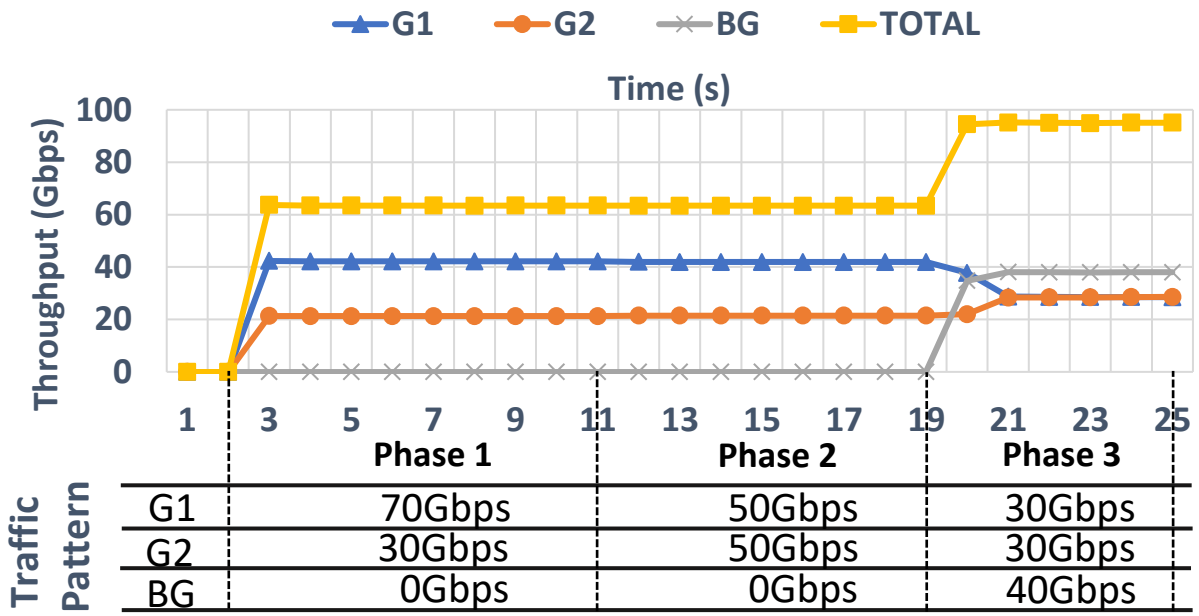
Traffic Pattern	Phase 1	Phase 2	Phase 3
G1	70Gbps	50Gbps	30Gbps
G2	30Gbps	50Gbps	30Gbps
BG	0Gbps	0Gbps	40Gbps

PANIC Evaluation



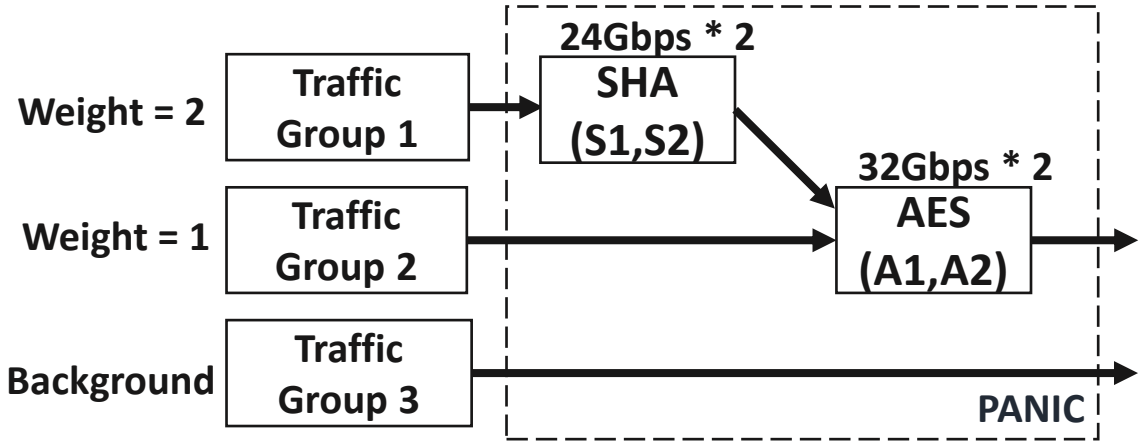
Setup:

- The sender server generate network traffic using different traffic pattern.
- 2 SHA engines and 2 AES engines are attached.
- Isolation Policy: weighted fair queuing.



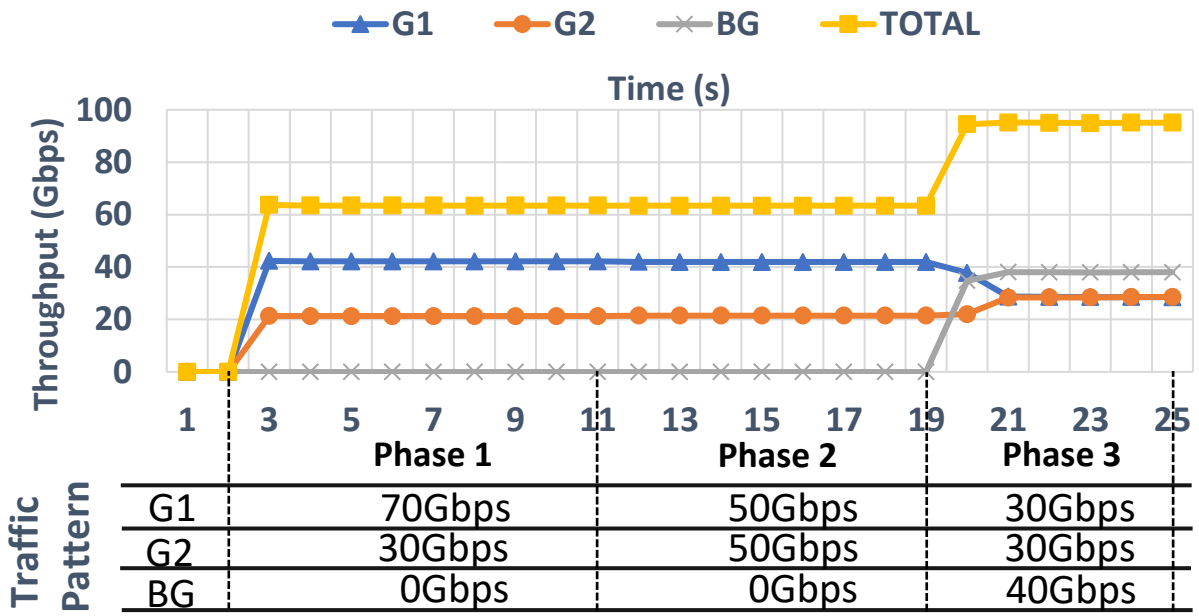
Traffic Pattern	Phase 1	Phase 2	Phase 3
G1	70Gbps	50Gbps	30Gbps
G2	30Gbps	50Gbps	30Gbps
BG	0Gbps	0Gbps	40Gbps

PANIC Evaluation



Setup:

- The sender server generate network traffic using different traffic pattern.
- 2 SHA engines and 2 AES engines are attached.
- Isolation Policy: weighted fair queuing.



Traffic Pattern	Phase 1	Phase 2	Phase 3
G1	70Gbps	50Gbps	30Gbps
G2	30Gbps	50Gbps	30Gbps
BG	0Gbps	0Gbps	40Gbps

Conclusion

- PANIC is a full line-rate programmable NIC design that overcomes current NICs limitation in multi-tenant environments.

Thank you for listening!

Questions?