Spirent™

VERSION 1.0

AI/ML Data Center Network Validation

Technical Brief

Table of Contents

AI/ML Data Center Network Validation

Concepts, Challenges, and Solutions for Assessing AI/ML Network Infrastructures

Introduction

AI Is Reshaping the Data Center

The scale and complexity of artificial intelligence (AI) and machine learning (ML) workloads have redefined data center design. To train trillion-parameter models efficiently, data centers are deploying thousands of GPUs and xPUs across high-speed fabrics. These distributed clusters must operate as a single, tightly synchronized computing platform.

This shift places immense pressure on the network. AI workloads demand low-latency, high-throughput, and lossless communication. Any packet drop or delay can stall the entire training process. As a result, the **network has become a critical performance component** of AI infrastructure [Ref1].

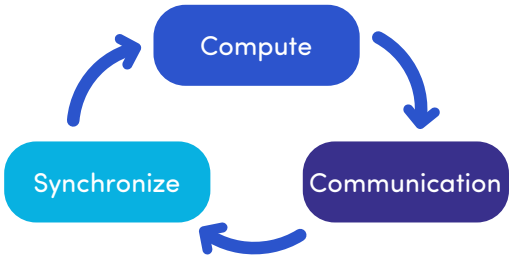
Spirent's testing solutions help organizations ensure their AI data center networks are optimized, reliable, and ready to scale.

Key Networking Concepts for AI Workloads

AI Traffic Patterns and Collective Communication Library (CCL)

AI workloads rely on a unique class of traffic patterns designed for high-volume, parallel data processing across distributed systems. Training modern AI models involves breaking massive datasets into chunks and distributing them across multiple xPUs (e.g., GPUs or custom accelerators) for simultaneous processing. The results from these operations must be synchronized repeatedly during training, requiring precise coordination and ultra-reliable network communication.

- Traffic patterns consisting of a large portion of Elephant flows
- Data and compute-intensive workloads
- Requires large number of short remote memory access
- Nodes start transmitting at the same time
- Progression of all nodes held back by any delayed flow
- AI applications are highly sensitive to network resiliency



Thousands of synchronized parallel jobs

Collective Communication Library is a software library designed to facilitate efficient data exchange and synchronization among multiple processes in parallel and distributed computing environments. It provides optimized implementations of collective communication operations, which enable groups of processes to work together on communication tasks.

NVIDIA's NCCL implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and networking. NCCL provides routines such as all-gather, all-reduce, broadcast, reduce, reduce-scatter as well as point-to-point send and receive that are optimized to achieve high bandwidth and low latency over PCIe and NVLink high-speed interconnects within a node and over NVIDIA Mellanox Network across nodes [Ref2].

NCCL finds significant application in Deep Learning Frameworks, where the AllReduce collective is heavily used for neural network training. Efficient scaling of neural network training is possible with the multi-GPU and multi- node communication provided by NCCL.

RingAllReduce

RingAllReduce is a distributed algorithm used primarily in deep learning for efficiently averaging gradients across multiple devices (such as GPUs or nodes) in a distributed training setup. Devices are arranged in a logical ring (which can be treated as one communicator). Each device communicates only with its immediate neighbors (the next and previous devices in the ring).

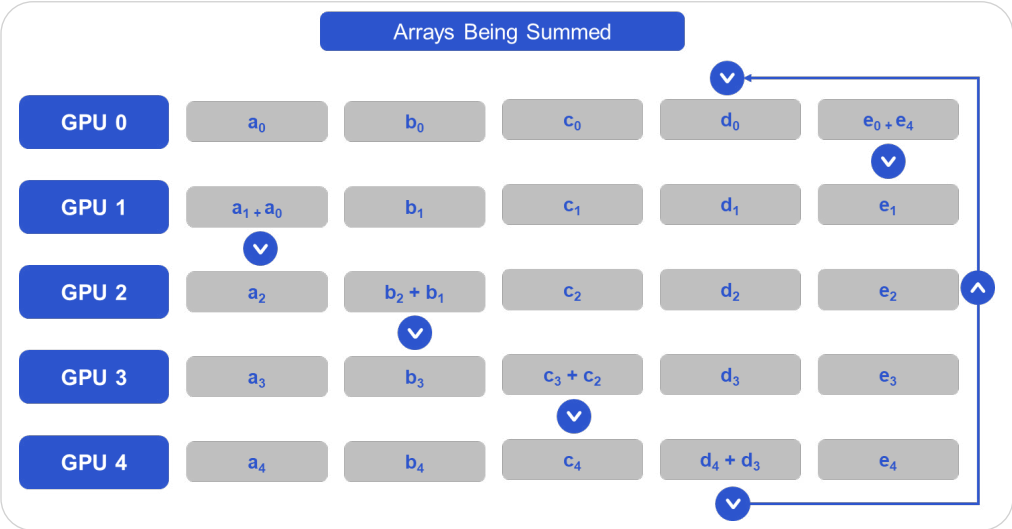
It has two phases: ReduceScatter (Aggregation Phase) and AllGather (Distribution Phase). [Ref3].

ReduceScatter phase:

1. The total data is split into N chunks, where N is the number of devices (or processes).
2. Each device sends one chunk of its data to the next device while receiving a chunk from the previous device.
3. Upon receiving a chunk, each device performs the reduce operation combining the received data with its local chunk. We will use the Sum operation as an example in this documentation.
4. This process continues for N - 1 steps, after which each device holds one chunk of the reduced result.

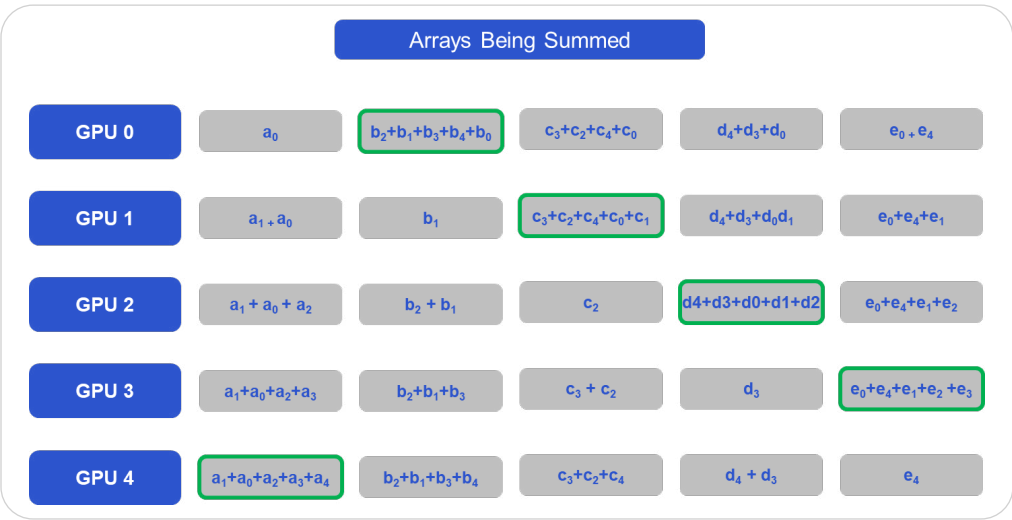
First step of ReduceScatter

After the first send and receive is completed, each GPU will have a chunk that consists of the sum of the corresponding chunk on two different GPUs. For example, the first chunk on the second GPU will be the sum of the values in that chunk from the second GPU and the first GPU.



Intermediate sums after the first iteration of ReduceScatter is complete

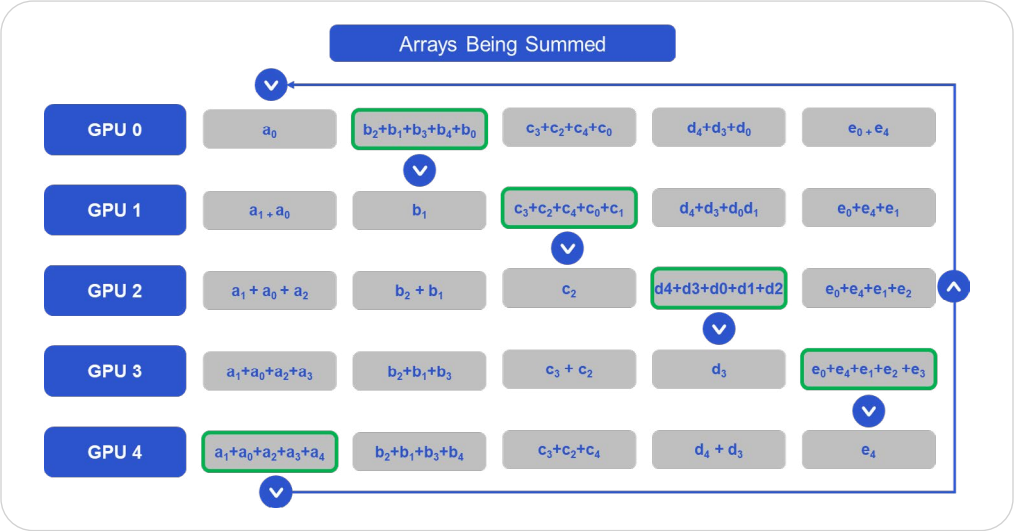
In NCCL operations arguments -o,--op <sum/prod/min/max/avg/all> specify which reduction operation to perform. Only relevant for reduction operations like Allreduce, Reduce or ReduceScatter. Default : Sum [Ref4].



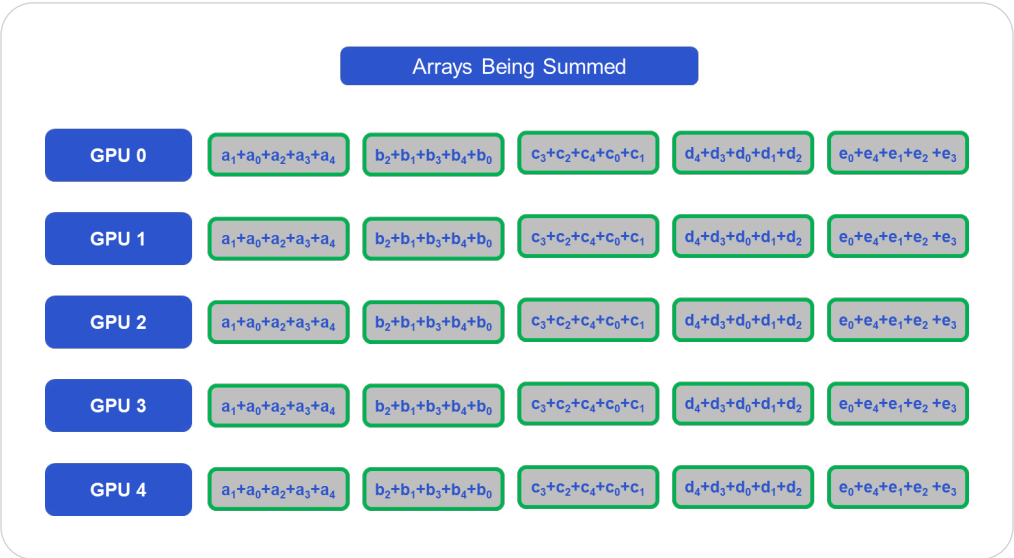
Final state after all ReduceScatter transfers

AllGather phase:

1. Each device passes its reduced chunk to the next device, while receiving another missing chunk from the previous device.
2. After N - 1 more steps, all devices receive the complete aggregated data.



Data transfers in the first step of the AllGather

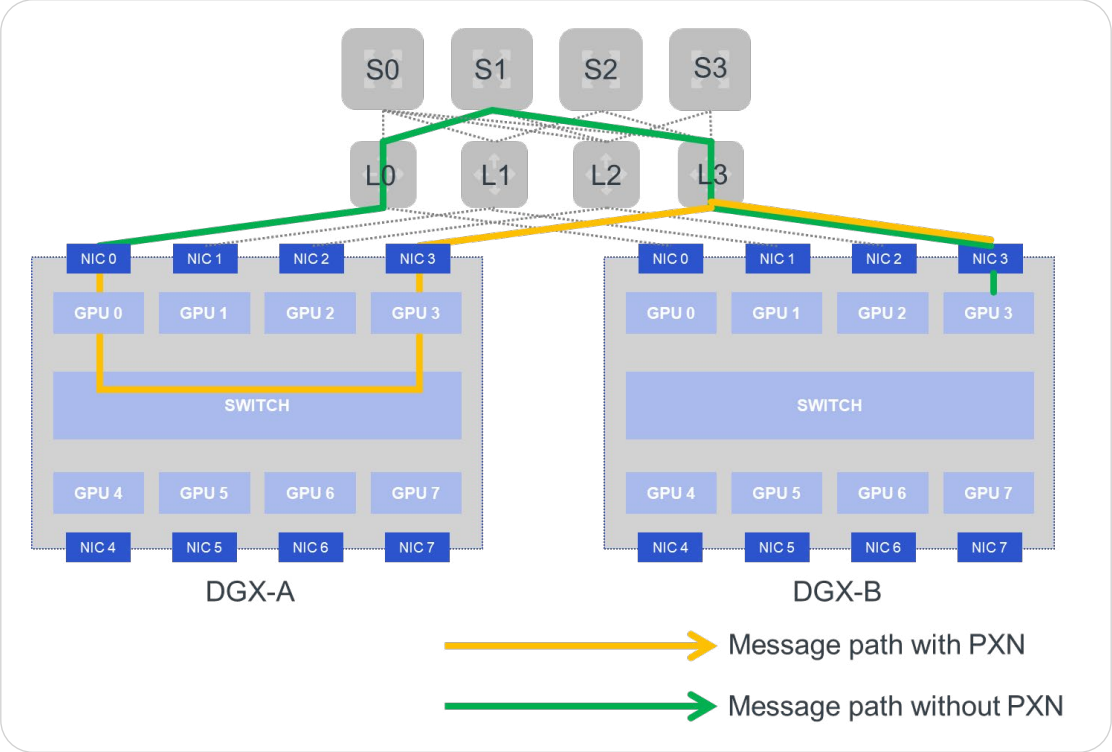


Final state after all AllGather transfers

By the end of AllGather phase, each GPU will have the fully accumulated values for the entire array. All gradients have been synchronized among all nodes.

AlltoAll

One of the most demanding communication patterns is **AlltoAll**, in which each processor exchanges data with every other processor in the cluster. This results in extremely dense communication flows and places significant demands on switch fabrics. All-to-all communication is a communication pattern where each xPU exchanges data with every other xPU. In NCCL 2.12, NVDIA introduced a new feature called PXN which optimizes message paths to improve efficiencies. [Ref5].

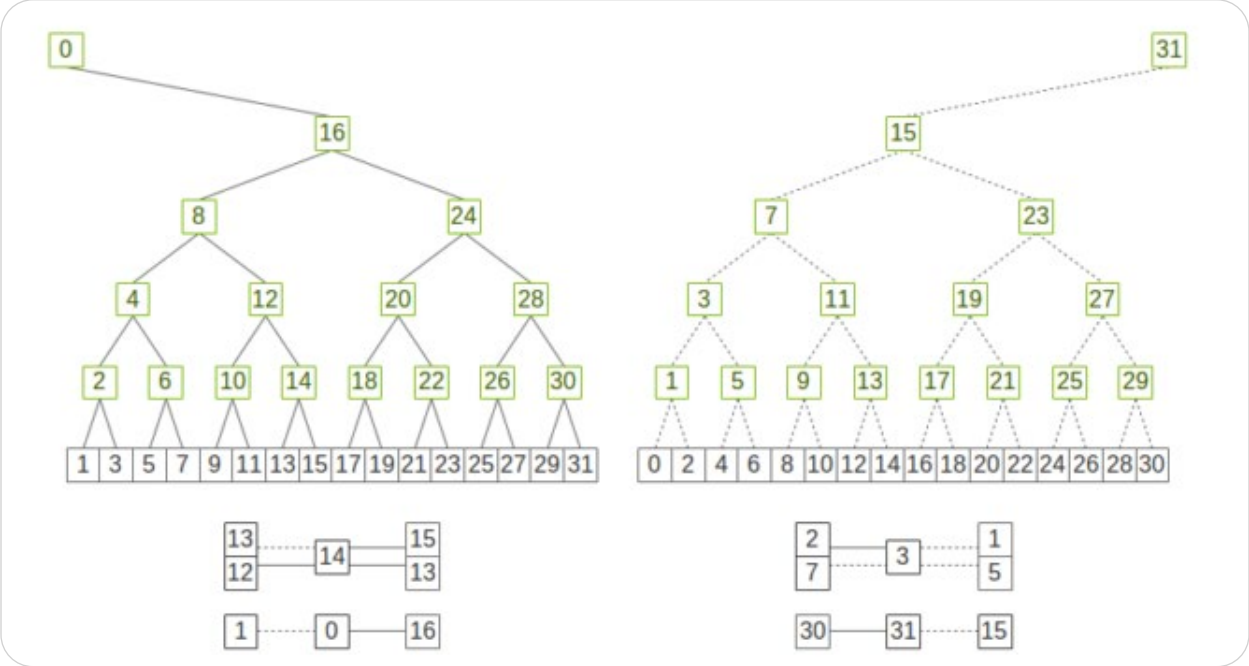


AlltoAll Example message path from GPU0 in DGX-A to GPU3 in DGX-B

In the Rail-optimized network topology, NIC-0 from each DGX system is connected to the same leaf switch (L0), NIC-1s are connected to the same leaf switch (L1), and so on. Without PXN, the message in the figure above would have traversed through three hops of network switches (L0, S1, and L3), potentially causing contention and being slowed down by other traffic. Messages passed between the same pair of NICs are aggregated to maximize effective message rate and network bandwidth.

Double Binary Trees

NCCL 2.4 introduced double binary trees, which offer full bandwidth and a logarithmic latency, even lower than that of 2D ring latency. In a double binary tree, half of the ranks in the first binary tree are nodes and the other half ranks are leaves. The second binary tree reverses this, using the leaves as nodes and vice versa for each binary tree. The figure illustrates how this pattern can be used to build a double binary tree by flipping the tree to invert nodes and leaves. [Ref6]



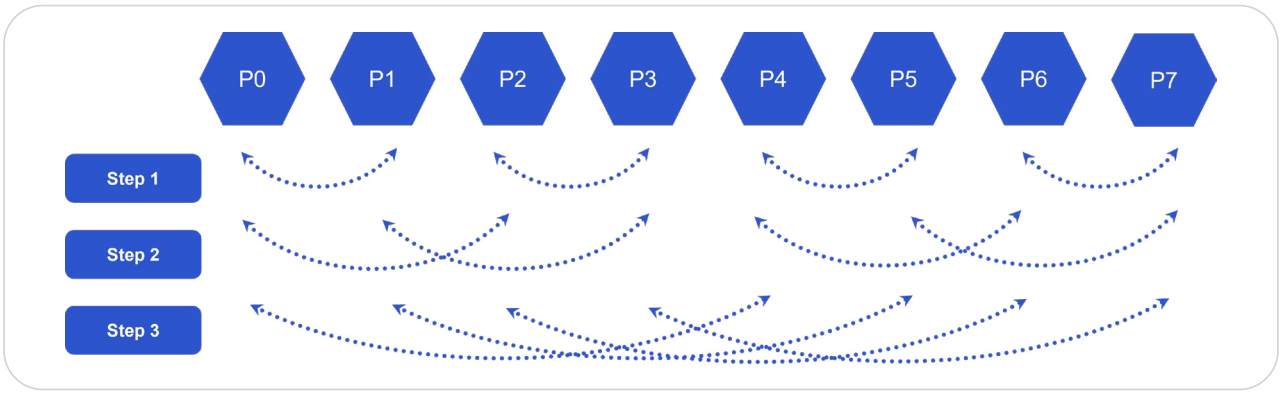
Two complementary binary trees

If you superimpose the two trees, all ranks have two parents and two children except for the root ranks, which have one parent and one child. If we use each of the two trees to process half of the data, each rank will, at most, receive and send half of the data twice, which is as optimal as rings in terms of data sent and received.

Halving Doubling

This algorithm combines a ReduceScatter, implemented with recursive vector halving and distance doubling followed by an AllGather, implemented by a recursive vector doubling combined with recursive distance halving (for AllReduce) [Ref7].

In the first step, the number of processes p is reduced to a power-of-two value. The first 2r processes perform pairwise from each even rank to the odd (rank + 1) the second half of the input vector and from each odd rank to the even (rank – 1) the first half of the input vector. All 2r processes compute the reduction on their respective half. The first step concludes by sending those results from each odd process (1 ... 2r - 1) to rank - 1 into the second part of the buffer.



Recursive Halving and Doubling for ReduceScatter

In the second step, the even/odd-ranked processes are sending the second/first half of their buffer to rank'+1 / rank'-1. In the next steps, the buffers are recursively halved and the distance is doubled. This process continues until the ReduceScatter stage is complete.

AllGather is the reverse procedure of ReduceScatter without operation.

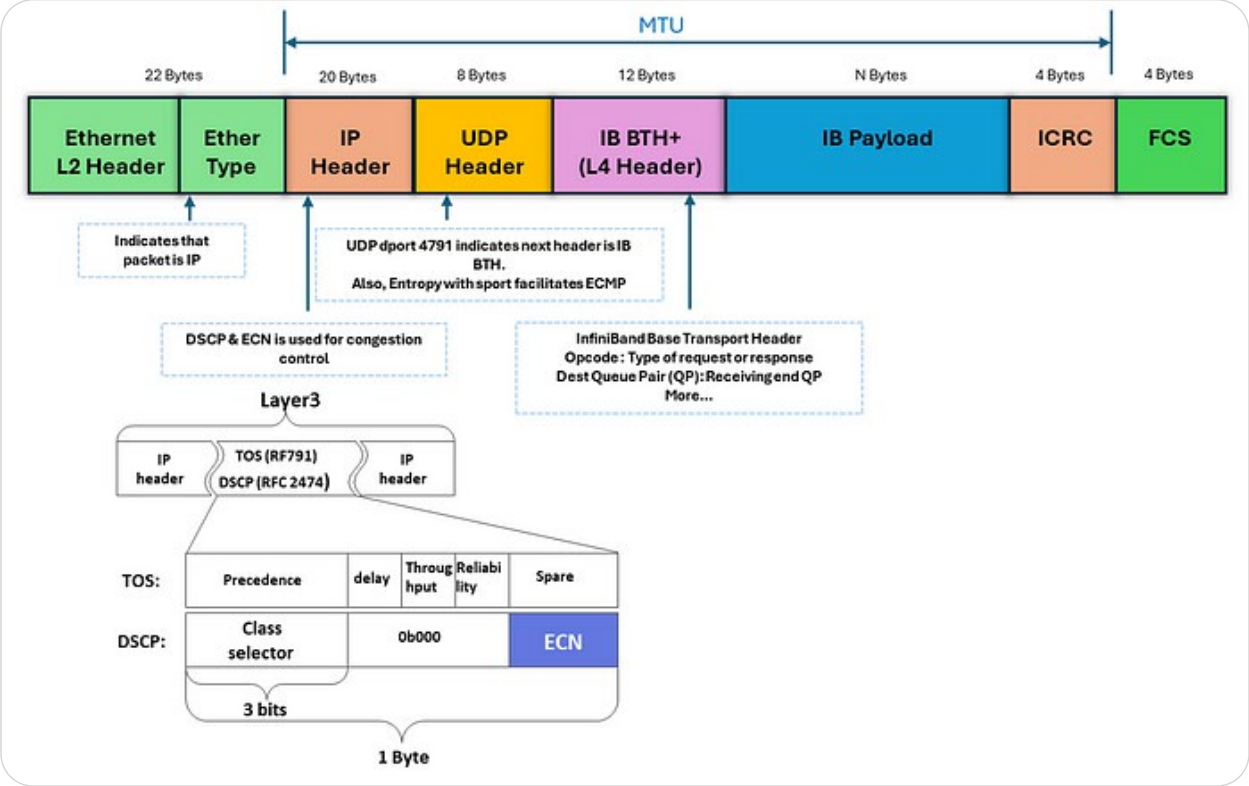
Depending on the number of xPUs, the vector length and the reduction routine, one or multiple traffic patterns will be invoked during training. Spirent TestCenter can emulate combinations of these traffic patterns across multiple ports. Additionally, non-AI training traffic can be added as background traffic.

RDMA over Converged Ethernet version 2 (RoCEv2)

Traditional networking stacks introduce latency due to CPU processing and kernel interaction. Remote Direct Memory Access (RDMA) eliminates these bottlenecks by enabling direct data transfers between the memory of different nodes without CPU involvement. This is especially beneficial for AI workloads, where synchronization events must be frequent, fast, and with minimal delay.

RoCEv2 (RDMA over Converged Ethernet version 2) is the most widely adopted RDMA protocol for data centers. It operates over standard Ethernet and supports routing across Layer 3 networks using UDP encapsulation. This flexibility allows AI workloads to scale across large, multi-rack, and even multi-site deployments.

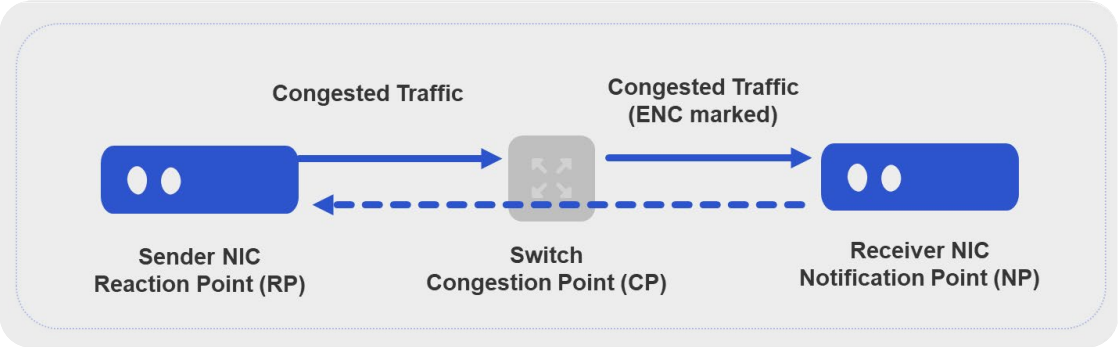
Since its introduction in 2014, RoCEv2 has been widely adopted in data centers for its low-latency, high- bandwidth capabilities. Its congestion control mechanism leverages IP ECN (Explicit Congestion Notification) bits for congestion marking and Congestion Notification Packets (CNPs) for acknowledgment and flow regulation. [Ref8].



RoCEv2 Date Frame Format

For RoCEv2 to function effectively, the underlying network must support **lossless transport.** This requires advanced congestion management protocols, including:

- **Priority Flow Control (PFC):** Prevents packet loss at Layer 2 by pausing traffic per priority class when buffers are full.
- **Explicit Congestion Notification (ECN):** Marks packets experiencing congestion so endpoints can reduce their transmission rate.
- **Data Center Quantized Congestion Notification (DCQCN):** A congestion control algorithm which reduces the transmission rate multiplicatively during congestion and gradually increases rate when congestion is cleared. This helps prevent network congestion and packet loss while maintaining high throughput and low latency in RDMA-based networks.
- **Congestion Notification Packets (CNP):** Part of DCQCN (Data Center Quantized Congestion Notification), used to inform senders to reduce their rates adaptively.



DCQCN Congestion Control Flow

Data Center Quantized Congestion Notification (DCQCN)

The pivotal congestion control mechanisms facilitated by switches and NICs are DCQCN and Priority Flow Control (PFC). In these large-scale environments, incorrect or non-optimized network settings can lead to poor application performance. Therefore, it is critical to validate switch fabric performance, optimize configurations and ensure network stability under congestion conditions.

DCQCN is a congestion control algorithm specifically designed for RoCEv2. It helps prevent network congestion and packet loss while maintaining high throughput and low latency in RDMA-based networks. When a sender receives a CNP, it reduces the transmission rate multiplicatively. If no congestion is detected over time, the sender gradually increases its rate additively. This ensures that network utilization remains high while avoiding congestion.

As the receiver:

- If a marked packet arrives for a flow, and no CNP has been sent for the flow within the last N microseconds, a CNP is sent immediately.
- The receiver generates at most one CNP packet every N microsecond (CNP Generation Interval) for the flow, if any packet that arrives within that time window was marked with ECN flag bit set [Ref9].

As the sender:

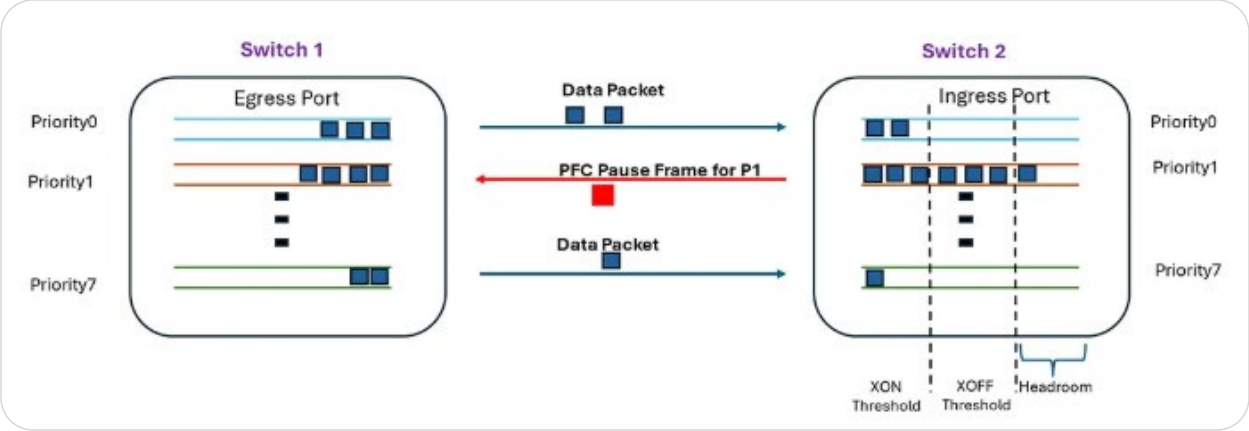
The Sender should increase the injection rate on a QP when a configurable amount of elapsed time and/or a configurable number of bytes have been transmitted on that QP since the reception of the most recent RoCEv2 CNP for that QP. In addition, it maintains a parameter called Alpha, that estimates the congestion level in the network and is used for rate reduction.

In the rate decrease phase, the QP rate is reduced if CNP was received for this QP. Alpha is used to control decrease rate until the Minimum Rate is reached, using the formula: RC (Current rate) = RC (1- Alpha/2) (0<Alpha < 2). If the CNP is received continuously, the rate will be reduced to the configured minimum rate.

When the Time Reset or Byte Reset threshold is reached after the last reduction, the system enters the increase phase. The rate will increase to the target rate.

Priority Flow Control (PFC)

PFC operates at Layer 2 (Data Link Layer) and is based on IEEE 802.1Qbb. Network traffic is divided into 8 priority levels (0-7). Each priority can be controlled independently.



PFC Operation

- When the switch buffer nears overflow (indicated by the Xoff threshold, signaling high buffer utilization in a specific priority queue), the switch sends PFC PAUSE frames to alert upstream ports to halt data transmission.
- Once buffer usage fails below the Xon threshold, the switch prompts the upstream port to resume traffic. [Ref10]

Each PFC pause frame includes a 2-octet timer value for every priority, indicating the duration for which traffic should be suspended. The timer is measured in pause quanta, where a quantum represents the time required to transmit 512 bits at the port's speed, ranging from 0 to 65535. Xon frame has zero Time value for an enabled priority to resume traffic.

PFC can be used to prevent traffic loss when congestion occurs on Layer 2 or Layer 3 interfaces. In Layer 2, Priority Code Points (PCP) in the VLAN headers can identify the priority of the traffic. Differentiated Services code point (DSCP) values in the Layer 3 IP header of the traffic are mapped into PCP values of layer 2.

The combined use of PFC and DCQCN optimizes RDMA performance. PFC efficiently manages congestion by slowing down senders. DCQCN effectively mitigates congestion per flow by signaling congestion anywhere along the data path to endpoints. DCQCN could serve as the primary congestion management mechanism with PFC acting as the fail-safe solution.

Common AI Testing Challenges

AI training traffic significantly differs significantly from traditional network traffic, introducing unique challenges that require specialized testing approaches. One prominent issue is the high-volume, synchronized bursts of data generated by AI workloads. These intense data flows can overwhelm network buffers and queues. Traditional testing methods, often based on enterprise or web traffic patterns, fail to adequately simulate these demanding conditions.

Another critical factor is the dominance of East-West traffic within AI clusters. Unlike typical data center operations, that prioritize North-South traffic (client-server interactions), AI training involves extensive lateral data movement between GPUs or xPUs. This full-mesh communication places significant demands on switch fabrics, requiring them to support simultaneous, high-speed communication across multiple nodes without creating bottlenecks.

Effectively managing congestion through fine-tuned control protocols like Priority Flow Control (PFC) and Data Center Quantized Congestion Notification (DCQCN) is also crucial. Misconfigured congestion management can lead to packet loss, delayed training jobs, or underutilized network links.

Quality of Service (QoS) misconfigurations, such as incorrect VLAN tagging, improper queue mapping, or inadequate buffer allocations, can silently degrade performance, making troubleshooting particularly challenging. Moreover, the scale and interdependencies of AI workloads complicate root cause isolation, as issues often involve multiple components and network layers simultaneously.

Testing through precise traffic emulation and advanced analytics, helping organizations identify exactly where and why performance degradation occurs.

Statistics That Reveal AI Network Health

Monitoring the health and performance of AI network fabrics requires capturing and analyzing specific, actionable metrics. Packet loss, for instance, signals failures in critical updates needed during training. Job Completion Time (JCT) reflects the overall speed and efficiency of training workloads traversing the network.

Tail latency, representing worst-case delays, can significantly slow down training jobs, while metrics such as dropped packet counts and reordered packets highlight congestion issues or problems with Equal-Cost Multi-Path (ECMP) routing behaviors. Tracking deviations in transmit (Tx) and receive (Rx) rates, will identify underutilized links and traffic imbalances.

Additionally, visibility into ECN, CNP, and PFC activities provides essential insight into flow control operations, enabling network teams to diagnose congestion handling issues proactively before they become performance issues.

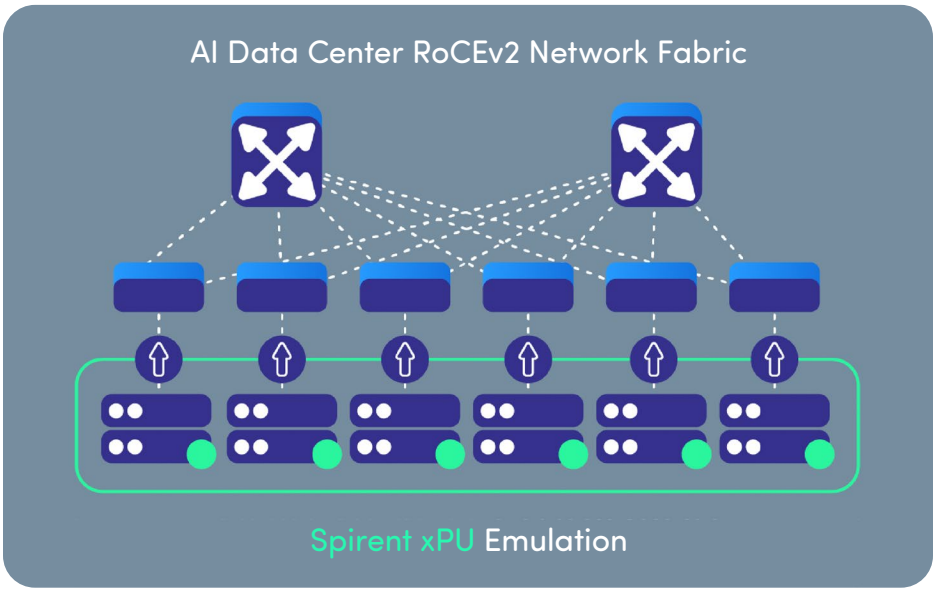
Pinpointing precisely which queue pair (QP) or port suffers from packet loss during peak loads, isolate problematic links contributing to tail latency, and validate whether congestion management mechanisms are functioning as intended. By identifying these issues early, Spirent ensures network problems do not impact live AI training operations.

Examples of Problem Indicators

Issue Observed	Possible Cause	Testing Insight
Packet loss during peak load	Misconfigured PFC thresholds or switch buffer overflow during synchronized training events	Pinpointing the affected queue pair (QP) or switch port and identifies the load level at which loss begins
Long tail latency in training	Flow path imbalance or resource contention at specific switch stages	Revealing which links or flows are delayed and correlates delays with topology and configuration
High JCT variance	Inconsistent ECN/CNP responsiveness or queue buildup	Compare algorithm performance under load, tracking JCT changes and degraded iterations
Congestion with no rate drop	ECMP algorithm or network topology needs optimization	Verify if there is congestion and no packet drop, STC will decrease the traffic rate during congestion

Spirent TestCenter AI Testing Solution Overview

Spirent TestCenter delivers high-density, multi-speed performance testing for AI data center environments. It supports the emulation of realistic AI workloads, including traffic based on RDMA (RoCEv2) protocols and Collective Communication Library (CCL) patterns such as AlltoAll, RingAllReduce, and more. This makes it ideal for validating fabrics that must support synchronized, high-bandwidth, and low-latency AI training communication.



Spirent's test appliances include the A1 and B3 series of High-Speed Ethernet appliances.

The A1-400-QD-16 platform offers up to 16 ports of 400G Ethernet and supports RoCEV2 on 100G, 200G and 400G, making it highly flexible for labs needing multi-user, multi-speed environments.

Spirent A1 400G 16-port Appliance

The B3 platform supports both QSFP-DD and OSFP 800G interfaces and offers up to 6.4 Tbps of traffic generation, providing an industry-leading port density advantage.

Spirent B3 800Gbps 8-Port and 4-Port Appliance

With built-in support for RoCEv2 features such as DCQCN, ECN and CNP, along with automation frameworks for performance benchmarking, Spirent enables users to simulate east-west AI traffic patterns across leaf-spine topologies, stress-test networks under dynamic congestion and traffic imbalance conditions and validate flow control mechanisms like DCQCN and PFC in high-performance environments.

Spirent TestCenter's automation tools allow testing across varying frame sizes, data sizes, port speeds, and traffic patterns, supporting continuous integration (CI/CD) workflows. The solution delivers actionable results through advanced reporting and interactive dashboards, empowering network architects to fine-tune settings, troubleshoot issues, and validate readiness for AI workloads.

Enabling Reliable, Scalable AI Data Center Operations

As AI workloads continue to scale, the performance of the underlying network fabric plays a critical role in ensuring timely, efficient model training and inference. The pressure to deliver consistent throughput, low latency, and reliable synchronization is especially high in distributed environments where even minor delays can lead to significant performance degradation or resource underutilization.

To address these challenges, organizations require test strategies and tools that reflect the unique demands of AI traffic—especially collective communication patterns, RoCEv2 transport behavior, and the effects of congestion and flow control. Early visibility into network behavior under stress is key to optimizing system design, configuration, and deployment.

The ability to emulate AI-specific traffic patterns and measure job completion time, packet loss, and tail latency helps engineers identify bottlenecks, validate configurations, and improve performance tuning. With these capabilities, teams can evaluate switching fabrics, congestion management mechanisms, and QoS policies before they impact live training operations.

By integrating traffic emulation, performance benchmarking, and flow-level analytics into the development and test process, network and infrastructure teams are better equipped to make informed decisions, reduce deployment risk, and deliver infrastructure that meets the demands of AI-scale computing.

To explore test strategies that support scalable, high-performance AI networks, visit spirent.com/solutions/testing-for-ai-networking.

Reference

[Ref1] *Aniket Khosla*, How AI Changes the Game for High-Speed Ethernet, *Spirent Byline, January 2025*

[Ref2] https://developer.nvidia.com/nccl

[Ref3] https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/

[Ref4] https://github.com/nvidia/nccl-tests

[Ref5] https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/

[Ref6] https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/

[Ref7] Rolf Rabenseifner, Optimization of Collective Reduction Operations, International Conference on Computational Science, June 7–9, Krakow, Poland, LNCS, Springer-Verlag, 2004.

[Ref8] https://en.wikipedia.org/wiki/RDMA_over_Converged_Ethernet

[Ref9] https://medium.com/@ravikishorechitakani/optimizing-ai-ml-and-hpc-workloads-exploring-rdma-rocev2-for-high-performance-data-center-8d130cda74ae

[Ref10] https://medium.com/@ravikishorechitakani/optimizing-ai-ml-and-hpc-workloads-exploring-rdma-rocev2-for-high-performance-data-center-8d130cda74ae

Americas 1-800-SPIRENT
+1-800-774-7368 | sales@spirent.com

Europe and the Middle East
+44 (0) 1293 767979 | emeainfo@spirent.com

Asia and the Pacific
+86-10-8518-2539 | salesasia@spirent.com

Rev B | 05/25