# AI Network Solution Guide

# Table of contents

# Table of contents

Arista is an industry leader in AI Networking. This document is an high-level solutions guide for building AI Networks with Arista. The purpose of this document is to lay out a range of core concepts, considerations, and principles that must be understood when creating an AI Network design. It is intended to be paired with a detailed design. Additional depth is available in our sample reference architecture documents and from Arista directly.

This document begins by introducing some core AI Networking fundamentals, followed by an introduction to building AI Networks with Arista. The document then describes the design of Scale Out networks in depth, covering key concepts, design principles, the implications of AI server architecture, and network sizing, before providing some high-level architectural guidance. A separate sample reference architecture describes one commonly implemented reference design in depth and provides the information required to tailor a design for a particular customer use-case. But given we have hundreds of AI customers with widely varying requirements, Arista typically works directly with customers to build design recommendations that are customized to their needs.

**Note:** this document focuses primarily on building AI Scale Out (aka Back End) networks as this is where the need is strongest. Future revisions of this document will cover other network fabrics in more depth.

## Fundamentals of AI Networking

### Four types of AI networks

Four distinct types of network (and three new AI-specific Fabrics) arise frequently in discussions about AI Networking. These types cover most customer use-cases, though all types are not relevant to all customers. Figure 1 below presents an overview of the different networks.

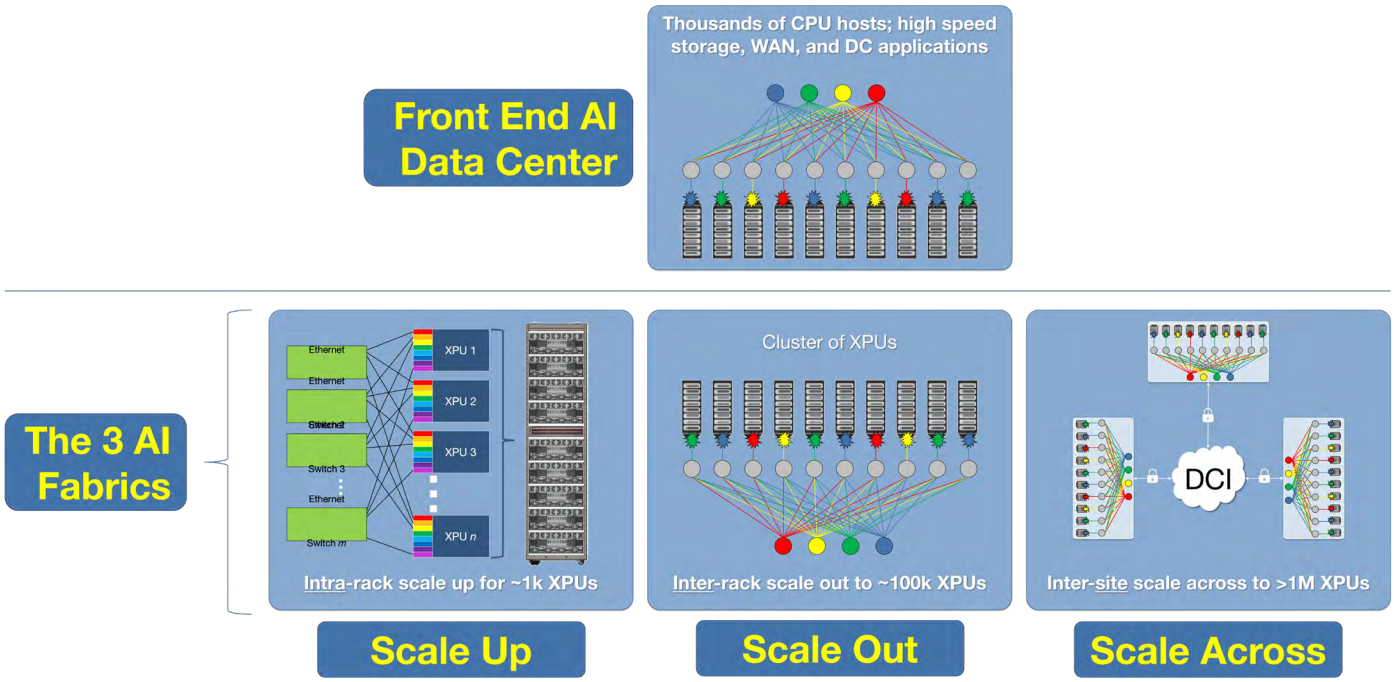


*Figure 1: The four AI Networks*

### Front End Network

All AI deployments have some form of front end network. The Front End network connects all of the AI server CPUs together as well as providing access to a variety of other services. These services include high-capacity high-speed storage, non-AI compute servers, and access to external Cloud services and wide-area networking. Within an AI server, the front end network typically attaches to a dedicated set of CPU-attached NICs.

This network is typically an extension to an existing network, as a Front End network is commonly deployed for other DC and Cloud use-cases. There are some differences in AI use-cases. It is typical to provision much larger per-server bandwidth (200G/400G NICs are already common and 800G NICs are beginning to appear). The additional bandwidth is primarily due to a need for high-bandwidth access to storage. As a result, it's also common to design the network facing the AI servers with lower over-subscription than other DC networks and to include high-bandwidth dedicated non-oversubscribed storage sub-networks.

### Scale Out AI Fabric

The Scale Out (or Back End) AI Fabric connects all the XPUs in a deployment together to form an AI/ML Cluster. It is common for XPUs to have dedicated NICs and these NICs are interconnected to form the Scale Out network. In some cases, specialized storage or memory cache systems may also be connected to the Scale Out network. The Scale Out network may be an isolated island or it may provide connectivity to the Front End network. It may also connect to Scale Out networks in other datacenters through an inter-DC Scale Across network. This is often the first AI network discussed and is often the largest and the most complex to design. The performance of the Scale Out network is critical to the performance of large-scale AI workloads.

### Scale Up AI Fabric

In addition to dedicated NICs for access to Scale Out networking, XPUs frequently have local connectivity also. This connectivity may only extend to the small number of XPUs within a single server (typically, 4-8 XPUs). But in some cases, the Scale Up fabric extends to all of the XPUs in a larger physical domain, such as an enclosure, a rack or a small set of adjacent racks. Typical Scale Up network sizes today range from 4 to 72 XPUs, but future designs may increase this to ~1K XPUs.

The bandwidth and performance requirements of the Scale Up network are typically higher than in the Scale Out case. It is common for an XPU to have 10x as much Scale Up bandwidth as Scale Out. Scale Up networks are typically used for direct memory transactions between XPUs and have stringent latency and reliability requirements. In some cases, XPUs are directly meshed together. Alternatively, single-hop switched networks are common. The network implementation may be proprietary (e.g. NVIDIA NVLink, AMD Infinity Fabric) or more open (e.g. PCIe, AMD Infinity Fabric over Ethernet, Intel Gaudi).

### Scale Across AI Fabric

As the scale of AI Cluster builds has grown and the challenges of power provision for AI datacenters have increased, AI/ML clusters can span multiple datacenters. This creates a requirement to connect the Scale Out fabrics in two or more datacenters together. This is referred to as the Scale Across AI Fabric. Many of the challenges are similar to other forms of inter-datacenter connectivity, but it is useful to distinguish the AI use-case because the requirements can be slightly different. The bandwidth demands are often far larger than traditional inter-DC networks. Additionally, latency is a key concern that dominates multi-DC performance for some types of AI workloads.

## Types of AI workload

AI workloads are continuously evolving and every major conference in the field discusses new algorithms and configurations. As a result, flexibility and fungibility are often key goals for AI Infrastructure design. The majority of the current industry focus relates to the creation and use of large language models (LLMs), but it is worth noting that other types of machine-learning solutions have been developed and are used to address specific use cases. Broadly speaking, AI workloads are divided into two major categories: training and inference. A specific AI Cluster may target a specific type of workload or may be designed to support a wide range of workloads.

### Training workloads

The foundational step in building a new AI model involves learning general patterns from a vast set of input data. This step is more accurately referred to as "pre-training", but is commonly just referred to as "training". The input data may be text-only or may include multiple types of media, such as audio, images, and video. The input data is typically unlabelled: this means that it is raw data and has not been carefully categorized and tagged. Huge volumes of storage may be required, particularly for multi-modal training involving video. Training a new model is a complex process that involves synchronously coordinating many XPUs as they iterate through large batches of data. (The duration of these iterations is often the key performance measure for the network.) Training jobs

can run for weeks or months and are extremely sensitive to both network problems and other types of failures. This is because all of the XPUs must synchronize regularly as part of the training process, often every few seconds or less. It is common that the training process generates checkpoints regularly so that it can restart if there is a failure, though a restart results in the loss of all progress made since the last checkpoint, which may be 10s of minutes in the past. Pre-training a new frontier model from scratch can take months of computation on fleets containing 10K-100K+ XPUs.

Once a base model is built through the resource-intensive pre-training process, the next step is often to fine-tune the model (or "post-train") for a specific use-case. This training might be to adapt the model to specific use-case, such as the chatbots that humans typically interact with, or it might be to adapt the model to a specific domain of operation, such as creating a model that has expertise in network troubleshooting, like Arista's AVA. Post-training can also involve compressing or distilling large models to produce smaller, lower-cost models. Post-training is much less resource intensive than training from scratch and may be repeated to build specific models for many different use-cases. Post-training may use smaller amounts of carefully labelled and categorized data to fine-tune the model. Post-training requires substantially less resources than pre-training: 10s to 1000s of XPUs and shorter periods of time.

### Inference workloads

Once a model is built for a specific task, it can be made available to be executed so that it can answer queries or run tasks. It is often accessed through some form of API. For some workloads, multiple models may be linked together and models may call other models in order to form a complete system. An alternate approach to fine-tuning models is common at the inference stage, particularly for enterprise use-cases. Retrieval-Augmented Generation (RAG) fetches relevant data from external sources at inferencing time and adds it to the context for the query. RAG adds fresh, proprietary, and domain-specific data at inferencing time without any need to re-train and allows sources to be cited.

Inference workloads may be real-time, handling requests coming directly from end-users for example, or they may be designed to operate on batches of requests. For some inference workloads, a Scale Out network may not be required as the model fits into a single XPU or a set of XPUs within a single Scale Up domain. In these cases, network access to and from the XPUs running the models occurs entirely via the Front End network. As a result, it is sometimes claimed that Scale Out networks are only required for training workloads. However, as inference models have become larger and more complex and as context lengths for those models have increased, the number of XPUs required to execute an inference model (or set of models) has steadily increased.

As a result, Scale Out networks have become more common even in dedicated Inference clusters, though the size of such dedicated clusters is typically smaller. The hardware requirements for inference range from a single XPU to 10s/100s of XPUs, depending on the size of the model, the size of the context the model operates on, and the performance requirements. There is also a recent trend where different stages of the inference execution are separated in order to allow optimization of the hardware that is used. For example, the initial processing of the input and context (known as prefill) may be separated from the generation of output tokens (decode). This configuration leads to Scale Out network traffic between systems as well as creation a requirement for the Scale Out network to support heterogeneous types of inference XPUs.

**AI workloads on the network**

From a networking point of view, AI workloads are very different from traditional cloud or enterprise datacenter workloads. Many XPUs may be involved in a single AI workload, particularly for training jobs. The XPUs involved in a job need to synchronize regularly as they iterate through a training job. These iterations are often on timeframes measured in seconds or fractions of a second. Large amounts of data (GBs) can be exchanged during these synchronization events. For inference workloads, large amounts of context state information may also be transferred directly between nodes. Additionally, nodes may generate large amounts of storage traffic as they read input data and write output data (including checkpoints of job state to allow for restarts after failures).

These differences lead to a very different network utilization profile to other network use cases. AI networks typically see small numbers of very high bandwidth data flows (often at line rate) and these flows may be tightly synchronized across many XPUs leading to very large peaks in network utilization. In large training networks, almost no network traffic smoothing or statistical multiplexing is seen and even localized congestion or packet loss can lead to a performance impact across the entire cluster.

### RDMA

Remote Direct Memory Access (RDMA) is a critical technology in delivering the scalable parallel processing demanded by AI applications. In traditional general-purpose TCP/IP networking, data for transmission by an application is generally copied from user space to kernel space before reaching the network driver and then the network device for transmission. This process is reversed on the other end. This introduces processing bottlenecks, data movement, and latencies that are unsuitable for the high bandwidth inter-accelerator flows demanded by AI and HPC jobs.

RDMA, on the other hand, enables the direct exchange of data in main memory without relying on the kernel and without copying the data to/from kernel space. Instead, the network device accesses main memory to read or write data directly. In the case of typical AI systems, RDMA is typically used in both CPUs (to/from main memory) and also in XPUs (to/from dedicated XPU memory). In some systems, this enables XPUs to read from and write to the network without involving the host CPU. RDMA dramatically improves throughput and performance, resulting in faster data transfer rates and lower latency between RDMA enabled systems. When RDMA is offloaded to a NIC, there is an asynchronous notification mechanism to signal completion.

#### *RoCE*

In Ethernet networks, the dominant protocol used to implement RDMA is known as RDMA over Converged Ethernet (RoCE). Version 2 of this protocol (RoCEv2) provides RDMA transport over UDP/IP. The use of RoCE requires the underlying network to be configured in a particular way. The precise requirements vary slightly between RoCE implementations but it is typical that the network is required to be configured to be lossless and to have congestion signaling enabled.

In order to support RoCE, QoS is enabled in the network with at least lossless and best effort traffic classes. The Differentiated Services Code Point (DSCP) field in the IP header is used to map traffic into network traffic classes. Priority Flow Control (PFC) is usually enabled to guarantee lossless operation. PFC is a hop-by-hop Layer 2 mechanism that allows a device or host to request that all data transmission for a traffic class be paused when the receiver is at risk of running out of buffer space and potentially dropping packets. Additionally, Explicit Congestion Notification (ECN) is enabled in the network in order to provide end-to-end notification of congestion in the network. ECN uses an IP header field to mark packets when network queues build to allow more granular notification of network congestion. The goal of ECN in this scenario is to enable a sender to reduce their transmission rate before the queues build to the point of triggering PFC. The combination of PFC and ECN to deliver lossless delivery and end-to-end congestion control for RoCEv2 traffic is also known as Data Center Quantized Congestion Notification (DCQCN).

RoCE is implemented in the Ethernet NIC for performance reasons. While RoCE is a somewhat open standard, vendor implementations vary substantially and often include proprietary enhancements, congestion control algorithms, and tuning parameters. This makes RoCE interoperability between different NIC implementations challenging and, as a result, virtually all AI Networks use a single NIC vendor and RoCE implementation for an entire cluster.

#### *Alternatives to RoCE*

There are several shortcomings associated with the implementation of RDMA through RoCEv2 and a DCQCN-enabled Ethernet network, particularly for large-scale or multi-tenant deployments. A wide range of proprietary network and protocol extensions have been bolted on top of RoCE over the years to address these shortcomings, but proprietary solutions work against the competitive forces that have made the Ethernet ecosystem hugely successful over the past 50 years.

The Ultra Ethernet Consortium (UEC), of which Arista is a founding member, is a standards organization that was formed with the goal of enhancing Ethernet for the needs of AI and HPC while maintaining the interoperability that has made Ethernet successful. In 2025, the consortium issued an initial specification and whitepaper. The latest specification is V1.0.3 (July 2026). Ultra Ethernet Transport (UET) forms part of the specification and provides an alternative RDMA transport protocol intended to replace RoCE and address many shortcomings in existing implementations. In particular, UET is designed to operate on a lossy network, removing the need for PFC, and UET uses many paths through the network, reducing the challenges associated with load balancing large flows within the network. Arista's Etherlink portfolio is Ultra Ethernet ready and Ultra Ethernet NICs have begun to appear. For more information, please refer to our white paper: Demystifying Ultra Ethernet.

More recently, OpenAI led a multi-vendor consortium that created and published an open OCP specification called Multipath Reliable Connection (MRC) along with a paper describing their experiences with using MRC to train frontier AI models. MRC is a more targeted solution than Ultra Ethernet. MRC extends RoCEv2 rather than replacing it and only supports the limited subset of RDMA operations used by AI Training workloads. This makes MRC simpler but less general. MRC reuses key elements of Ultra Ethernet, including packet trimming, selective acknowledgments, and congestion control. Like UET, MRC is designed for a lossy network and can adaptively load balance and spray traffic across many paths through the network. MRC can also, optionally, leverage SRv6 in the network to enable direct source routing of traffic through the network. For more information, please refer to our white paper: Multiplanar Scale Out fabrics with MRC.

**Collectives and CCLs**

Given the complexity of AI workloads, it is rare to discuss network communication in terms of raw RDMA traffic. The vast majority of AI frameworks provide an additional layer of abstraction that enables multiple XPUs or CPUs within a job to jointly perform a higher level communication operation. These operations are known as collectives. A single collective can span all the XPUs involved in a job or only a subset. Collectives can operate on large amounts of data in a single operation (MBs or GBs). Collectives provide both relatively simple messaging primitives (e.g., Broadcast, to send data from one node to all nodes) as well as more sophisticated operations that process data in a structured manner as part of the communication operation. For example, an AllReduce collective can combine data from a large set of sources and then distribute the combined data to all participants.

A Collective Communication Library (CCL) is the generic term for the software that implements collectives in a given AI framework. CCLs are often very sophisticated and can structure the underlying communication patterns to leverage different types of physical connectivity in an optimal fashion. Commonly used vendor CCLs include NVIDIA's NCCL and AMD's RCCL. There are also third party CCLs that support multiple types of XPU.

**Server Architecture**

In traditional network design, you may not need to know much more about the servers than the NIC speed and expected utilization levels. AI networks are much more directly influenced by the architecture of the AI systems. There are often multiple NICs for different purposes, some CPU attached and some XPU attached. Additionally, there is often internal connectivity between the XPUs and that can influence network design. Finally, for workload optimization reasons, it is common for a Scale Out network to be specifically designed to accommodate a large homogeneous deployment of identical servers. So, it is important to gather as much information as possible about the intended server architecture that will be deployed before choosing a final network design. This section presents two key aspects to consider: XPU configuration and NIC configuration.

The server architecture for a network design may not yet be fully known. Or support for multiple heterogeneous server architectures on the same network may be required; this is becoming more common as large-scale inference platforms evolve to deploy multiple types of CPU and XPU within the inference pipeline. These requirements can be a strong argument for choosing a more fungible network architecture that can flex as the needs change. Depending on the scale, that might mean a modular chassis or DES solution is preferable to a fixed form-factor solution. Or it might mean using a wide and flexible spine design rather than a smaller one.

XPU Configuration

The first thing to determine is the XPU type, count, and configuration within the server. Specific questions include: the type of XPU, the number of XPUs per server, whether there is a Scale Up network between the XPUs within the server, and whether groups of servers form a larger Scale Up domain. For DC design purposes, you might also want information on power draw, the use of liquid cooling, and whether servers are delivered individually, in pre-integrated rack units, or as a rack-scale system.

The range of options is large and growing. At one end of the spectrum are standalone server solutions that contain between one and eight separate standalone XPUs (often implemented in PCIe form factors). These servers do not have internal Scale Up networks between the XPUs and, for instance, gain no benefit from a Rail-aligned architecture (described below). In the middle of the spectrum are standalone servers that provide a group of XPUs (typically four or eight) with a dedicated Scale Up network within the server. The Scale Up network might be implemented through a proprietary solution such as NVIDIA's NVLink or AMD's Infinity Fabric or it might be Ethernet or PCIe based. These solutions benefit significantly from Rail-aligned designs.

At the upper end of the spectrum are tightly coupled rack-scale solutions that deliver multiple servers together with a Scale Up fabric within the rack. Current examples include: NVIDIA's GB200/300 and Vera Rubin NVL72 products that deliver 18 4-GPU servers together with NVLink switches to create a 72-GPU Scale Up domain; and AMD's Helios AI rack that delivers the same number of AMD GPUs together with an Ethernet-based Scale Up fabric. These solutions require network designs that can scale to large demands and large steps in XPU capacity. They also benefit from Rail-aligned designs, but to a somewhat lesser extent due to the large Scale Up domains that already connect a large number of XPUs.

### NIC Configuration

Of equal importance to Scale Out network design is the NIC count, configuration, and type(s) within each AI server. These details drive key aspects of the optimal design and also impact the network features that are either mandatory or desirable.

The first thing to determine is the number of NICs in each server, their type, and their internal connectivity within the server. It is common for there to be either one or two Front End NICs in each server, often DPU-based Smart NICs with additional capabilities and security features. These NICs are typically attached to the server CPUs via PCIe and are connected to the Front End network, so they can be discounted for Scale Out network design. Additionally, there is typically a separate set of NICs dedicated to serving the XPUs. These NICs are often directly attached (usually over PCIe) to the XPUs, though lower-end servers may attach both XPUs and NICs to the server PCIe connections. It is common for there to be one NIC per XPU, but some server types share NICs between XPUs or provide multiple NICs per XPU. The number of Scale Out NICs and their connectivity to the XPUs can drive important network design elements such as the ideal Rail-aligned architecture. A block diagram of server PCIe connectivity is often a good place to start.

Next we must consider the type and capabilities of the individual Scale Out NICs. One key design input is the capacity and port speeds that the NIC supports. 400G and 800G NICs are now common in AI deployments. Understanding the supported port speeds is important for multi-plane configurations and also because aggregate NIC capacity may not align with the supported port speeds. For example, NVIDIA's Connect8-X is an 800Gb NIC, but the maximum Ethernet port speed supported is 400GbE, so the network must support 2x400GbE links.

The other key input is the capabilities of the NIC in terms of "smart" or hardware accelerated features. As mentioned earlier, for instance, multi-plane operation typically requires NIC support for efficiency. A NIC may only support standard RoCEv2 RDMA transport or there may be support for additional transport protocols including both proprietary extensions as well as standards-based alternatives such as Multipath Reliable Connections (MRC) or Ultra Ethernet Transport (UET). And it is important to note that a NIC may support physical port breakout without necessarily supporting optimized multi-planar RDMA transport in that configuration. These choices influence the optimal network design and may eliminate some options.

Finally, it can also be beneficial to understand whether alternative NICs can be deployed in the chosen server architecture. Some server architectures tightly integrate a specific NIC. Other architectures allow the customer to choose the NIC to employ for their use case. A summary of the key features of a number of commonly-used AI NICs appears in Table 1 below. Note that neither Ultra Ethernet nor MRC are generally available yet. This table reflects our understanding of vendor intentions and should be confirmed.

| Vendor | NIC | Capacity | Max Eth Speed | Ultra Ethernet (future) | MRC (future) | Multi-plane (hw/fw support)[1] |
|---|---|---|---|---|---|---|
| AMD | Pollara | 400G | 400G | Yes | Yes | Yes |
| AMD | Vulcano | 800G | 800G | Yes | Yes | Yes |
| Broadcom | Thor 2 | 400G | 400G | No | No | No |
| Broadcom | Thor Ultra | 800G | 800G | Yes | Yes | Yes |
| NVIDIA | ConnectX-7 | 400G | 400G | No | No | No |
| NVIDIA | ConnectX-8 | 800G | 2x400G | No | Yes? | No |
| NVIDIA | ConnectX-9 | 800G | 800G | No | Yes? | Yes |

**Table 1:** Common AI Scale Out NICs

[1] *All NICs that support multiple interfaces can support some degree of multi-plane operation through host and CCL configuration. This column distinguishes NICs that have some level of hardware or firmware support for load balancing and failure detection when operating across multiple planes.*

## Building AI Networks with Arista

Arista is a leader in AI Networking, across all types of AI Network. This section briefly presents Arista's solutions and the advantages of building and operating AI networks with Arista. For more depth, refer to https://www.arista.com/en/solutions/ai-networking and our AI Networking white paper.

### Etherlink AI Portfolio

Arista's Etherlink portfolio offers ultra high-performance standards-based Ethernet systems with smart features for all types of AI Network. Smart AI-Aware features include a portfolio of specialist load balancing and congestion control features, including Arista's RDMA Aware QoS and load balancing capabilities which ensure reliable packet delivery. Etherlink features are supported across a broad range of 800G and 1.6T systems. Etherlink is forwards compatible with both Ultra Ethernet and Multipath Reliable Connection (MRC).

Arista offers fixed, modular, and distributed network platforms for AI, scaling from small clusters to topologies supporting over 100,000 accelerators. The 7060X series provides 400G, 800G, and 1.6T systems up to 102.4Tbps. The 7800R4 series delivers up to 460Tbps with 576x800G or 1152x400G ports, enabling large, efficient single-switch clusters or serving as the AI spine or leaf in large 2-tier networks. Additionally, the 7800 supports internet-scale routing and the key encryption, traffic segmentation and traffic engineering features that are required for Front End and Scale Across networks. Finally, the 7700R4 Distributed Etherlink Switch (DES) scales the R4 architecture to support over 9,000 400GbE accelerators in a distributed, single-hop system, ensuring deterministic, fair, 100% efficient lossless forwarding — the ideal AI network fabric.

### EOS as a foundation for AI

Arista Extensible Operating System (EOS) serves as the foundation for next-generation data centers and high-performance AI networks, offering industry-leading reliability and the lossless environment essential for accelerated computing. Built on a unique multi-process state-sharing architecture, EOS separates network state from protocol processing to ensure maximum uptime and system stability. It offers extensive programmability through a rich set of APIs and SDKs. Key features include real-time telemetry for network-wide visibility, zero-touch provisioning, and support for containerized or virtualized deployments, all aimed at reducing operational costs and increasing business agility.

By utilizing a single binary software image across all Arista platforms—including AI-optimized hardware like the 7800R4, 7060X6, and 7060XE7 series—EOS ensures operational consistency and simplifies the deployment of complex AI fabrics. Its programmable nature and support for advanced security, segmentation, and policy capabilities allow organizations to automate workflows and scale their AI compute resources while maintaining rigorous performance and safety standards.

### AI Operations

Arista's platforms with EOS provide extensive telemetry capabilities for AI networking. These capabilities combine with Arista CloudVision for AI Networks to create a data-driven networking architecture where EOS provides the telemetry and CloudVision provides the intelligence and centralized management. CloudVision for AI Networks adds workload job visibility and analysis to the normal network-oriented views. Server CPU and NIC metrics can be integrated through Arista AI Agent, providing deep visibility into AI workload operation to improve cluster deployment time and operational stability. And Arista's Autonomous Virtual Assistant (AVA) provides AI-driven support to operators.

These solutions can be flexibly combined with two open-source automation frameworks to deliver a complete solution for end-to-end provisioning, deployment, and operations. Arista AVD (Architect Validate Deploy) provides a collection of tools and best practices for deploying and managing Arista networks using infrastructure-as-code principles. Complementing AVD, the Arista Network Test Automation (ANTA) framework allows on-demand validation of the network state against intended configurations, ensuring the network operates reliably and as expected. Arista solutions provide open APIs for customers who wish to integrate their own systems.

### Smart System Upgrade (SSU)

Arista Smart System Upgrades are designed to address one the most challenging operational tasks in AI fabrics: infrastructure maintenance and device upgrades. Whether to deploy new features, install new hardware, or to mitigate security vulnerabilities, maintenance that causes network downtime can be hugely impactful to job completion times and overall efficiency, impacting the ROI of AI clusters. SSU is the solution to that problem, building on the robust multi-process state-sharing architecture of EOS. Combining protocol-based graceful shutdown, insertion, and unique redirection techniques, traffic can be diverted around the devices under maintenance. And in the critical leaf layers facing the XPUs, packet forwarding simply continues during device upgrades with workloads unaffected. Operations teams and network administrators now have the ability to perform seamless maintenance to any infrastructure element. Through CloudVision, organizations can fully automate these upgrade operations. To learn more, refer to our SSU White Paper.

### BGP - Maintenance Mode

Introduced in Arista's EOS 4.15.2F, Maintenance Mode is a method to allow for easy maintenance of a switch or specific elements of a switch. Arista's Maintenance Mode provides a set of commands with a wide range of flexibility that make our network operations lives a bit simpler. With Maintenance Mode we support the removal and reintroduction of either a whole switch, portions of the switch or even specific BGP neighbors in a graceful operation that minimizes network disruption. The feature leverages industry standards including the original Graceful Shutdown (RFC 1997) and the updated Graceful BGP Session Shutdown (RFC 8326). Please refer to the Arista TOI and Community Central article for further details.

## Scale Out Network Design

### General Design Principles

Our detailed design and deployment guides explore these topics more deeply, but it is worth outlining some high-level design principles for Scale Out networks as context for some of the discussion that follows.

- **Symmetry is critical:** Unlike a generic datacenter network, symmetry is strongly desired in AI networks. This is to ensure that network performance is homogeneous and that workload placement need only consider simple locality rules for optimal performance. Symmetry is one of the primary features of leaf-spine architectures and one of the reasons they are common in AI Networks. In pursuit of symmetry, hosts are typically evenly distributed across leaf switches and leaf->spine connectivity is also homogeneous.
- **Scale Out Networks are 1:1 or undersubscribed:** As mentioned early, AI workloads generate large symmetric bursts of line-rate traffic. This traffic puts huge pressure on the network and, as a result, AI Scale Out Networks are almost never over-subscribed. In fact, some deployments are under-subscribed (i.e., over provisioned). This may be done to ensure the network doesn't congest even with load imbalances or the presence of some failures in the network.
- **Spine scaling is typically planned up front:** In Scale Out networks that have a spine layer (discussed in more depth in the next section), it is important to consider the maximum intended scale. This is for two reasons. 1) Scaling the spine by adding devices is highly intrusive and can involve substantial recabling and disruption to the existing network. 2) The need for symmetry requires consistent leaf-spine connectivity and this is difficult to maintain during scaling operations. As a result, the strong advice is to build the spine for the maximum intended scale of the network, even if servers and leaf switches are being deployed incrementally.

### Topologies

#### Tiers and Fat Tree Networks

A key network design consideration is the number of network tiers that are required to meet a given network demand. The need for additional tiers to maintain connectivity as a network scales is a fundamental driver of non-linear costs in large-scale networks, as the complexity and overheads climb with the number of tiers.

In the simplest case, only a single tier of network devices is required. In this case, all connectivity is directly between network clients and network devices. There may be a single device or there may be a set of independent devices that all clients connect to in parallel. But, in both cases, there is no internal network connectivity and all ports serve network clients. Provided the network devices used are internally efficient and lossless, single tier networks have no need to consider load balancing, flow placement, or congestion management within the network. The simplicity also means that they do not require "smart" NICs. For some use cases, a single tier can meet the entire network demand, particularly when you consider modular systems.

Given finite limits on single device size, a tiered network is often required. Scaling beyond a single tier introduces inefficiency as additional network devices and internal links are required to provide connectivity between the devices that face the external workload or clients. The scaling limits for a specific network topology typically depend on the device radix. Radix is a graph theory term that refers to the fan out from a network device to other devices within the network. In the simplest case, the radix matches the port count. But the network radix can also be higher than the port count (where ports are broken out into separate lower-speed interfaces) or lower than the port count (where parallel connections are used between devices).

There are many ways to build a tiered network. A common tiered network design is a fat-tree (or Clos) network, where a set of leaf devices provide external connectivity and are connected together by a set of spine devices to form a single large network. Fat-tree networks are popular for AI networks as they scale to large capacity demands and provide a high degree of internal symmetry and homogeneity.

Fat-tree networks can be built with different degrees of internal over- or under-subscription. In this section, we focus on uniform non-blocking (1:1) fat-tree networks as these are common in Scare Out networks and provide full non-oversubscribed bandwidth

between all ports. But fat-tree networks can be designed to deliver over-subscription, where the bisectional bandwidth across the network is less than the port capacity. Or they can be designed to be under-subscribed where there is more internal capacity than client port capacity, which can protect network performance in the presence of failures or load imbalances.

For a device of radix *k*, a two-tier uniform fat-tree requires $k+k/2$ devices (with $3k^2/2$ total ports) to deliver $k^2/2$ ports of external capacity. This means that a two-tier network introduces 200% overhead compared to a single tier. For example, with a radix of 64, a two-tier network requires 96 devices (with 6,144 ports) to deliver 2,048 externally facing ports. The remaining ports are consumed by 2,048 internal links between devices in the fabric (consuming 4,096 total ports).

Efficiency decreases further in a three-tier fat-tree network. The number of ports that can be delivered in a three-tier fat-tree network increases enormously to $k^3/4$, but the overhead increases to 400% because you now need $5k^2/4$ devices with $5k^3/4$ total ports to build the network.

**Figure 2** below shows the maximum size one-, two-, and three-tier networks that can be built with toy devices of radix 4. This illustrates the scaling inefficiency as you add tiers. Two- and three-tier networks designed with higher radix devices scale very rapidly to much larger port counts (since port count is proportional to either the square or the cube of the radix). The overheads remain the same as you scale up.

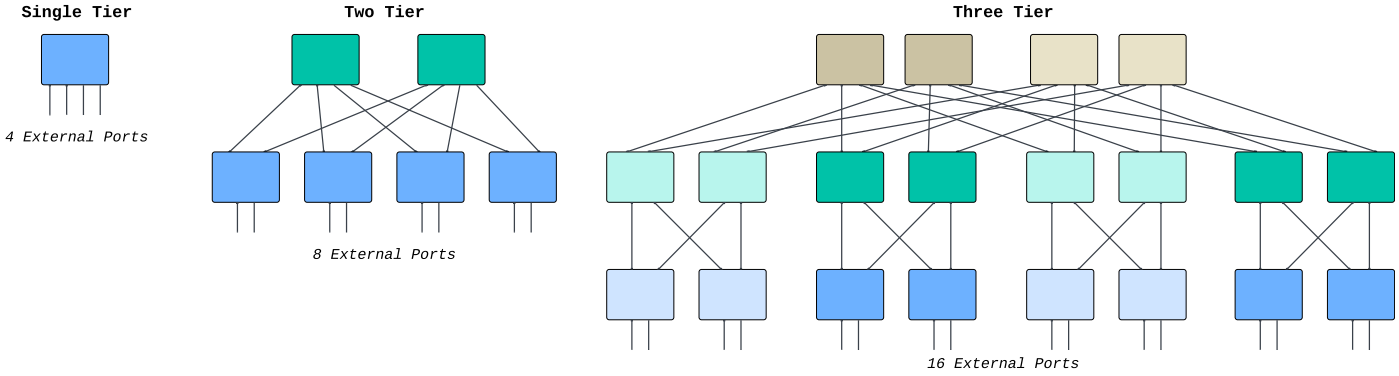


*Figure 2: Fat tree (Clos) network scaling (switch radix 4)*

The increasing inefficiency of networks as the tier count increases represents an opportunity for large port-count solutions, such as the 7800. Single-tier designs scale to larger demands as we mentioned already, but additionally, two-tier designs can be scaled to meet far larger demands. Because port count in a two-tier network scales with the square of the device radix, we can leverage the higher radix of a modular system in either one or both tiers to scale to much larger sizes.

## Rails and Planes

Rails and Planes are two separate architectural concepts that come up frequently in Scale Out network designs. The choice of Rail-aligned designs for enhanced performance is very common. The use of multiple Planes for scaling efficiency or redundancy is increasingly common as system support for multi-plane operation is becoming more widely available. It is important to understand both concepts.

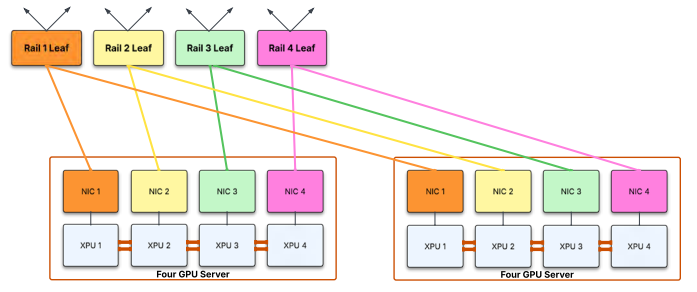


*Figure 3: Rails*

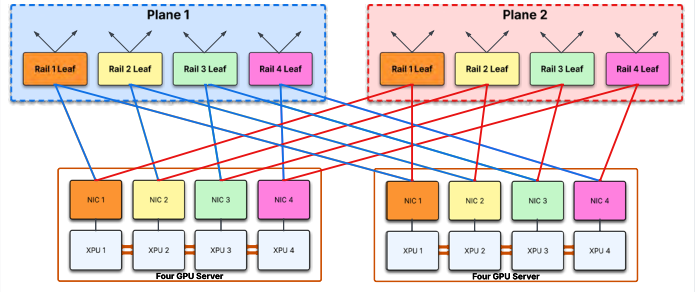


*Figure 4: Planes and Rails*

A Rail groups subsets of XPUs on AI servers into affinity groups based on the local numbering of XPUs (known also as the "local rank"). For example, XPU1 on every system might connect to Rail1, XPU2 to Rail2, and so on, with a Rail for each XPU in the server (as in **Figure 3**). Or, there may be a Rail for every two XPUs (XPU1/2 -> Rail1, and so on). XPUs on each Rail are then connected to separate network devices. This is referred to as a Rail-aligned (or Rail-optimized) design. The reason for deploying a Rail-aligned design is to improve workload performance: this topology increases the number of servers that are connected to the same set of first tier network devices and therefore the number of servers that are one hop away from each other.

When a Rail-aligned network design is deployed, collective communication libraries can then leverage this design by supporting Rail-aligned communication. In this model, communication patterns are optimized to benefit from the Rail-aligned connectivity. This is typically done by having servers only communicate on the Scale Out network between XPUs on the same Rail. The higher-capacity Scale Up network within the server is used for communication between XPUs on different Rails and to facilitate communication between XPUs in different servers that are on different Rails. This approach is efficient for many collective communication patterns.

In a Rail-aligned design, the Rails are typically connected together at some higher tier in the network. However, some network designs call for disjoint Rails with no such connectivity. This is referred to as either a Rail-disjoint or Rail-only design. While collective communication libraries (CCLs) may generate no Scale Out network traffic between Rails in normal operation, it is important to note that other types of traffic and CCL or NIC behaviors in a failure may lead to a requirement for connectivity between Rails. As a result, the decision to deploy a Rail-disjoint network needs to be considered carefully, taking both CCL behavior and workload requirements into account.

A Plane is a separate concept in a Scale Out architecture. (There is sometimes confusion between Rails and Planes, partly due to prior use of the term plane in the context of Rails.) A Plane is a parallel, completely independent network that connects to all XPUs. In a multi-plane design, parallel independent planes are built and each XPU connects to all of the planes in parallel, as illustrated in **Figure 4**. The connectivity from each XPU may be achieved through multiple NICs or by attaching a single NIC to multiple planes, or both. In all cases, the planes are identical. Multi-plane solutions offer an immediate benefit in redundancy, as the system can survive many types of failure and the individual network planes are completely isolated from each other. There is no connectivity between the planes, so the NIC or host must manage both load balancing between planes and detection and handling of connectivity failures within planes.

Both Rails and Planes can (and often should) be deployed together. The ideal network design, given a server architecture that supports it, can be both multi-plane and multi-rail. But this is never required and the concepts can be implemented independently. **Figure 4** above illustrates the connectivity for a network with four Rails and two Planes. Each Plane is identical and all NICs connect to both the blue and red planes. There are four Rails, one for each XPU in each server. As a result, every NIC in each server connects to a different Rail-aligned leaf switch. This pattern is then replicated across the two Planes since they are required to be identical.

### Multi-planar designs

Dual-plane designs are common today, particularly for NVIDIA deployments that use the ConnectX-8 NIC, which is natively 2x400G. Dual plane designs do not typically require explicit NIC support as the AI software stack can detect and operate efficiently on a dual-plane network without hardware support. However, overheads climb and efficiency falls with larger plane count unless you have hardware support.

As mentioned above, solutions such as MRC and UET that natively support multi-planar Scale Out networks are emerging and these solutions efficiently support as many as 8 planes per NIC. These solutions are particularly attractive because the maximum scale of a two-tier network increases with the square of the radix of the network devices used. This means that the same network device can support much larger scales if the required port speed is reduced, leading to significant cost and scale benefits. As an example, a 64 x 1.6Tbps switch such as the 7060XE7 can support 2,048 x 1.6Tbps clients in a two-tier architecture. But the maximum scale increases rapidly as the port speed is reduced: 8,192 x 800G, 32,768 x 400G, or 131,072 x 200G. (Even larger sizes are possible with modular AI Spines.) **Figure 5** below illustrates the high level approach.

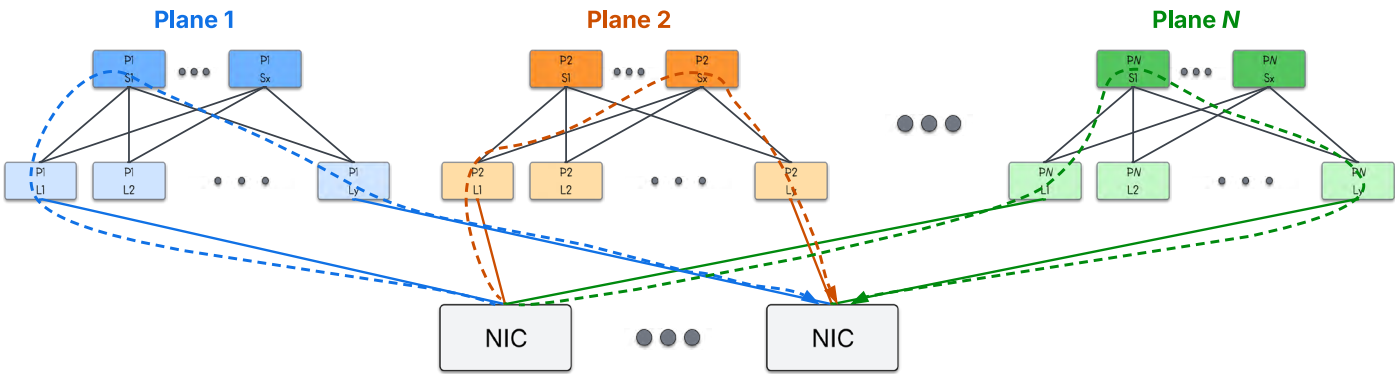


*Figure 5: Multiple two-tier planes*

**Network sizing**

An AI Scale Out network is typically planned out in advance for the maximum scale that is required. This doesn't mean that it is all installed in a single deployment, but key aspects of the design, such as the number and type of spine devices will be planned based on the intended final scale. Upgrading a Scale Out network isn't impossible but it can be challenging and may require downtime or a large number of incremental steps to avoid disrupting workloads.

As a result, it is important to establish the intended maximum scale before beginning design work. Understanding any planned phasing of deployments is also important as that can influence the detailed design. The outcome of this discussion should be an understanding of the maximum planned number of XPUs, the designed capacity per XPU, whether Rail alignment is desirable, and the number of Planes that are planned. Also, if the deployment will be phased, it is useful to understand the increments.

For customers that are focused exclusively on inference, it is also important to understand whether a full Scale Out network connecting all XPUs is actually required. Individual inference model workloads require far fewer XPUs than training[2]. Many inference models can be executed either on a single XPU or on the set of XPUs within a single Scale Up domain (server or rack): these deployments generate no Scale Out traffic. And even when Scale Out connectivity is required (for the largest models, largest context sizes, and highest request rates), models execute on sets of 10-100s of XPUs rather than 1000s. As a result, a partitioned Scale Out network that provides pods of XPUs may be sufficient, or the Scale Out network may be oversubscribed based on the proportion of jobs that require Scale Out connectivity.

### Overprovisioning (aka under-subscription)

In some cases, it doesn't make sense to use all the available XPU-facing ports on each leaf switch. This may be due to datacenter layout considerations or it may relate to rack-scale XPU deployments where the number of Scale Out NICs in a rack does not evenly align with the number of XPU-facing ports in a leaf switch. When this happens, you can choose either to reduce the spine capacity to match (by reducing the number of spine devices). Or you can choose to overprovision so the network is 1:1 even during failures.

The most common scenario where this arises is with rack-scale XPU deployments, where the number of NIC ports may be a multiple of 72 rather than 64. In a Rail-optimized deployment, this leads to multiples of 18 XPU-facing downlinks per rack. This may lead to a configuration with, for example, a requirement for 54x400G downlinks on each leaf switch. This leaves you with a choice on what to do on the uplink side. You can choose to provision 1:1 and have 54x400G or 27x800G as uplink to the spine. Or you can choose to provision additional uplink capacity (and matching spine devices) to deliver a degree of overprovisioning. This overprovisioning provides additional capacity that provides resilience against link and switch failures as well as reducing the risk of congestion under very high loads.

[2]*Communication between models, such as in an agentic workflow, typically uses the Front End network.*

**General Scale Out Architecture Guidance**

This section contains some rules of thumb to help in selecting a specific Scale Out architecture to use to meet the requirements for a specific deployment. As with all rules of thumb, some judgement is required and a different architecture may better fit a particular set of requirements. Indeed, we are proud at Arista to be able to offer a number of different solutions to best match customer requirements rather than have a single one-size-fits-all approach. We do have detailed reference architecture guidance that cover common scenarios, such as two-tier Scale Out deployments of our 7060X6 platform. But we also frequently work directly with customers to produce a design customized to their requirements.

### Single-Tier Architectures

For deployment scales that have a per-plane capacity requirement that is less than or equal to the capacity of a 7800 modular chassis , we recommend a **Single-Tier Architecture**. This is by far the simplest design to deploy and operate. Depending on the desired scale, single-tier deployments of either 7060 or 7800 are possible and a single device is deployed per plane.

With current platforms, we can support up to 64x1.6G, 128x800G, 256x400G, 512x200G, or 512x100G per plane with 7060 and up to 576x800G, 1,152x400G, 2,304x200G, or 4,608x100G per plane on 7800. For example, a dual-plane 7800R4 solution can support up to 1,152 XPUs with 2x400G per XPU.

### Single-Hop Distributed Etherlink Switch (DES)

For deployment scales above the scale supported in a single-tier, Distributed Etherlink Switch (DES) offers a distributed single-hop routing solution that retains many of the advantages of a Single-Tier modular chassis deployment. The DES system provides a resilient, deterministic, fair, and 100% efficient lossless forwarding fabric. This architecture is a particularly strong fit for customers who do not have significant in-house network engineering and automation capabilities and are looking to partner with Arista to deliver a complete solution.

The current DES7700 solution can support up to 4,608x800G, 9,216x400G, 18,432x200G, or 36,864x100G per plane.

### Two-Tier Fixed Form Factor

The next step in terms of capacity requires the deployment of a two-tier architecture. For medium scale needs, this might be based on two tiers of fixed form factor devices. Two-tier designs leveraging 7060X6 can scale to 2,048x800G, 8,192x400G, 32,768x200G, or 131,072x100G per plane. With 7060XE7, these numbers increase to 2,048x1.6T, 8,192x800G, 32,768x400G, and 131,072x200G.

### Two-Tier Modular Chassis

For larger scales or increased flexibility in terms of incremental deployment and feature sets, the final set of reference architectures combine 7800 Spines with 7060 leaf devices. With our current platforms, these designs scale to meet very large Scale Out network demands: 18K x 800G, 73K x 400G, or 294K x 200G. These numbers double with the next generation of platforms on the roadmap.

**Note:** Arista does not currently recommend three-tier designs for Scale Out deployments and doesn't have a three-tier reference architecture. While three-tier designs can, of course, be built with Arista platforms, a three-tier design introduces additional operational complexity and doubles the internal overhead in the network. As a result, scaling up to meet the demand with two-tier designs and modular chassis spines has lower TCO as well as lower operational overhead. The increasing availability of AI systems that support multi-planar designs also reduces the requirement for three-tier architectures.

## References

- AI Networking white paper
- Three Genius Ideas for AI Fabrics
- Multiplanar Scale Out fabrics with MRC
- Demystifying Ultra Ethernet

**Santa Clara—Corporate Headquarters**
5453 Great America Parkway,
Santa Clara, CA 95054

Phone: +1-408-547-5500
Fax: +1-408-538-8920
Email: info@arista.com

**Ireland—International Headquarters**
3130 Atlantic Avenue
Westpark Business Campus
Shannon, Co. Clare
Ireland

**Vancouver—R&D Office**
9200 Glenlyon Pkwy, Unit 300
Burnaby, British Columbia
Canada V5J 5J8

**San Francisco—R&D and Sales Office 1390**
Market Street, Suite 800
San Francisco, CA 94102

**India—R&D Office**
Global Tech Park, Tower A, 11th Floor
Marathahalli Outer Ring Road
Devarabeesanahalli Village, Varthur Hobli
Bangalore, India 560103

**Singapore—APAC Administrative Office**
9 Temasek Boulevard
#29-01, Suntec Tower Two
Singapore 038989

**Nashua—R&D Office**
10 Tara Boulevard
Nashua, NH 03062