Chapter 12: Flow, Flowlet, and Packet-Based Load Balancing

Introduction

Though BGP supports the traditional Flow-based Layer 3 Equal Cost Multi-Pathing (ECMP) traffic load balancing method, it is not the best fit for a RoCEv2-based AI backend network. This is because GPU-to-GPU communication creates massive elephant flows, which RDMA-capable NICs transmit at line rate. These flows can easily cause congestion in the backend network.

In ECMP, all packets of a single flow follow the same path. If that path becomes congested, ECMP does not adapt or reroute traffic. This leads to uneven bandwidth usage across the network. Some links become overloaded, while others remain idle. In AI workloads, where multiple high-bandwidth flows occur at the same time, this imbalance can degrade performance.

Deep learning models rely heavily on collective operations like all-reduce, all-gather, and broadcast. These generate dense traffic patterns between GPUs, often at terabit-per-second speeds. If these flows are not evenly distributed, a single congested path can slow down the entire training job.

This chapter introduces two alternative load balancing methods to traditional Flow-Based with Layer 3 ECMP: 1) Flowlet-Based Load Balancing with Adaptive Routing, and 2) Packet-Based Load Balancing with Packet Spraying. Both aim to improve traffic distribution in RoCEv2-based AI backend networks, where conventional flow-based routing often leads to congestion and underutilized links. These advanced methods are designed to handle the unique traffic patterns of AI workloads more efficiently.

RDMA WRITE Operation

Before we explore the load balancing solution, let's first walk through a simplified example of how the RDMA WRITE memory copy operation works. In Figure 12-1, we have two GPU servers: Host 1 and Host 2, each with one GPU. By this point, the memory has already been allocated and registered, and the Queue Pair (QP) has been created on both sides, so the data transfer can begin.

On GPU-0 of Host 1, gradients are stored in memory regions highlighted in green, orange, and blue. Each colored section represents a portion of local memory that will be written to GPU-0 on Host 2. To transfer the data, the RDMA NIC on Host 1 splits the write operation into three flowlets (green, orange, and blue). Rather than sending the entire data block as a single continuous stream, each flowlet is treated as a segment of the same RDMA Write operation.

RDMA Write First

The first message carries the RDMA Extended Transport Header (RETH) in its payload. This header tells the receiving RDMA NIC where in the remote memory the incoming data should be written. In our example, data from memory block 1B of GPU-0 on Host 1 is written to memory block 2C of GPU-0 on Host 2.

The RETH contains the R_Key, which gives permission to write to the remote memory region. It also includes the length of the data being transferred and the virtual address of the target memory location on Host 2.

The operation code in the InfiniBand Base Transport Header (IBTH) is set to RDMA Write First, indicating that this is the first message in the sequence. The IBTH also describes the Partition Key (interface identifier), the Destination Queue Pair number, and the Packet Sequence Number (PSN) that helps ensure packets are processed in the correct order.

When this first packet arrives at Host 2, the RDMA NIC uses the Virtual Address information in the RETH header to write the payload directly into memory block 2C.

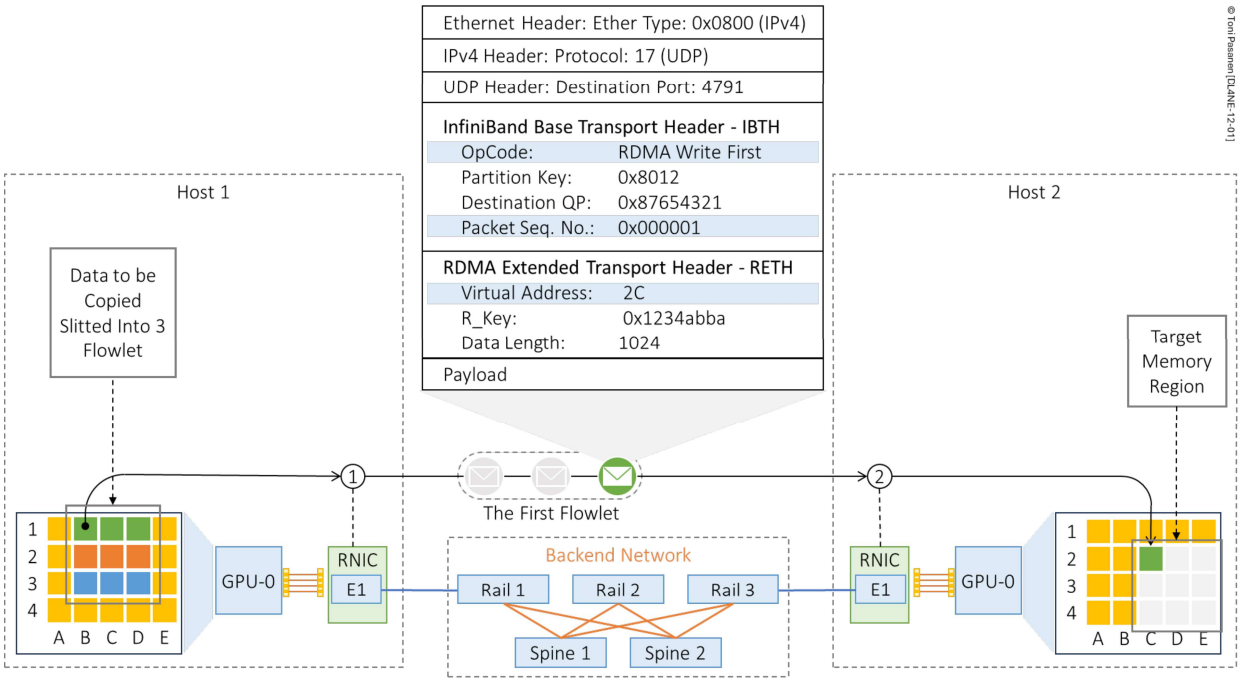

Figure 12-1: *RDMA WRITE First.*

RDMA Write Middle

The second message has the opcode RDMA Write Middle and PSN 2, which tells the receiver that this packet comes after the first one with PSN 1. The payload of this Flowlet is written right after the previous block, into memory block 2D on Host 2. The RDMA NIC ensures that the order is maintained based on PSNs, and it knows exactly where to place the data thanks to the original offset from the first packet.

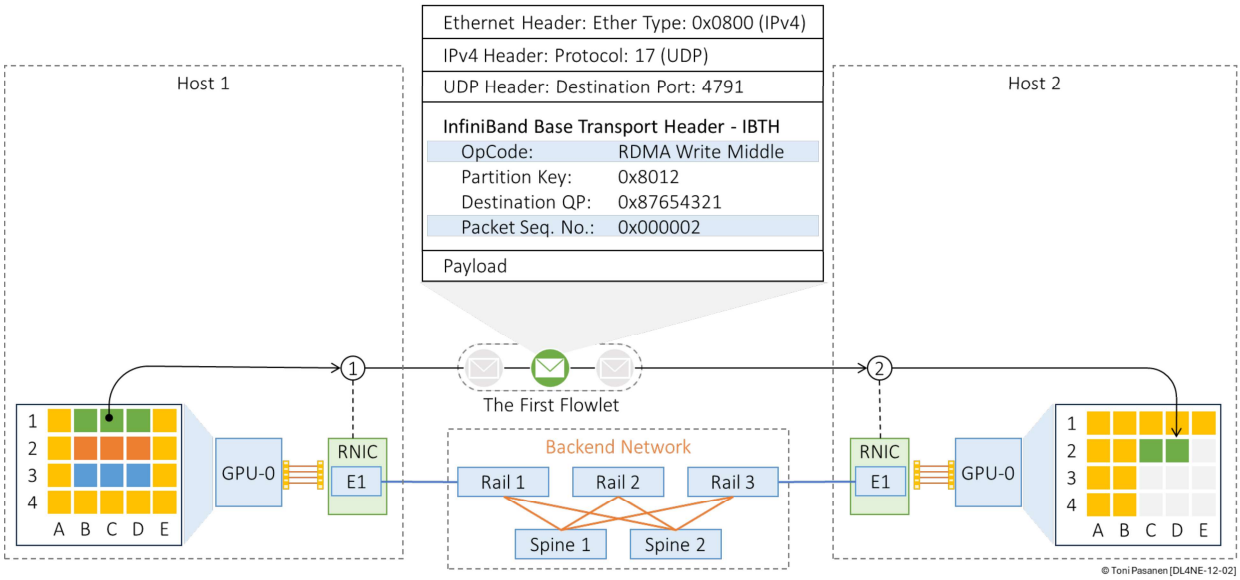

Figure 12-2: *RDMA WRITE Middle.*

RDMA Write Last

The third message has the opcode RDMA Write Last, indicating that this is the final message in the sequence. With PSN 3, it follows directly after PSN 2. The payload in this packet is written into memory block 2E, which comes directly after 2D.

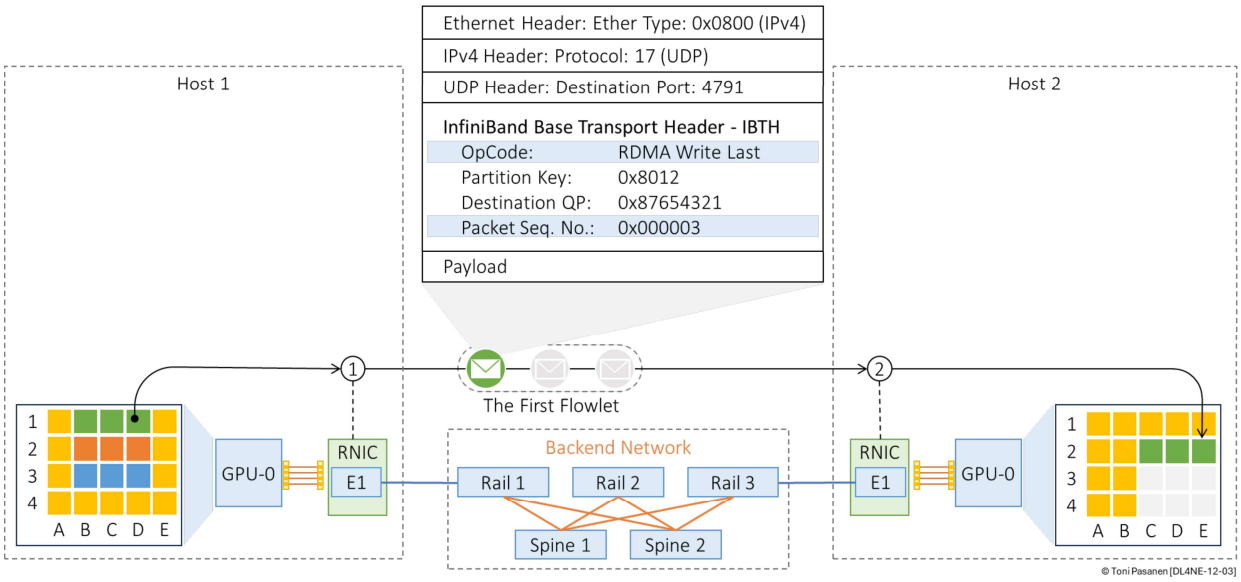

Figure 12-3: *RDMA WRITE Last.*

In a multi-packet RDMA Write operation, each Flowlet represents a continuous block of data being transferred from the source GPU to the destination GPU. Data within packets must arrive in the correct order because only the first packet includes the full addressing information in the RDMA Extended Transport Header (RETH). This header tells the receiver where in memory the data should be written.

Packets marked as RDMA Write Middle and RDMA Write Last depend on this information and must follow the sequence defined by the Packet Sequence Numbers (PSNs). If packets are delivered out of order, the receiving RDMA NIC cannot process them immediately. Instead, it must hold them in memory and wait for the missing earlier packets to arrive. This buffering increases memory usage and processing overhead. In high-

speed environments, this can lead to performance degradation or even packet drops, especially when buffers fill up under heavy load.

Flow-Based Load Balancing with Layer 3 ECMP

Figure 12-4 depicts the problem with flow-based load balancing when used in an AI fabric backend network. In our example, we have four hosts, each equipped with two GPUs: GPU-1 and GPU-2. The RDMA NICs connected to GPU-1s are linked to switch Rail-1, and the RDMA NICs connected to GPU-2s are linked to Rail-2. Traffic between NICs on Rail-1 and Rail-2 is forwarded through either Spine-1 or Spine-2.

We use a basic data parallelization strategy, where the training dataset is divided into mini-batches and distributed across all eight GPUs. To keep the example simple, Figure 12-4 only shows the all-reduce gradient synchronization flow from the GPU-1s on Hosts 1, 2, and 3 to the GPU-2 on Host 4. In real-world training, a full-mesh all-reduce operation takes place between all GPUs.

As a starting point, the GPU-1s on the three leftmost hosts begin the RDMA process to copy data from their memory to the memory of GPU-2 on Host 4. These GPU-1s are all connected to Rail-1. Instead of sending one large flow, the RDMA NICs split the data into flowlets, small bursts of data from the larger transfer. These flowlets arrive at the Rail-1 switch, where the 5-tuple L3 ECMP hash algorithm unfortunately selects the same uplink for all three flows.

Since the switch cannot forward all the data at wire speed, it stores some of the packets in the buffer, causing congestion. Similar congestion may also occur at the spine switches. As explained earlier in Chapter 12, egress buffer overflow may trigger ECN (Explicit Congestion Notification) and PFC (Priority Flow Control) mechanisms to prevent packet loss.

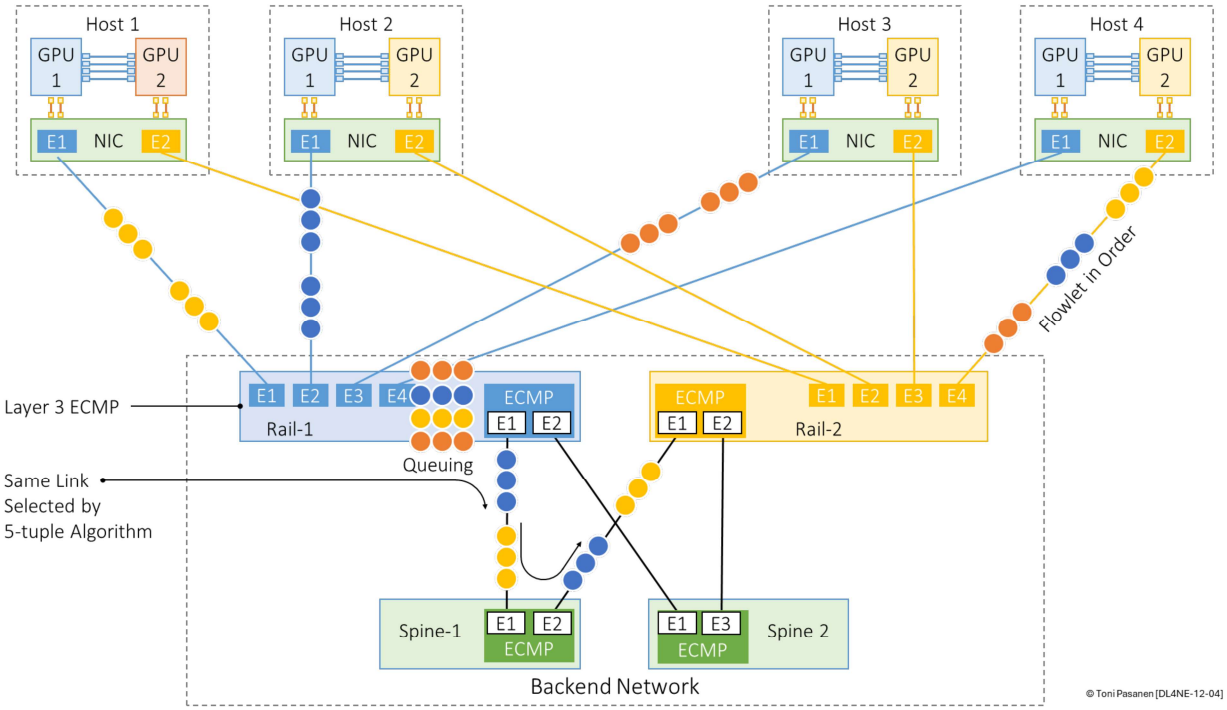

Figure 12-4: *Layer 3 Load balancing.*

Flowlet-Based Load Balancing with Adaptive Routing

Adaptive routing is a dynamic method that actively monitors link utilization and reacts to network congestion in real time. In Figure 12-5, the 5-tuple hash algorithm initially selects the same uplink for all flowlets, just like in the previous example. However, once the utilization of the inter-switch link between Rail-1 and Spine-1 goes over threshold, the adaptive routing mechanism detects the increased load and starts redirecting some of the flowlets to an alternate, less congested path, through Spine-2.

By distributing the flowlets across multiple paths, adaptive routing helps to reduce buffer buildup and avoid potential packet drops. This not only improves link utilization across the fabric but also helps maintain consistent throughput for time-sensitive operations like RDMA-based gradient synchronization. In AI workloads, where delays or packet loss can slow down or even interrupt training, adaptive routing plays a critical role in maintaining system performance.

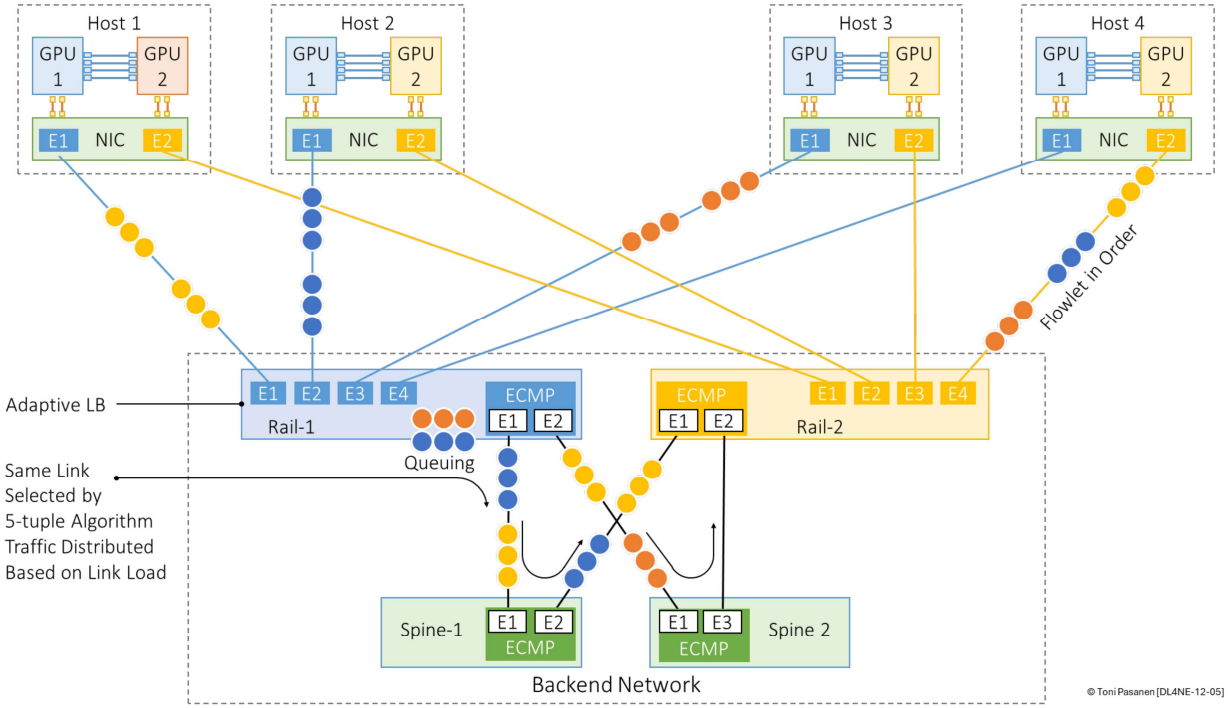

Figure 12-5: *Dynamic Flow Balancing.*

Packet-Based Load Balancing with Packet Spraying

Packet spraying is a load balancing method where individual packets from the same flow are distributed across multiple equal-cost paths. The idea is to use all available links evenly and reduce the chance of congestion on any single path.

In a RoCEv2-based AI backend network, however, packet spraying presents serious challenges. RoCEv2 relies on lossless and in-order packet delivery. When packets are sprayed over different paths, they can arrive out of order at the destination. This packet reordering can disrupt RDMA operations and reduce the overall performance of GPU-to-GPU communication.

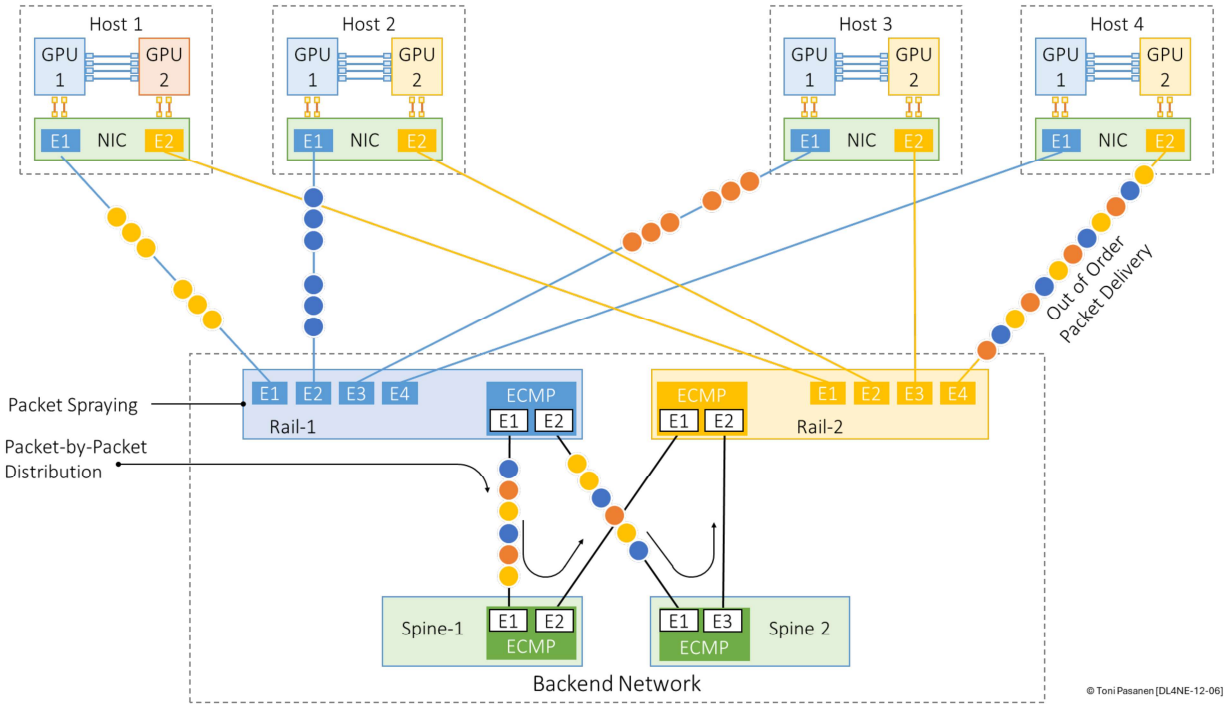

Figure 12-6: *Packet Spraying: OpCode: RDMA Write First, Middle, and Last.*

RDMA Write Only

NVIDIA's RDMA NICs starting from ConnectX-5 support the RDMA Write Only operation, where a RETH header is included in every packet. Figure 12-7 shows how the RDMA NIC uses the OpCode: RDMA Write Only in the IBTH header for each message. With this OpCode, every message also includes a RETH header, which holds information about the destination memory block reserved for the data carried in the payload. This allows the receiving RDMA NIC to write data directly to the correct memory location without relying on prior messages in the transfer sequence.

RDMA Write Only, when combined with Packet-Based Load Balancing using Packet Spraying, brings significant benefits. Since each packet is self-contained and includes full memory addressing information, the network fabric can forward individual packets over different paths without worrying about packet ordering or context loss. This enables true flowlet or even per-packet load balancing, which helps spread traffic more evenly across available links, avoids hotspots, and reduces queuing delays.

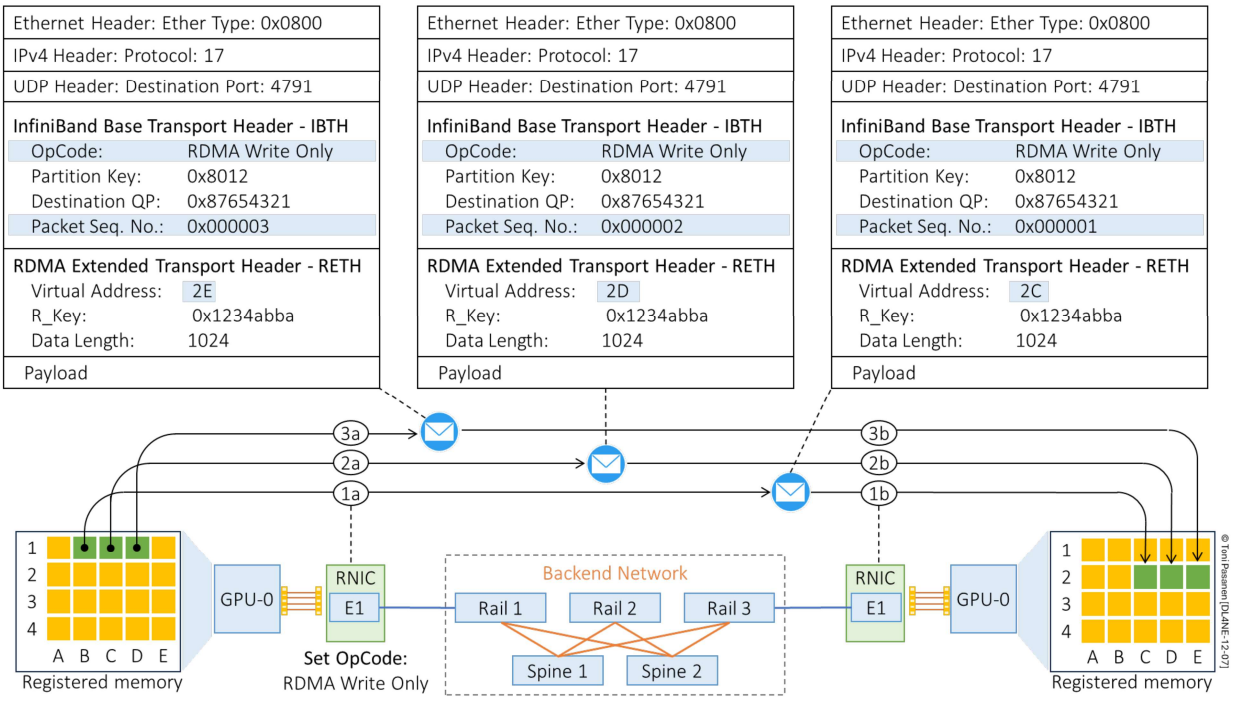

Figure 12-7: *Packet Spraying: OpCode: TDMA Write Only.*

Configuring Per-Packet Load Balancing on Cisco Nexus Switches

At the time of writing, Cisco Nexus 9000 Series Cloud Scale switches (9300-FX3, GX, GX2, and HX-TOR), starting from NX-OS Release 10.5(1)F, support Dynamic Load Balancing (DLB)—including flowlet-based and per-packet (packet spraying) load balancing. DLB is supported on Layer 3 physical interfaces in IP-routed and VXLAN fabrics for unicast IPv4 and IPv6 traffic.

When DLB is enabled, egress QoS and access policies are not applied to flows using DLB. Similarly, TX SPAN configured on an egress interface does not capture DLB traffic. For hardware and software support details, refer to Cisco's official documentation.

Example 12-1 shows a basic configuration for enabling per-packet load balancing:

```
switch(config)# hardware profile dlb
switch(config-dlb)# dlb-interface Eth1/1
switch(config-dlb)# dlb-interface Eth1/2
switch(config-dlb)# mac-address aa:bb:cc:dd:ee:ff
switch(config-dlb)# mode per-packet
```

Example 12-1: *Configuring Per-Packet Load Balancing Packet Spraying.*

Note: The DLB MAC acts as a virtual next-hop MAC address. It's not tied to any specific physical interface. This decouples the MAC from the physical path, allowing the switch to choose a different egress port for each packet. The same DLB MAC address must be configured on all participating switches. If you do not specify a DLB MAC, the default DLB MAC 00:CC:CC:CC:CC:CC is applied.