White Paper
April 2025

IT@Intel: Scaling Intel's Data Centers with Software-Defined Networking and Automation

Intel IT has chosen an open, standardized approach to software-defined networking

Intel IT Authors

Sanjay Rungta
Senior Principal Engineer, Network Architecture

Greg Botts
Infrastructure and DevOps Engineer

Matthew Gray
Network Automation Engineer

Seth Gehring
Network Automation Engineer

Mohammad Ali
Infrastructure and DevOps Engineer

Table of Contents

Background 2
Selecting an SDN Approach and Architecture Components 2
Improving Scalability by Adopting a Leaf-Spine Network Architecture . . . 2
Strategy for a Scalable, Robust SDN Architecture 3
Solution Architecture 5
Migration Strategy 10
Results 11
Next Steps 12
Conclusion 12
Related Content 12

Executive Summary

As Intel's business expands, the demand for data center network capacity has surged by over 25% annually. Furthermore, business pressures necessitate that new capacity be operational within 24 hours. As early as 2014, we acknowledged the potential of software-defined networking (SDN) to help address these challenges.

After evaluating the components and architectures of SDN, we opted for an open, standards-based architecture rather than a supplier-centric solution. As our SDN architecture has evolved, we have established a standardized and scalable data center network architecture that extensively uses automation. The open interface provides the flexibility to integrate additional business-driven automation, enabling us to meet our growth objective and timeline requirements.

Our network architecture strategy relies on five pillars:

- **Scalability through standardization.** We maintain compliancy with our standards for hardware, OS, device roles, topology, configurations, and solutions to enable automation and rapid scalability at large data centers.[1]
- **Programmability.** This allows our workforce to adapt to significant growth in network scale with improved speed. It also facilitates full lifecycle provisioning of network infrastructure from Day 0 to end of life.
- **Security.** We have the capability to segment the network using a common infrastructure to support various use cases and enhance data center security.
- **Resiliency.** Built-in network resiliency helps ensure the continuous operation of network functionality, facilitates rapid recovery, and maintains performance even in compromised conditions.
- **Supportability.** We strive to be sure the network maintains its designed level of performance and availability of the network. Adhering to standards facilitates troubleshooting.

Over the past five years, we have migrated the majority of our data centers to a new SDN architecture that employs a leaf-spine underlay combined with overlay networks. By utilizing Industry-standard components and protocols, we have significantly improved network delivery times while reducing the need for human resources, thereby enhancing overall efficiency. Additionally, we have increased the stability and reliability of the network and consolidated multiple dedicated customer networks onto a common infrastructure that's integrated with enhanced security controls.

[1] Early in our SDN adoption, we relied on naming conventions to define attributes of a device. In our new system, we no longer rely on naming conventions, but rather store those attributes in our Network Source of Truth (NSoT).

Acronyms

ACL	access control list
ASN	autonomous system number
BGP	Border Gateway Protocol
CMDB	Configuration Management Database
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DOME	Design, Office, Manufacturing and Enterprise
eBGP	External Border Gateway Protocol
EVPN	Ethernet Virtual Private Network
HPC	high-performance computing
LACP	Link Aggregation Control Protocol
MLAG	Multi-Chassis Link Aggregation Grouping
NSoT	Network Source of Truth
POD	point of delivery
SDN	software-defined networking
SMF	single-mode fiber
STP	Spanning Tree Protocol
TOR	top-of-rack
VTEP	Virtual Tunnel End Point
VRF	virtual routing and forwarding
VNI	VxLAN Network ID
VxLAN	Virtual Extensible Local Area Network
WSGI	Web Server Gateway Interface
ZTP	zero-touch provisioning

Background

Intel's data centers[2] are the heart of a thriving, complex business. Intel IT operates 55 data center modules at 15 data center sites. These sites have a total capacity of 126 MW, housing more than 418,000 servers that underpin the computing needs of approximately 100,000 employees. To support the business needs of Intel's critical business functions—Design, Office, Manufacturing, and Enterprise (DOME)—while operating our data centers as efficiently as possible, Intel IT has engaged in data center network modernization for many years. Intel's business is becoming increasingly data-driven, relying on machine learning, AI, big data analytics, and automation. As data explodes, we are experiencing greater than 25% growth in demand for network capacity every year. In parallel, we desire to put the new capacity into production within 24 hours once received to optimize the value of the investment.

In 2014, we began evaluating software-defined networking (SDN) solutions as a way to meet these data center challenges. Until that time, traditional networking approaches using fixed-purpose hardware met the needs of client/server computing. But, with the proliferation of cloud-based services and server virtualization, along with continued business growth, we needed a way to keep up with a more dynamic computing environment, and SDN offered a lot of potential.

[2] IT@Intel, "Data Center Strategy Leading Intel's Business Transformation," https://www.intel.com/content/www/us/en/it-management/intel-it-best-practices/data-center-strategy-paper.html.

Our SDN solution provides us with an interface that enables programmatic manageability. It also offers an integrated and automated control plane, which allows us to scale while maintaining a standardized environment. We now use our data center SDN architecture in three of the four DOME environments; however, the Manufacturing environment uses a different approach due to its unique business drivers.

Selecting an SDN Approach and Architecture Components

In 2018, as we started exploring how to adopt 100 Gbps technology, we scanned the industry and SDN solutions. As the SDN market evolved, we noted that solutions tended to fall into two categories:

- Closed-loop SDN using supplier-centric technologies.
- Open, standards-based SDN that supports next-generation data center architectures featuring underlay and overlay designs.

While each approach has its advantages, we determined that developing standardized, scalable building blocks for our data center network architecture would better support the automation necessary for on-demand provisioning, self-healing, and scalability. The open architecture enables us to integrate additional, business-driven automation capabilities to meet our specific requirements. Plus, it helps avoid vendor lock-in and takes advantage of a growing, evolving ecosystem.

Once we settled on an overall SDN approach and a switch vendor, we started large-scale migration in 2019, and we have migrated over 90% of Intel's data centers to SDN technology over the last five years. During this time, we introduced new technologies and uplifted our orchestration platform for scale with container-based infrastructure and an open-source Network Source of Truth (NSoT) and network automation platform.

Improving Scalability by Adopting a Leaf-Spine Network Architecture

Traditionally, Intel's data center network architecture was implemented with a three-tier hierarchical model. This industry-standard method of connectivity consisted of Core, Distribution, and Access layer switches. Using L3 protocols for routing between the Core and Distribution layer switches and L2 protocols between the Distribution layer and the Access layer switches enabled simple, intuitive deployment of services that helped increase the supportability of our critical data centers. However, this architecture could not scale well enough to support Intel's growth needs within its massive Design centers that use high-performance computing (HPC); nor could it support the growing complexity within the Enterprise data center environments. In addition, our Design and Enterprise data center network traffic experienced a significant shift from primarily north-south traffic to mostly east-west traffic. This shift caused congestion on the Core and Distribution layers. To better support the new traffic patterns and Intel's growth, we are continuing to modernize our network architecture.

We are replacing the three-tier hierarchical model with a leaf-spine architecture. (See "Migration Strategy" later for our approach to transparently transitioning the network from one model to the other.)

In a leaf-spine architecture, the leaf switch is connected to multiple spine switches. This approach indirectly provides higher bandwidth and improved redundancy. By adopting a scalable unit of leaf and spine (also called a point of delivery, or POD), it is easy to scale the data center network using fixed-configuration switches on an as-needed basis (Figure 1).

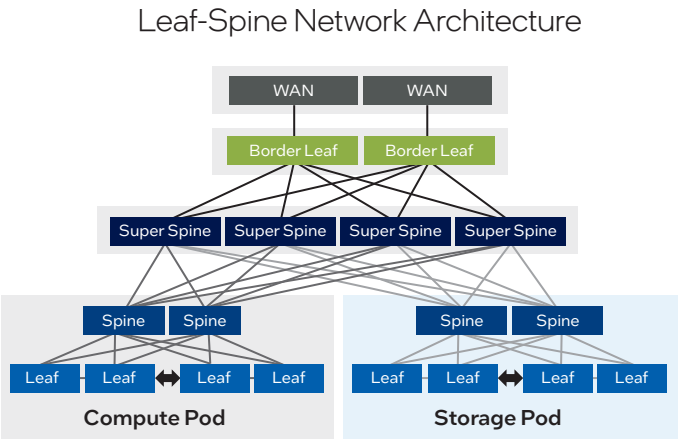

Figure 1. A leaf-spine network architecture better supports Intel's data centers, compared to a traditional three-tiered hierarchical network architecture.

Network Fabric Design Details

To make the adoption and scaling of a leaf-spine architecture most efficient, we require every aspect of the architecture to have repeatable building blocks (such as PODs), deterministic communication flow, and solution flexibility to meet a growing number of use cases required by Intel's business units. This building-blocks approach enables large-scale deployment at an increased deployment velocity. Our network architecture is also built with strict standards and guidelines that encompass the full stack of our network.

To the fullest extent possible, we automate a switch's lifecycle from onboarding to end of life. This lifecycle automation enables transparent deployment and maintenance:

- Day 0 zero-touch provisioning (ZTP) and onboarding.
- Day 1 configuration for fabric deployment.
- Day 2 configuration for a specific use case.
- End of life removal or decommissioning of the switch from the network.

For the underlay network, the leaf-spine design and Border Gateway Protocol (BGP) routing are critical aspects. To provide L2 mobility across the fabric and highly secure, transparent enclaves, overlay networks are built using Virtual Extensible Local Area Network (VxLAN) and BGP's Ethernet Virtual Private Network (EVPN) capabilities. To optimize the network for non-blocking network communications, we have eliminated the Spanning Tree Protocol (STP) from the network. Also, all network switches are non-blocking-capable devices; this means that the switch can carry ingress/egress network traffic at wire speed (the maximum bandwidth of the interface).

Strategy for a Scalable, Robust SDN Architecture

Historically, our network strategy has been optimized predominantly for cost, although we also considered network performance. To better support Intel's growing business, we have redefined our network strategy to pursue technological advances to modernize and transform the network to help ensure not only cost-effectiveness but also best-in-class service quality. The following sections provide some details about the five pillars that underpin our data center network strategy.

Scalability through Standardization

When we set our initial goals for SDN, we realized the solution we developed needed to be automatable and scalable both locally and globally. This necessitated a well-defined set of conventions that covered both local configuration parameters and those that would potentially have global relevance. This was an early, critical acknowledgment. To that end, everything was designed with standardization in mind. We also constructed the documentation so it could be interpreted by developers. We embedded all configuration specifications in our architecture guide to encourage and enable automation. The documentation includes variables, input parameters, and configuration outputs.

Some of the critical conventions that we defined include the following:

- **Device naming.** We originally implemented device naming so that the name indicates a device's location information and function. From the naming, we could derive configurations that are location-specific (such as local Domain Name System [DNS] and directory services) and function-specific (such as spine versus leaf configurations). The adoption of this naming convention enabled us to later programmatically migrate our fleet into our new NSoT, where we no longer need to rely on encoding the attributes of a device into the hostname, but rather store those attributes in our structured NSoT.
- **VLAN definitions and parameters.** We identified VLAN use cases and configuration parameters. Each VLAN is assigned to a security zone and carries certain attributes within the zone. Much of the automation configuration is based on this information. Over time, we have found the VLAN definition to be the most dynamic network aspect, as we continually add new use cases. Our VLAN construct has been invaluable in maintaining structure within the fabric as we manage new deployments.
- **BGP autonomous system number (ASN) allocation.** We allocated ranges based on location and within each data center function (Design or Office/Enterprise). Similar to other conventions, this allows for predictable, automatable deployments.

- **Connectivity assignments.** We pre-allocated which ports would be assigned based on device and functionality. Depending on placement within the fabric, device types were assigned along with the connectivity conventions to neighboring elements.
- **Device types and OS.** We used a limited set of certified devices in the solution to simplify spare parts inventory and device support. New devices are only added as critically needed. Sometimes, this forces us to use devices that aren't a perfect fit, but the need for consistency outweighs the use of one-off device types. We minimized the device type list to help reduce the OS count and specifications that we must test against. When we certify a new OS, we push the upgrade across the install base, which helps ensure that all features are available and perform as expected. Our approach to device types and OS use allows us to design without having to account for deployment inconsistencies.
- **Base-build configurations and security specifications.** We identified and propagated common configurations that incorporate security across all devices to help ensure stability and a common configuration to build upon.
- **Security zones.** Each identified security zone receives the appropriate and relevant conventions.
- **VxLAN Network ID (VNI) allocations.** We globalized VNI mappings with ranges pre-assigned to each data center, so when we implemented Data Center Interconnect (DCI), there were no VxLAN VNI conflicts. We could use legacy VLAN information within each data center without worrying about VLAN conflicts in other locations.

Together, these conventions enable a highly automatable and scalable deployment as well as a significant reduction in mean time to deploy (MTD) and mean time to repair (MTR).

Programmability

Our previous network solution had limited automation capabilities. Onboarding network devices required physical touch; could be accessed only by older methods like SSH and command-line interfaces (CLIs); and had to be configured and managed individually, mostly with human intervention.

With our new SDN solution, the devices register themselves with our NSoT and configure themselves with a base build, all from a small, distributed set of scripts. This enables us to enforce standardization and change control and have a single source of truth for our network environment.

Once we could efficiently manage our fleet of devices, we programmatically generated all the relevant configurations for them. Having the Standardization components already defined algorithmically provided us with configuration templates and the variables that would be used per site and per device. It also provided us with the algorithms for computing the values of those variables. We created Python code to compute the values and pass them into Jinja2 templates, rendering a device configuration that is complete with its specific values. See the Orchestration and Automation Framework section later for additional details.

The combination of standardization and programmability gives us consistency across the network environment, drastically reducing human error and downtime while allowing us to quickly deploy new network capabilities.

Security

A critical aspect of the new design was to enable multi-tenant support with appropriate security at all layers over the common underlay data center IP fabric. We enabled multiple security capabilities—such as large-scale access control list (ACL), virtual routing and forwarding (VRF), traffic redirect, etc.—in the toolbox so that the right tool can be used at the underlay or overlay layer to control the traffic flow. Integration with external security capabilities like a firewall was also essential. It was also critical for the security solution to scale beyond 10 Gbps performance with next-generation firewalls. Finally, we used sFlow processing in the design to keep the visibility in the environment.

Resiliency

Our goal is to enable continuous operations of network functionality even in the face of network failures and rapid recovery. Here are some of the ways we are increasing network resiliency:

Expanding the routing domain to create an equal-cost/multiple-path design. We are using External BGP (eBGP) to significantly scale data centers at locations with multiple availability zones (see Figure 2).

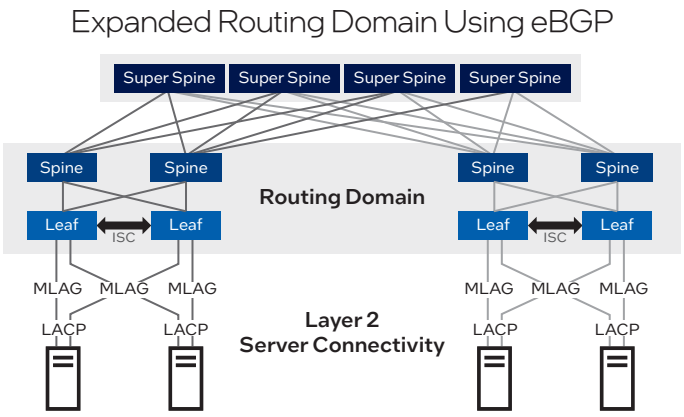

Figure 2. We are using eBGP to expand the routing domain, while concurrently reducing L2 connectivity to improve network resiliency.

- Reducing L2 outage domains within racks. This includes eliminating the STP to improve data throughput and using a /31 subnet mask to conserve IP address space for point-to-point links. The latter technique eliminates a port channel between the leaf and spine, which in turn, eliminates the possibility of uneven load balancing (hash polarizations).
- Deploying dual home servers to increase server uptime and enabling the network team to perform maintenance without affecting customers.

- Using the standards-based Link Aggregation Control Protocol (LACP) within IEEE 802.3ad to allow the logical bundling of links while negotiating with far-end devices to enable graceful removal of links that are not transmitting the LACP. This approach reduces cabling issues and link faults.
- Employing Multi-Chassis Link Aggregation Grouping (MLAG) to deliver system-level redundancy to servers. MLAG logically teams two switches to appear as one logical switch from the server's perspective.
- Collocating critical services such as DNS and Network Time Protocol with servers. We deployed a DNS solution in our HPC data centers to help ensure that WAN outages would not impact local data center functionality. Without communication to a DNS, all servers and services fail within the data center.
- Implementing a zero-congestion strategy. Network traffic congestion is difficult to correct quickly. Our network designs include downlink-to-uplink bandwidth ratios to avoid congestion on links.

Supportability

The other four pillars — standardization, programmability, security, and resiliency — combine to provide us with the ability to maintain the designed level of performance and availability of the network. Our use of standardization leads to reproducible configurations and designs and reduces or eliminates non-compliance and difficult-to-support one-off designs. This, in turn, leads to repeatable and standards-compliant predictive troubleshooting. The result is a modern, highly automated, and resilient SDN that powers Intel's digital transformation through seamless secure connectivity.

Solution Architecture

The following sections detail some of the high-level features of our SDN architecture.

Physical Layer

There are three key vectors used to design large-scale data centers:

- Ethernet switch
- Data center topology
- Physical media selection

Industry-wide discussions with fellow travelers have helped us discover that networks inside hyper-scale data centers are converging to a CLOS topology. This is because the CLOS topology is modular, scalable, and flexible. In this topology, multistage switches interconnect to support thousands of Ethernet ports, which in turn support servers within the data center.

We use a five-stage CLOS topology, as shown in Figure 3. Connections between servers and the leaf are typically direct-attach copper (DAC) cables for cost-effectiveness. In contrast, connections between the leaf and spine and between the spine and super/universal spine require fiber-optic single-mode fiber (SMF). This is because switches can be more than 300 meters apart. Multimode fiber can support a distance of only 70 meters, even at 25 Gbps. SMF also increases the longevity of the design, as it can support not only 100 Gbps but 400 Gbps and beyond.

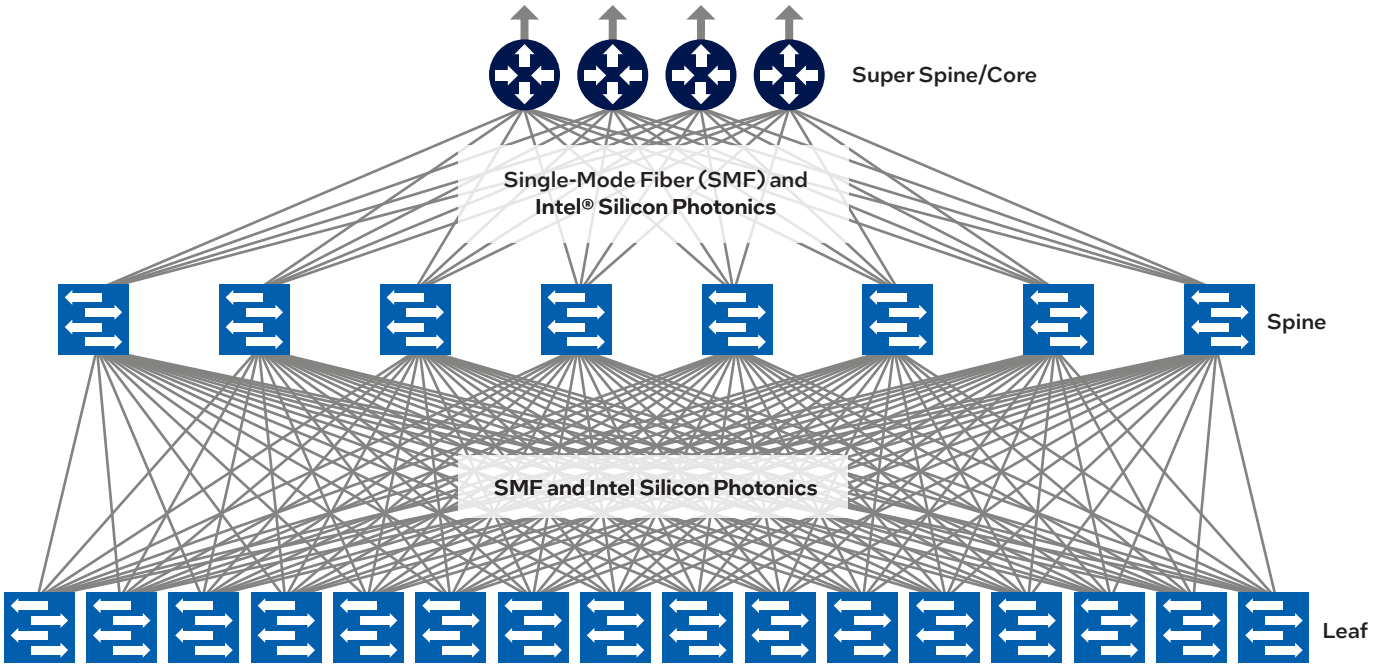

Figure 3. The optical connections (gray lines) can be hundreds of meters apart. We are migrating to SMF as we deploy more 100/400 Gbps connectivity.

Our CLOS topology allows fixed-form-factor switches at most of the topology stages, including top-of-rack (ToR), leaf, and spine. We also use a chassis-less design (small form factor) instead of investing in a bigger chassis. Our evaluations show the cost of a port is 2x to 3x higher than a small chassis with numerous interconnections.[3] Fixed-form-factor switches and a chassis-less design have enabled us to adopt new technology faster compared to using other designs. Therefore, we can increase our use of 100/400 Gbps connectivity at a lower total cost.

When we began to transition to 100 Gbps, we upgraded the fiber infrastructure to support SMF within our data centers. We selected a product that required only two strands (either 100G CWDM4 or 400G, depending on usage) for the transceiver, which lowered our fiber costs. As seen in Figure 3, there can be hundreds of transceivers required to interconnect the switches to build the high-speed data center fabric. To optimize cost efficiency, it was important to select the appropriate type of transceiver. We chose Intel® Silicon Photonics transceivers[4] over conventional optics or discrete laser-based technology. Our evaluation showed that Intel Silicon Photonics transceivers offer the following benefits compared to other solutions:

- Less power consumption
- Support for higher density
- Optimum total cost equation

We standardized on the 100G CWDM4 MSA QSFP28 model because it can span up to 2 kilometers on SMF. This model can also operate in a wide range of temperatures (0° to 70° Celsius) at low power (3.5 watts), which was important for a variety of data center deployments. Later, we also began using 400G QSFP-DD FR4 transceivers (2 kilometers, duplex SMF, 0°-70° Celsius, 10 watts).

Orchestration and Automation Framework 1.0

Comprehensive SDN at our scale is not possible without an automated management plane. We originally developed an orchestration and automation framework (we'll call it version 1.0 here) that integrates with the SDN controller to drive the overall orchestration in both the Design (that is, HPC) and Office/Enterprise data center environments. Consistent with our typical approach to deploying frameworks, we used existing in-house platform and hosting solutions whenever possible. Examples include server builds, DBaaS,[5] the Cloud Foundry application service, Ansible, our in-house Git repository system, DHCP, and DNS. We used these standard network services to aid in network automation.

When we first deployed the 1.0 version of our orchestration and automation platform, we were able to support our initial deployments by using short Python scripts that used Jinja2 templates and yaml seed files to enable automatic provisioning, streaming telemetry, and standard configuration management. The orchestration solution provided zero-touch provisioning (ZTP), where a network technician can edit a DHCP scope and power on a new switch, allowing it to provision itself enough to onboard into the orchestration system. From there, our scripts, templates, and yaml files could push the proper configurations and image onto the switch with just a few clicks.

However, we quickly realized we needed an NSoT for our network attributes—a centralized, API-accessible repository that could provide our scripts and templates with the attribute data they needed for device configuration, such as VLAN, ASNs, authentication servers, and management IP addresses. Initially, we used disparate yaml files for this purpose, but they quickly became unmanageable. We also found ourselves limited by the orchestration development environment because we could not reference other scripts and did not have access to an integrated development environment. We were limited to simply editing siloed scripts in a browser.

To solve these issues, we used our in-house, enterprise-grade DBaaS to create an NSoT database, for which we designed our own schema. We moved our code that generated configurations, along with the templates they consumed, into our in-house Git repository. To avoid having to manage application server operating systems, we built a Web Server Gateway Interface (WSGI) in our in-house Cloud Foundry environment, which provided a remotely accessible backend to our orchestration controller. Next, we changed the scripts on the orchestration platform to be lightweight, rarely changing "caller" scripts that gather local device data and pass it via API call to our WSGI. The configurations are then rendered off-server and returned to the caller script. Then, the orchestration platform deploys those configurations to the devices (see Figure 4).

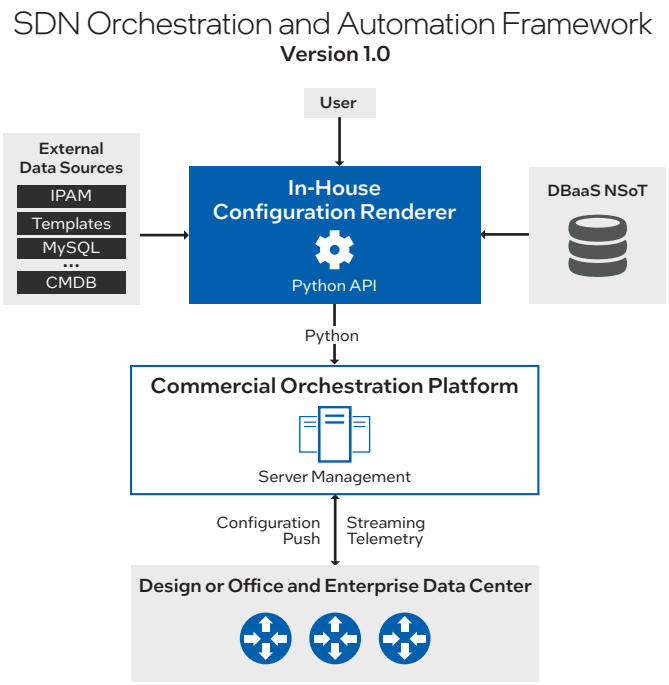

Figure 4. Our SDN orchestration and automation framework 1.0 used a commercial management plane portal, along with in-house capabilities for network configuration data, templates, scripts, and more.

[3] Based on internal Intel IT measurements.

[4] Intel® Silicon Photonics 100G CWDM4 Brief," intel.com/content/www/us/en/architecture-and-technology/silicon-photonics/optical-transceiver-100g-cwdm4-qsfp28-brief.html.

[5] IT@Intel, "Increase Business Velocity with Enterprise Database as a Service," https://www.intel.com/content/www/us/en/it-management/intel-it-best-practices/increase-business-velocity-with-enterprise-dbaas-paper.html.

In addition to our in-house resources, we chose a turn-key commercial management and orchestration platform to help us quickly provision and manage the new equipment, which consisted of more than 2,000 new switches across Office/Enterprise and Design data centers. The supplier offered several platform choices, including supplier-provided appliances, a VM image hosted on-site, and an as-a-service cloud instance. We chose the VM option. This entailed procuring our own infrastructure servers, adding our hosting-supported OS build, and installing an open-source hypervisor (KVM) to host our 17 regional orchestration clusters.

While the initial version of the orchestration and automation framework solved many problems, it wasn't flawless. Here are some of the challenges we faced:

- **Server sprawl and lack of scalability.** For each of the 17 regional orchestration clusters, we had three servers for failover redundancy (for a total of 51 servers). We had to manage all of these infrastructure servers ourselves. Additionally, each VM instance could scale only to a certain number of network devices. As our network grew, so did the number of VMs. Our team was good at its main job—designing and deploying networks. However, since the commercial platform was not easily scalable, we ended up performing most of the development work ourselves, which meant we had less time to spend on our primary responsibilities.
- **OS sprawl.** Although we benefitted from in-house hosting platforms without a fleet of application/database servers to manage, we found ourselves with a large fleet of hypervisors that were hosting our orchestration clusters. Even though we had standard builds for specific environments, we ended up with a total of 90 operating systems using three different Linux builds. We took advantage of our in-house managed Ansible platform to distribute files, perform upgrades, add monitoring agents, and perform other tasks.
- **Cost.** The recurring per-device licensing costs for the commercial platform grew into the millions as we expanded to thousands of network devices in our environment. In short, the system became cost prohibitive. For these reasons, we sought alternative solutions.

Benefits of Moving to an Open-Source Orchestration and Automation Solution

Due to the challenges with the 1.0 version of our orchestration and automation framework, we considered alternative solutions. We found only two or three acceptable, high-quality commercial offerings, and they did not satisfy our need for cost reduction. Therefore, we decided to explore the open-source community's orchestration and automation offerings.

There are several benefits of using open source:

- **Innovation.** Open-source solutions enable us to take advantage of collective innovation while mitigating the risks associated with increased technical debt. In addition, they meet our requirements for regular updates and patches that enhance Intel's security and agility. In fact, open-source software releases are often on par with or exceed vendors' support timelines. Embracing open-source, off-the-shelf solutions enables us to focus our resources on next-level innovation rather than routine maintenance.
- **Vendor neutrality.** Because the communities are "open," the software is vendor-neutral, protecting us from risk associated with relying on one or two suppliers.
- **Open standards for programmability.** In alignment with our previous commitment to network programmability, open-source automation and orchestration solutions typically use open standards, which promote programmability—meaning that our automation would be similarly programmable.

Orchestration and Automation Framework 2.0

The open-source orchestration and automation solution that we chose is called Nautobot. It is scalable and uses a software stack that is closely aligned with our existing stack. We were also confident that the chosen solution would easily deploy on Intel® architecture-based infrastructure. Our 2.0 version of the orchestration and automation framework (Figure 5 on the following page) continues to use our internally hosted DBaaS platform and other in-house capabilities. However, the new framework differs from version 1.0 in several crucial aspects:

- **Modular deployment with containerization.** Rather than being VM-based, 2.0 uses containers. We can spin up new containers (frontend/backend and workers) dynamically. We use our in-house secret store, container image repository, container orchestration systems, and load balancing infrastructure. We take an infrastructure-as-code (IaC) and continuous integration/continuous delivery (CI/CD) approach to alleviate administrative burdens.
- **Enhanced scalability.** The modular deployment enabled by containers allows us to scale horizontally as needed. Minimal configuration is needed to simply increase the number of workers and redeploy. In our previous system, we were often constrained by the physical resources of a bare-metal server, and the only way to scale was to add additional clusters.
- **No servers to manage.** We continue to use the in-house DBaaS servers, and we are now using infrastructure servers provided by Intel's own hosting team, which means our team no longer has to manage any servers at all. The ability to scale, combined with leveraging self-service load-balancing and database services, has provided much-needed relief to our administrative overhead. We can focus all our effort on network design and deployment.
- **Off-the-shelf database schema for the NSoT (Nautobot).** We now use an off-the-shelf NSoT system that has the schema already defined, which saves us a significant amount of work.
- **Our production NSoT is architected to spread across two geolocations, ensuring high availability with the ability to deploy each lifecycle automatically through CI/CD pipelines.** We adhere to a development, staging, and production lifecycle for our NSoT, which has proven useful in our Greenfield and Brownfield migration efforts as well as in our development processes. This gave our research-and-development team a place to simulate production changes without impacting our production NSoT itself.

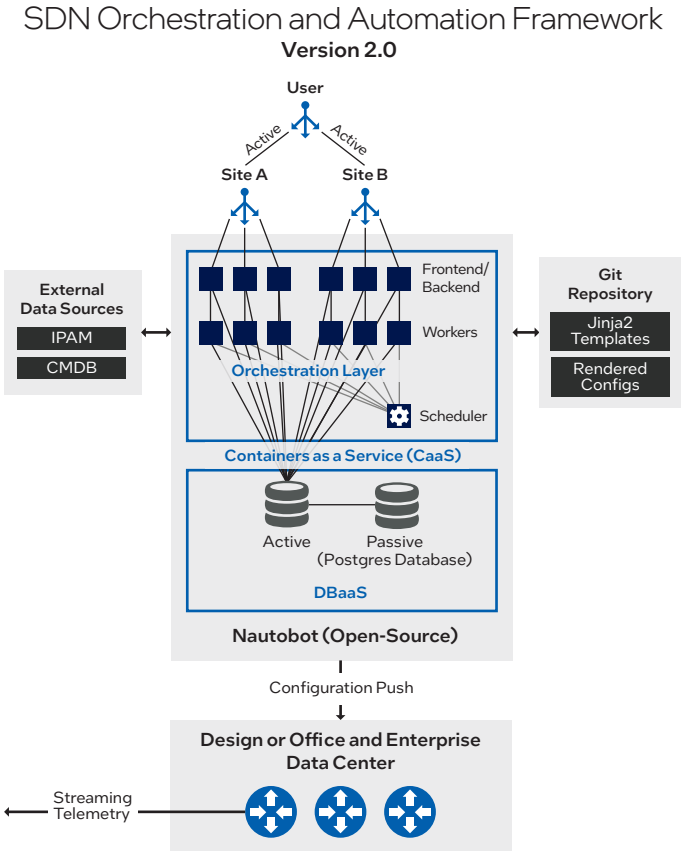

Figure 5. Version 2.0 of our orchestration and automation framework is based on open-source software, eliminates management of infrastructure servers, and is highly modular and scalable.

Application Layer

Having selected our core toolsets and deployed them on our in-house hosted platforms, we built the foundation of the application layer to facilitate not only migrations of our existing network environments but also the rapid provisioning of new buildouts. We focused initially on building the integrated components of the application, adhering to a "customize when necessary" mentality to enable our small team to meet our aggressive deadlines.

We first looked at all the data we needed to generate our standard configurations and mapped out how they would be represented in the NSoT's schema. We then crafted a GraphQL query that would fetch this data and pass it along to our repository of Jinja2 templates. This flow was completely included in the NSoT application, requiring only that we bring our knowledge of the network data components, some basic templates, and a repository.

We then were able to build a ZTP system, using the capabilities of our network platform with our new NSoT. Our out-of-the-box devices now boot up then download and run a small Python script that registers the device into our NSoT. Devices also upgrade their images based on the appropriate image definition for that platform (defined per-platform in our NSoT). This proved rather easy to tailor to our environment and gave us the flexibility to adapt over time as necessary.

Next, we needed a mechanism to get our network data into our NSoT, both for our existing networks as well as new provisioning. We used our NSoT's native self-service script-running feature to provide this service. This framework allowed us to build our own logic around obtaining relevant data components and provided a mechanism to insert the data into the proper areas of the data model. This was a fundamental component our network engineers needed to handle the volume of data that needed to be ingested.

Beyond the native components in the application stack, we've been able to use available libraries and modules to ease programmatic needs as they arise. This has made the system more approachable for our network engineers as they ramp up their automation skillsets.

Underlay Technology

The primary goal of the underlay network is to provide a routed path for the overlay networks, so that VxLAN Virtual Tunnel End Points (VTEPs) can communicate with each other. Our overlay network is built on top of a highly redundant underlay network, using L3 point-to-point connections to build our fabric (see Figure 6).

The underlay network is documented in a VRF global table, so that the information is available to all overlay networks. We use the same dynamic routing protocol that we use for overlay networks (although other options do exist), because doing so offers the following benefits:

- Ease of management, because we are using a single protocol.
- Lower complexity due to reuse of the same autonomous system and configuration blocks.
- Ability to scale well in large topologies.
- Support for the BGP open standard. We use Interior BGP at the leaf layer (redundant L2/L3 pairs) and eBGP between spine layers (no route reflectors needed).

Our approach to the underlay network differs slightly between the Enterprise and Design data center environments.

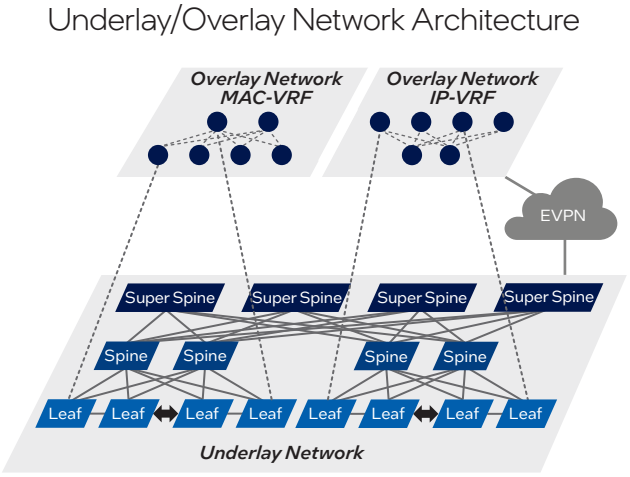

Figure 6. Our overlay network is based on a highly redundant underlay network.

Office and Enterprise Data Center

The Enterprise data center environment had use cases for overlay networks from the beginning, so we built the underlay network with that in mind. However, initially we did not choose to expose the underlay network's global VRF table outside of each data center, which prevented us from being able to easily extend an overlay across data centers (because overlays need underlays). We have implemented a workaround to this situation, but it would have been easier if the underlay was exposed across data centers.

Using an underlay/overlay approach in the Enterprise environment enables us to extend L2 VLAN everywhere over an L3 network using Type 2–Host MAC and IP Addresses (MAC-VRF), so there are no more looping outages. Also, we can isolate networks using VRF tables and extend that isolation throughout the fabric.

Design Data Centers

The Design environment is a large-scale HPC infrastructure with multitenancy requirements. We built the leaf/spine infrastructure using the same principles and practices as we would for an underlay that was going to support an overlay:

- Direct peer-to-peer peering to physical interfaces for faster convergence.
- Equal-cost multipath to help improve latency and optimize data flow.
- All devices have loopback interfaces that are in the global default VRF table but are not used to peer with for underlay.

To support multitenancy, Type 5–IP prefix information (IP VRF) model is used.

One of the key underlying challenges was to scale the multiple PODs interconnected in a mega data center. At one of our large data centers, we had to implement a five-stage CLOS architecture by introducing a chassis-based super-spine layer that can support 100 Gbps and 400 Gbps port speeds with minimum oversubscription between the spine and super-spine layers.

Overlay Technology

An overlay network creates a logical structure on top of the physical structure of the underlay network. In our Office/Enterprise data centers, we needed to provide L2 mobility across the underlay fabric, while in the Design data centers, we built an L3 secure enclave overlay network. Some important attributes of our overlay networks include the following:

- VxLANs allow encapsulation for cross-site network extensions, enabling both VLAN and VRF extensions. We use the BGP's EVPN extension for dynamic VxLAN learning. We are also currently conducting a proof of concept to explore the use of static VxLAN mapping for cross-site network extensions.
- The EVPN control plane is a distributed, dynamic learning plane that is not tied to a central controller.
- Distributed L3 means that within zones, we can use anycast IPs for distributed default gateways, which helps ensure the shortest routed path between systems in the same VRF table.

Our overlay networks build enclaves, which are networking environments that operate with a common set of security controls. The demilitarized zone (DMZ) is a networking environment that buffers between discreet networking environments and consists of a VPN and Proxy environments. Typically, one of these is untrusted, which usually is the internet. Then, we define separate security zones or enclaves (see Figure 6) for external and secure internal hosting (also called a secure internal zone, or SIZ). The enclaves include backup and recovery networks, pocket networks (dedicated network environments that are application-purpose-built and protected by a firewall) and network management.

Office and Enterprise Data Centers

Because the Enterprise domain encompasses a diverse range of use cases, each with its own unique security requirements, we deployed enclaves in our Enterprise data centers from the beginning (Figure 7). For internally hosted applications, we gradually established multiple security zones to isolate components of two-tier and three-tier applications. In certain instances, we also deployed specific application-level enclaves. Generally, we create distinct security zones for each internet-facing service and application.

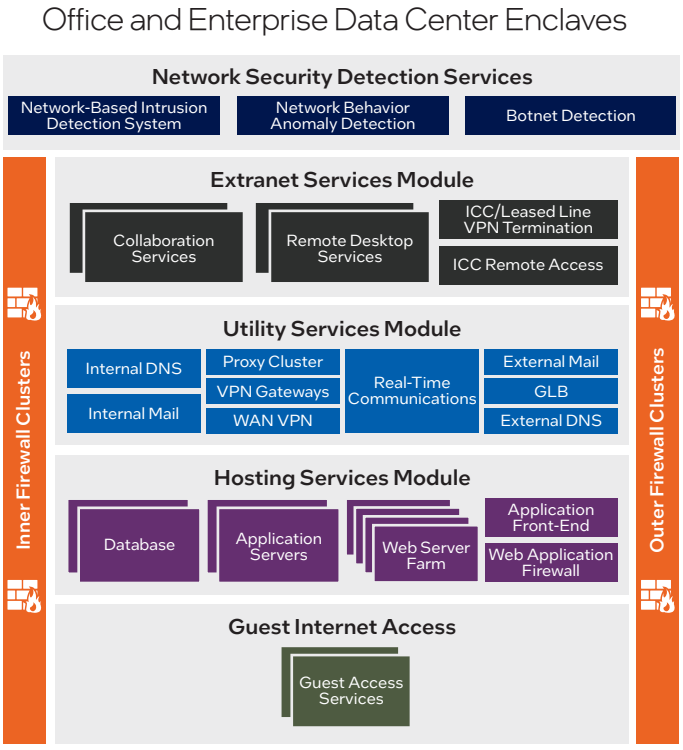

Figure 7. We use enclaves (indicated by the colored boxes), which are networking environments that operate with a common set of security controls, to increase security posture.

Design Data Centers

Our Design environment originally did not need overlay networks. But as new use cases were introduced, we needed to move beyond router ACLs to adding enclaves with special security features. In particular, adding next-generation firewall-grade security for select HPC networks was challenging, labor-intensive, and often took weeks to complete because of the need for separate network switches and dedicated racks. We resolved these issues by creating a solution that is portable, granular, and scalable:

- **Portable.** The ability to provide security to disparate, existing networks as well as new networks and be location-independent within a given data center.
- **Granular.** The ability to run select subnets through the firewall while letting others bypass.
- **Scalable.** The ability to support multiple tenants with low configuration overhead, where the security posture is handled by the Information Security Team (not by the network team).

Our solution starts at the leaf, where we use a VRF construct for our segmentation, providing security by routing (or lack thereof). All subnets inside the VRF can freely talk to each other but cannot talk to anything outside the VRF. This solves the segmentation on the leaf, but the VRF isolation is only locally significant. The next component of our solution involved extending that VRF across the data center to wherever our firewalls were located, often several hops away. We used VxLAN to extend the VRF and used EVPN for the controller. Then, the service leaf pair that was connected to the firewall could serve as the VTEP, decapsulating the VRF traffic and sending it to the firewall policy for processing.

The HPC security solution provides the following benefits:

- We can use an app that spans multiple subnets in multiple physical areas of the data center, and all these subnets can be in the same isolation bucket (VRF). The subnets can talk to each other without having to go through the firewall, but any other traffic in/out of that bucket must traverse the firewall policy.
- We are able to extend any enclave or secure network throughout the data center; there is no need to move enclosures.
- We have improved provisioning time; now it takes only two hours instead of eight days to secure the network.
- All security happens at the firewall with enhanced monitoring and logging.
- There is no impact on non-secure network traffic flow.

We have implemented this new security solution across all Design data centers, and it is pervasively used for segmentation and isolation.

Migration Strategy

To move our Office/Enterprise and Design data centers to the new SDN architecture, we built the new IP fabric in parallel in the data center. Any new systems were deployed directly to the new fabric while we began migrating existing racks a few at a time to the new fabric. For the HPC environment, there was no downtime for a compute rack move, while file server migration was done without downtime by working closely with the file server administrator team. In Enterprise data centers, we used quarterly scheduled downtime to migrate L3, firewalls, and load balancers to the new fabric (only one period of downtime per data center), and migrated one row at a time to the new fabric. We found that it takes six to eight hours of downtime for large data centers and three to four hours for medium data centers for the Enterprise migration.

Figure 8 on this page and Figure 9 on the next page, respectively, show our migration strategy for the Enterprise and Design data centers. Because the data centers have a low tolerance for outage windows, we adopted a phased migration approach, where layers from the legacy environment are removed first, with the client connections still intact. Subsequent phases involve staging redundant connections to the newly built infrastructure and then simultaneously cutting the links to the legacy environment while bringing up the new links.

We have migrated 90% of the data centers over the last five years (Figure 10 on the following page).

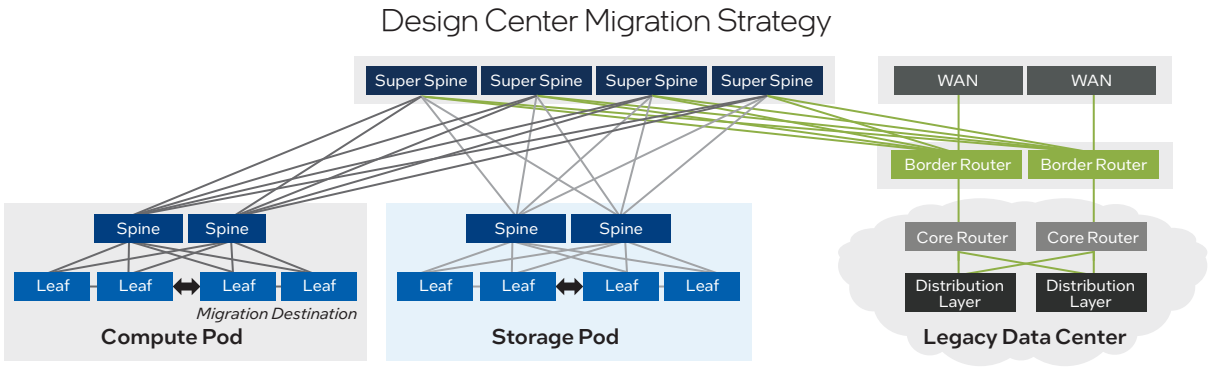

Figure 8. Interconnect leaf layer connecting legacy data center with new fabric.

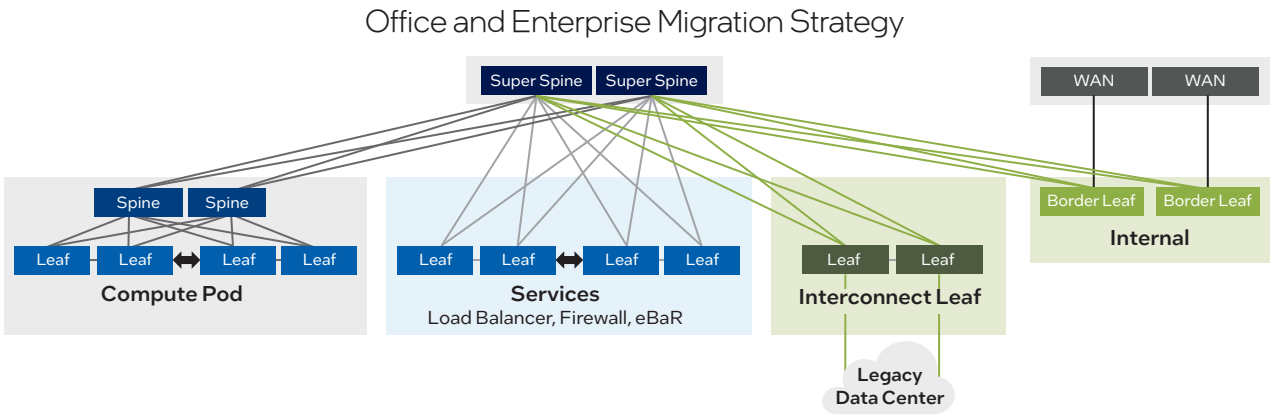

Figure 9. Border router connecting the old topology with the new topology.

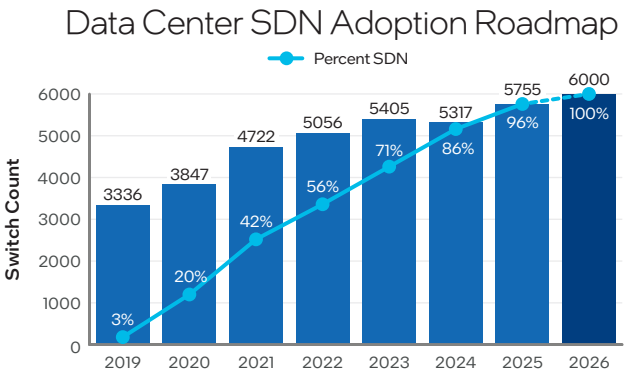

Figure 10. Over the last five years, we have migrated 90% of all data centers to the new SDN architecture.

Results

Our adoption of SDN and automation architecture has provided numerous benefits to Intel, as detailed in the following sections.

Overall SDN Benefits

- **Network provisioning improvement.** It used to take nearly eight hours to provision networks for entire racks of servers from TOR switches. Multiple manual components contributed to the long lead time, including initial switch standard software and baseline configuration, provisioning of L2 and L3 networks, configurations of L2 and L3 redundancy, setup of DNS records, and, finally, giving the correct persona to a switch. With the new software-defined and automated architecture, all these components are built as part of baseline configurations and integration with IP address management, which has reduced the provisioning time to less than two hours.
- **Improvement in reliability and stability.** We have improved the reliability of the data center and reduced the number of performance-related incidents by 70%. Multiple factors contributed to these improvements, such as using multiple 100 Gbps connectivity (which increased the bandwidth by 2-8x) and standard deployment automation that eliminates human errors in configuration. As shown in Figure 11, we have not had a major network incident in over four years across Design data centers worldwide — amply illustrating the robustness of the solution.
- **Efficiency.** We saw 25% year-over-year growth in Intel's Design data centers. Our network team was able to support this higher volume of work without increasing staff. This was only possible due to the direct value of SDN and automating Day 0 and Day 1 tasks. We have achieved greater than 20% efficiency improvements with the SDN architecture to date.
- **Flexibility.** An additional benefit of the open, standards-based SDN and orchestration layer is the ability to add custom network layers to meet unique business requirements. In contrast, a closed-loop, supplier-centric SDN solution offers very limited ability to make these types of changes. Over the years, we have made multiple value-add changes to the automation to adjust to the architecture changes in the data center.

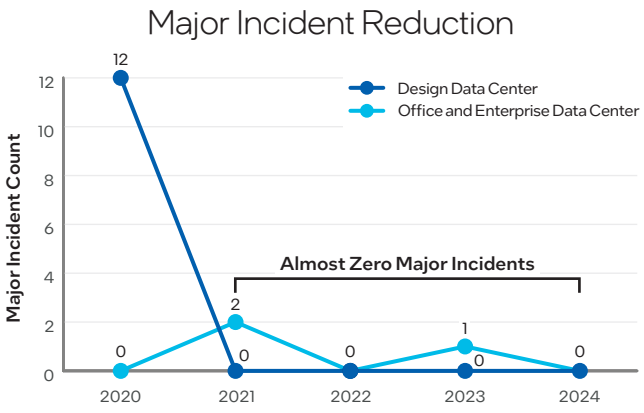

Figure 11. Our adoption of SDN across Design, Office, and Enterprise data centers has enabled us to nearly eliminate major network incidents.

Benefits Specific to Orchestration and Automation 2.0

As we began integrating our version 2.0 of the orchestration and automation framework through a pilot project and some small data center migrations, we observed some expected efficiency gains, as well as some unexpected benefits:

- **Greenfield device provisioning:** As expected, our new ZTP is paying dividends on each new installation. The process requires only a single step (minimal initial device data upload) compared to the previously required multiple steps (initial data upload, phase 1 ZTP, image delivery, baseline configuration generation, and configuration push). Our new process also alleviates limitations from the previous system by pre-loading the appropriate software image (determined by the values in our NSoT).
- **Brownfield data import:** The tool we developed to pull data in from existing devices has proven accurate and efficient. The engineer simply inputs a list of all devices for the environment they are migrating, and our script crawls each device, fetches the relevant data values, and then stores them appropriately based on the schema of our NSoT.
- **Reconciliation to standard:** While our previous system generated the majority of the configurations to standard, our engineers had the latitude to easily customize a specific device's configuration, leading to many deviations. An unexpected side benefit to our method of migration is a forced reconciliation to standard, as the new configurations are rendered completely against the NSoT data. The trade-off is a more time-consuming final step in the migration to ensure the reconciliation to standard doesn't have a negative impact. However, the upside is an environment that is significantly more compliant—and as our environment standardization increases, the ease and ability to automate it correspondingly increases.

Next Steps

Over the next three months, we plan to complete the migration of all our data center network environments to version 2.0 of the orchestration and automation framework. Following this, we will shift our focus to enhancing the framework by integrating network validation tools (both pre- and post-change), streaming telemetry, and expanding self-service functionality and IT Service Management integration. As our network provisioning becomes increasingly automated, we are already undertaking efforts to transform our network engineering team's skill set and mindset to align more closely with DevOps methodologies. Our vision is to evolve into a team that possesses not only a deep understanding of the networking aspects of our environment but also the ability to automate and develop it—an amalgamation of skill sets that are increasingly rare in today's job market.

Conclusion

We believe that our selection of open, standards-based technologies for constructing underlay and overlay networks, combined with an open-source orchestration layer, has been essential in granting us the flexibility to adapt to business needs and realize the value of a broader ecosystem.

Our network architecture and strategy are deliberately designed and data-driven to help ensure the performance levels and network availability that our customers need to succeed.

The leaf-spine underlay architecture utilizing open, standards-based protocols within an SDN environment allows us to accommodate the 25% annual growth in network capacity while achieving a 4x reduction in provisioning time. This new resilient solution significantly enhances the reliability and stability of our data center network. We can now integrate various security use cases onto a unified infrastructure and onboard new security applications with minimal additional effort. The foundation for these advancements was the comprehensive development of automation and standardization of the data center architectural elements, enabling them to be easily replicated in modular building blocks.

Related Content

If you liked this paper, you may also be interested in these related stories:

- IT@Intel: Validating and Evolving Intel IT's Multicloud Strategy
- IT@Intel: Data Center Strategy Leading Intel's Business Transformation
- IT@Intel: Building a Multi-Cloud-Ready Enterprise Network
- IT@Intel: Transforming Industrial Manufacturing with Software-Defined Networking

For more information on Intel IT best practices, visit intel.com/IT.

IT@Intel

We connect IT professionals with their IT peers inside Intel. Our IT department solves some of today's most demanding and complex technology issues, and we want to share these lessons directly with our fellow IT professionals in an open peer-to-peer forum.

Our goal is simple: improve efficiency throughout the organization and enhance the business value of IT investments.

Follow us and join the conversation on **X** or **LinkedIn**. Visit us today at **intel.com/IT** if you would like to learn more.