POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Design of an Enterprise-Grade Software-Defined Datacenter Network

Supervisors

prof. Marco Giuseppe Ajmone Marsan (Politecnico di Torino)

....

prof. Giovanni Pau (Sorbonne Université)

....

Candidate

Gianstefano Monni

[illegible]

April 2018

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE MASTER’S DEGREE
OF DOCTOR IN COMPUTER ENGINEERING
AT THE POLYTECHNIC UNIVERSITY OF TURIN

To dad and mum.

ACKNOLEDGEMENTS

I would like to thank my supervisors, prof. Pau, whose support gave me the confidence to start this work and really helped me in gathering the ideas for this paper, and prof. Marsan for his precious and continuous feedback during the development of this work.
A special thank is for my wife Francesca, who always supported me, and without whom this journey would have never been started, and, finally, I would like to thank Alex, Jeff, and all the colleagues of the Zurich Enterprise Architecture and Service Design team whose concepts and principles inspired part of this work.

Table of Content

Table of Figures

List of Tables

1 Introduction

1.1 Abstract

This paper discusses the design of a modern Enterprise-grade Software-defined datacenter network (DCN). The work begins with the identification of the high level and domain-specific requirements and constraints of modern DCNs, then introduces some design principles and technical solutions (mainly network virtualization, software-defined networking, network function virtualization and network function chaining) to overcome the constraints and to implement the requirements presented earlier. Then, a distributed service delivery model for the Enterprise DCN and three high-level designs (HLDs) using different platforms are introduced and compared to each other and to a theoretical model, and against a well-known Datacenter Market report.

The last chapter, using some guidelines, best practices and lessons learned by the early adopters of SDN (Facebook, Google, and Microsoft), briefly presents a possible design for a future Software-Defined Enterprise Global network able to face the challenges posed to a world-wide Enterprise company.

1.2 Organization of the document

The paper is organized in three sections and sixteen chapters:

- SECTION I – DATACENTER NETWORK REQUIREMENTS (Chapters 1-3): this section introduces the most important high-level and domain-specific requirements for an Enterprise DCN.
- SECTION II - DATACENTER NETWORK SPECIFICATIONS AND DESIGN TOOLS (Chapters 4-9): this section identifies the specifications for the requirements described in Section I, then the core deployment concepts (network virtualization, Software-Definition, Network Function Virtualization and Service Function Chaining) are briefly discussed. The section finally introduces three different technologies that will be used in Section III to propose different implementations of an Enterprise DCN.
- SECTION III – DESIGN OF AN ENTERPRISE DATACENTER NETWORK (Chapters 10 -16): the section presents a service delivery model for the Enterprise Datacenter network and then three different high-level designs adopting the proposed model and implementing the requirements listed in Section I and using the solutions proposed in Section II are presented and compared.

1.3 Scope of the document

This document focuses on the requirements, specifications and high-level design of an Enterprise Datacenter network. The design of other features of the datacenter (i.e. power, cooling, storage, compute) is out of scope of the document.

1.4 Abbreviations, acronyms and Descriptions

Abbreviation	Description
AAA	Authentication, Authorization and Accounting
AD	Active Directory
BUM	Broadcast, Unknown, Multicast (Traffic)
BYOD	Bring Your Own Device
CAPEX	CAPital EXPenses
CI/CD	Continuous Integration/ Continuous Delivery/Continuous Deployment
CNF	Carrier Neutral Facility
COTS	Commercial Off the Shelf
CSB	Cloud Service Broker
CSP	Cloud Service Provider
DCI	Datacenter Interconnect
DDC	Distributed Datacenter
DCN	Datacenter network
DNS	Domain Name Service
DDC	Distributed Datacenter
ECMP	Equal Cost Multi Path
GSLB	Global Server Load Balancer
IG	Inter-Zone Gateway
IPAM	IP Address Management
NBI	Northbound interface

Abbreviation	Description
OPEX	OPerating EXpenses
PDU	Protocol Data Unit
PNF	Physical Network Function
QOS	Quality of Service
SAE	Service Access Exchange
SDN	Software defined Networking
SIAM	Service Integration and Management
SPN	Service Providers Network
TCO	Total Cost of Ownership
TIAM	Technical Integration and Management
UAT	User Acceptance Test
VNF	Virtual Network Function
WAN	Wide Area Network

Table 1-1- List of Abbreviations, acronyms and their description

1.5 Main Revisions and Release Plan

Date	Version	Notes	Pages
14/09/2017	**DRAFT 0 - R22**	**TOC, Chapters: abstracts, draft structure**	**58**
25/09/2017	DRAFT 1 – R2	Chapters 1-4 - Completed Section I	114
09/10/2017	DRAFT 2 – R2	Chapters 5-9 - Completed Section II	156
02/12/2017	**ALPHA 4 – R12**	**Chapter 10 - 16 - Completed Section III**	**197**
23/01/2018	BETA 3	Added comparison of MSDCNs, reviewed Chapter 16	220
04/02/2018	**RC1**	**Added Future SDN and Submitted for final Exam**	**223**
31/03/2018	**FINAL**	**Final Editing and updates. Submitted for the discussion**	**226**

Table 1-2 – Main Revisions History and Release Plan

SECTION I- DATACENTER NETWORK REQUIREMENTS

2 High level Requirements

2.1 Abstract

This chapter presents the high-level requirements for an Enterprise datacenter. The requirements are defined in natural language, and originate from different standpoints:

- Business: these requirements, often implicit, are the highest-level requirements, but also from the main stakeholders' perspective, the most important ones, on the base of which all the others are defined.
- Security: security requirements are defined as a combination of Confidentiality, Integrity Availability of Data and infrastructure. Non-repudiation, Resiliency, and DR capability are usually added to this standard definition (some could argue that the latter two are part of availability)
- End-user: the user of the datacenter covers a very broad spectrum of categories. In this paper, by *Datacenter user* we mean the Corporate/external user, not the developer, the sysadmin or other technical user.
- Other/external requirements: regulatory (i.e. GDPR), compliance to domain-specific standards, corporate rulesets, best practices, standards, etc.

2.2 Business requirements

In an Enterprise, the main driver to build an infrastructure such a datacenter, and a datacenter network, is always to (further) enable / facilitate business processes. This paragraph presents the requirements as they would be stated from the top stakeholders, who have a very high-level vision of the infrastructure (*must enable business*). These requirements are then further detailed in the following paragraphs.

2.2.1 Effective and efficient

First and foremost, the datacenter network, as any other infrastructure, must enable business to accomplish its goals as fast as possible, and carry out them (effectively) without wasting resources in the process (efficient).

2.2.2 Ability to change

The market is always changing, and, with it, the needs of the business; this dynamicity translates into new requirements for the architecture driving the design of the infrastructure.

So, the ability to *adapt, change and transform* is a core feature, and often an implicit requirement, of any Enterprise/Business solution.

2.2.3 Improve business operations

Another implicit requirement, sometime misunderstood, is the management' expectation that a new infrastructure must improve business operations. This requirement should always be considered even when, as quite often happens, operations are more connected to business constraints rather than infrastructure,

2.2.4 Secure

Every architecture and infrastructure must be able to handle business applications and data "*securely*". In this context "security" is an umbrella term covering not only the *classic* concepts of confidentiality, integrity and availability of data and applications but also including resiliency, safety, non-repudiation and the overall protection of business core assets.

From business perspective, every IT architecture should be secure(d) In this broad sense. An introduction to Zero Trust Model applied to Datacenter network security architecture is presented in [1] and shows how security of complex systems could/should be seen as a holistic process, involving the ability to relate and co-relate different sources and the need of defining a fuzzy, continuous level of "trust" and not just a binary property (Secure vs. not secure).

2.2.5 Cost Effective

As already mentioned, cost effectiveness is an implicit, and often central, business requirement. The resources allocated to implement/transform the datacenter infrastructure should be used in the most effective way from the cost perspective. This often implies also to consider that Enterprises are shifting the IT Cost from Capex to Opex (see below).

2.2.5.1 CAPEX and OPEX

Operating expense, or Opex, is the cost for running a product, business, or system: an Opex is an ongoing cost that recurs regularly, usually monthly. It is generally provided under a contract by the service provider. Examples of operational expenses are utility bills. These costs recur regularly on an ongoing basis. Enterprise cloud storage and IaaS products are IT-related examples of operational expenses. The Opex counterpart, called capital expenditure (Capex), is the cost of developing or providing non-consumable parts for the product or system, or a one-time cost to acquire an infrastructure.

2.3 Enterprise Requirements

From an enterprise perspective, the following capabilities must be provided:

- IT Service Management
- Service Delivery through Service Catalogue
- Service monitoring / Reporting
- Information/lifecycle management

2.3.1 IT Service Management

Any IT solution must support:

- the implementation of ITIL V3, IT Service Management (ITSM) Framework for problem, change, incident, release, configuration and capacity management disciplines;
- the design and adoption of an ITSM Service Desk to support partner organizations and customers;
- a set of service tools enabling the provisioning of IT services (e.g. integration between change, release, configuration management tools with automated deployment of IT services).

2.3.2 Service Catalogue

The architecture must support the development and implementation of a service catalogue and service portfolio management function.

2.3.3 Service monitoring / Reporting

To improve service quality through monitoring and reporting, the architecture must be able to set up standardized metrics and to measure performance against Key Performance Indicators (KPIs).

2.3.4 Information/lifecycle management

To provision storage solutions in a cost-effective way, while ensuring enforcement of Company policies, the architecture must allow Partner organizations to identify the minimum data retention policies applicable for information lifecycle management.

2.4 Business User requirements

2.4.1 The disappear of the work-place

"Work is verb, not a noun" [2]: Pip Marlow's, managing director for Microsoft Australia, statement is becoming increasingly important. Staff in modern Enterprises is facing a major change: In the past, the typical workflow was commute, logging in from 09 am to 05 pm, and then return home. For the most part, critical tasks were confined to "*standard*" business hours. Today's approach is often "I do it *with my own device, anytime from anywhere*" which translates in three transformations:

1. *Consumerization of IT*: Staff does not use only Enterprise devices: quite often they want to use their own personal device.
2. *Mobility and collaboration*: staff is no longer limited to the office.
3. *Always ON, always connected*: staff does not work only during "standard" business hours.

2.4.1.1 Consumerization of IT and BYOD

In a study conducted in 2014 [3] , IDG Enterprise defines *consumerization of IT* as the *propensity for users' experiences with technology as consumers to affect their expectations regarding their technology experiences at work*. The study shows that:

- *"83% of organizations are planning to invest in mobile technology in the next 12 months, with tablets and employee training (49% each) leading the priority list.*
- "The top four investment priorities are: buying smartphones (43%), network consulting and integration services (34%), application development, improve user experience (33%)
- "Enterprises with over 1,000 employees are significantly more likely to invest in mobile apps to increase customer satisfaction than their SMB counterparts (46% versus 38%).
- Enterprises are also outspending SMBs on mobile apps designed to increase customer retention (38% versus 20%). The following graphic shows the breakdown by benefit area"

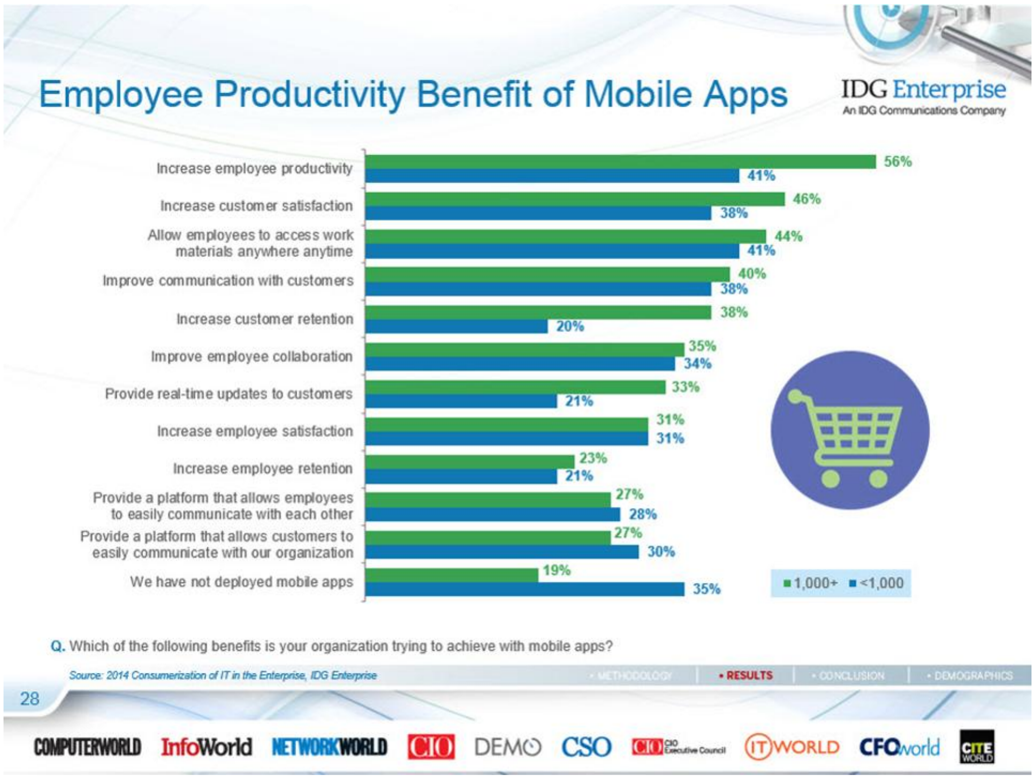


Figure 2-1 - Employee Productivity benefit of mobile Apps [3]

2.4.1.2 From Location-based to Mobile-Centric

Personal mobile device support is expected to increase significantly, with tablets leading all categories with an increase of 12% from 43% today to 55% in the next 12 to 18 months. The following graphic illustrates the trend across device categories:

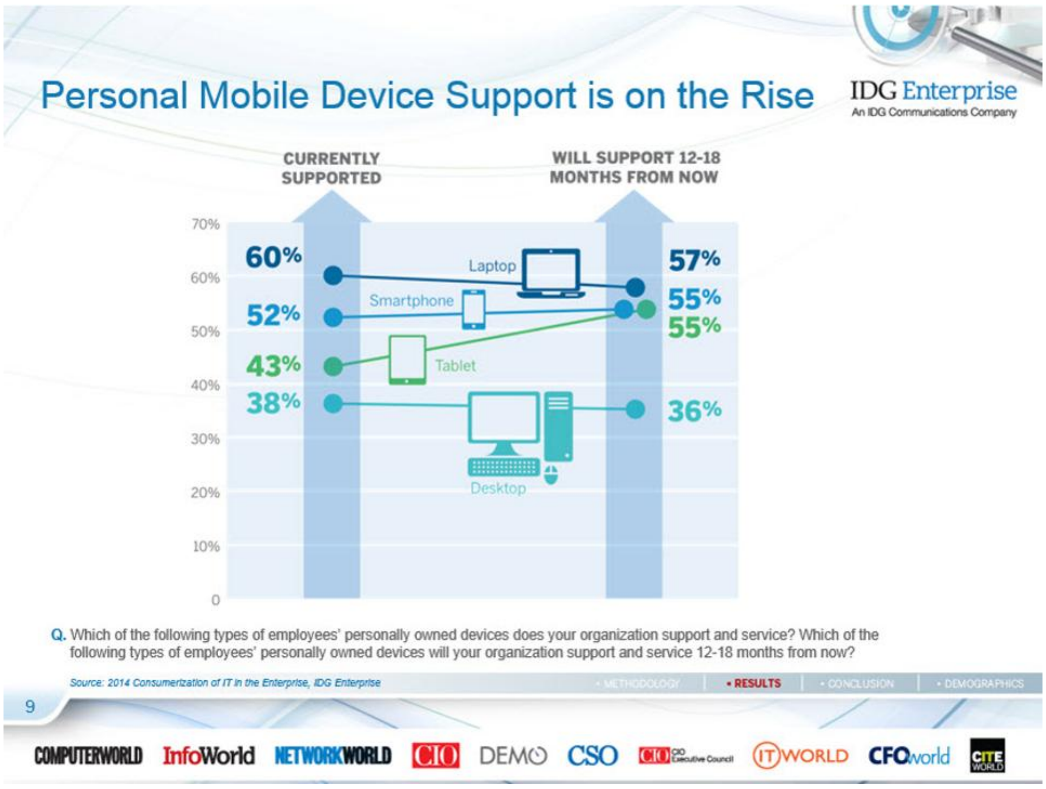


Figure 2-2 - Rising of Personal Mobile Device support in the Enterprise [3]

56% of respondents indicated that either the CIO/top IT executive or the IT department were the primary leader in driving change through the consumerization of IT at their organizations. It is worth mention that the support to mobility is needed not only for standard customer facing/frontend applications but also for internal/corporate users and third parties. This need triggers a change on the deployment model and has an impact on dataflows inside the datacenter as well.

2.5 Software Development requirements: Dev/OPS and Infrastructure as Code

The dynamicity, elasticity, responsiveness, the need to adapt and transform fast are the main drivers of a profound change in both hardware and software infrastructure delivery models.

On the software side, the need to quickly adapt and implement software changes triggered a change in the software delivery models, from the classical waterfall shown in Figure 2-3 to a continuous process called Continuous integration/ Continuous Deployment/ Continuous Delivery model (CI/CD).

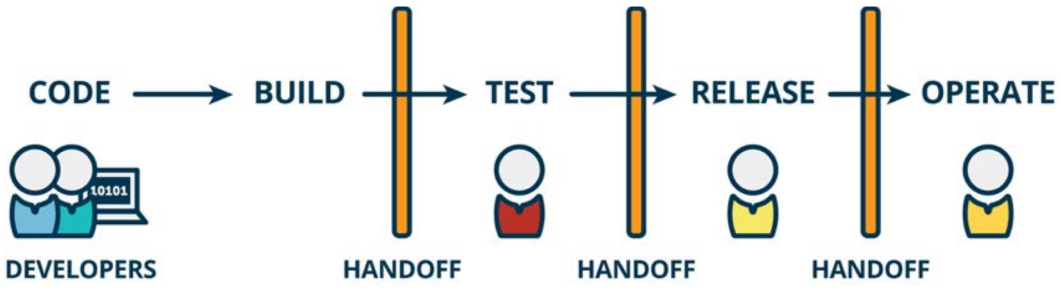


Figure 2-3 - Waterfall model

On the hardware side, the very same need Is driving a massive change in the IT infrastructure and operations management model: to dynamically adapt, change and transform the infrastructure, the infrastructure must be dynamic, fluid: essentially the infrastructure is (should be) seen as something controlled by a library, an API: infrastructure (computing, storage, network) as a code is born (for a deeper discussion about Infrastructure as a code, see [4])

2.5.1 From Waterfall to CI/CD

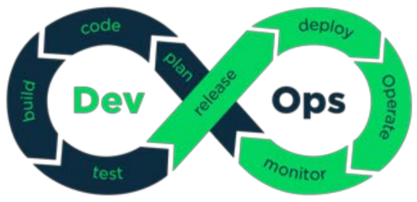


Figure 2-4 - Dev/OPS and CI/CD relation

Nowadays, continuous delivery, continuous deployment and continuous integration are common approaches to modern software development, moreover in the Enterprise context, where the need for agility, speed and overall responsiveness is one of the major transformation drivers. In the next few paragraphs a brief description of CI/CD is presented and then the connection with Enterprise Datacenter network is shown.

2.5.1.1 Continuous Integration

Continuous Integration (CI) is the practice of constantly merging development work with the Main branch so that changes can be tested into the context of other related changes.

The underlying principle is to test the code as soon and as often as possible, so issues can be caught as soon as possible. In the CI process, most of the work is done through automated testing. Usually a build server specifically designed to perform these tests is used, so that the development team can continue merging requests even during the testing phase.

2.5.1.2 Continuous Delivery

The constant delivery of code to an environment when the developer team feels the code is ready to release (it could be UAT, staging or production) is called *Continuous Delivery* (CD). The principle behind CD is that the code is constantly delivered to a user base, whether it be Quality Assurance (QA) or customers for review and inspection.

The basis of CD is to have small batches of work continually fed to the next step, so it can be consumed more easily, and issues can be found earlier.

2.5.1.3 Continuous Deployment

The deployment or release of code to production as soon as it's ready it is called *Continuous Deployment* (CD). All the testing can be completed before merging to the Mainline branch and can be executed on production-like environment, and this removes the need for a long UAT process or for a large batching in staging before production. The production branch is always stable and ready to be deployed by an automated process. The automated process is key because it anyone should be able to perform it in a matter of minutes (preferably by the press of a button).

After the deployment, logs must be inspected to determine if the key performance indicators are affected, positively or negatively. Some of these metrics may include revenue, response time or traffic and should be graphed for easy consultation.

The key CD feature is that it needs continuous integration and continuous delivery because without them, errors will be generated in the release.

2.5.1.4 How They Work Together

Several automation mechanisms must be in place when the CD process is implemented. The CI build server must be fully automated from continuous delivery to staging, and the process must be capable to deploy to production automatically.

In an ideal workflow, the CI/CD process could be automated from begin to end:

1. Developers check in the code to development branch.
2. CI server imports the change, merges it with the Master/Trunk/Mainline, performs the testing and merges to staging environment based on test results.
3. If Step 2 is successful, developers deploy it to the staging environment and QA tests the environment.
4. If Step 3 is passed, the move to production is voted and the CI server picks this up again and determines if it’s ok to merge into production.
5. If Step 4 is successful, it will deploy to production environment.

2.5.2 The Dev/Ops approach

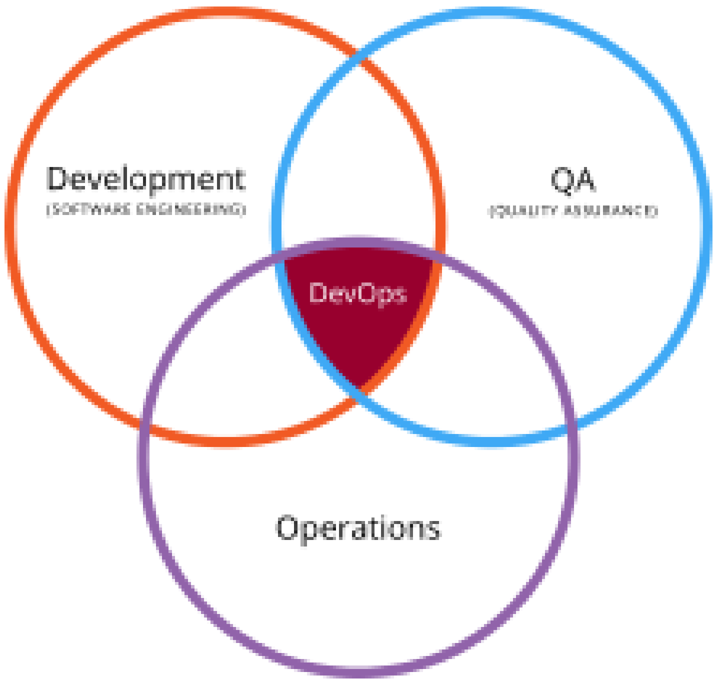


Figure 2-5 - Dev/OPS as the common ground for QA, Ops and DEV

In his book [5], Mike Loukides describes the state of DevOps and the current debate about different approaches. Here is how DevOps is defined in the book:

…modern applications, running in the cloud, still need to be resilient and fault tolerant, still need monitoring, still need to adapt to huge swings in load, etc. But those features, formerly provided by the IT/operations infrastructures, now need to be part of the application, particularly in “platform as a service” environments. Operations doesn’t go away, it becomes

part of the development. And rather than envision some sort of uber developer, who understands big data, web performance optimization, application middleware, and fault tolerance in a massively distributed environment, we need operations specialists on the development teams. The infrastructure doesn't go away – it moves into the code; and the people responsible for the infrastructure, the system administrators and corporate IT groups, evolve so that they can write the code that maintains the infrastructure. Rather than being isolated, they need to cooperate and collaborate with the developers who create the applications. This is the movement informally known as "DevOps".

2.5.2.1 Dev Vs Ops

Dev and Ops often have opposing priorities:

- Development (Dev) produces innovation and delivers it to the users as fast as possible.
- Operations (Ops) must ensure that users have access to a stable, fast, responsive system.

While Dev and Ops' goal is to satisfy the business, their views of how to achieve this goal are inherently divergent. It is a symptom of what is referred to as Water-SCRUM-Fall: developers want and must deliver new features, fast. Operations want and must produce a stable system, always.

When Developers and Operations lived in completely separated environments, this dichotomy was not an issue: both were working on a schedule, with few interactions, typically at release times. Developers knew when the release date was. Stability was maintained "easily".

The virtualization of systems and the CI/CD process triggers a massive shift on the infrastructure and on the operational model. The number of environments and their instances increase by several orders of magnitudes. Now CI Builds are released daily, often multiple builds a day. All these releases must be tested and validated. This requires new environment instances to be brought up as fast as possible, often with configuration changes. Logging into server consoles to perform manual changes individually isn't an option anymore. Moreover, the need for speed pushes further. Developers' build creates a backlog, as the systems to just test them on are not available as required.

2.5.2.2 Infrastructure as code

To address the battle between Dev and Ops, two concepts should be introduced:

- Versioning Environments: maintaining multiple configurations and patch levels of environments that are now required by development, on demand, needs Ops to change the way they manage change and operate these environments. Any change Ops makes to an environment should be viewed as creating a new 'version' of the environment (not simply changing a text file via SSH). The only way to do this properly is by scripting all changes. When executed, these scripts would create a new version of the environment they are executed on. This process, while maintaining Ops best practices (ITIL or otherwise), rationalizes and simplifies change management, enabling it to scale.

- Cycle time: the average time taken from the time a new requirement is approved, a change request is submitted or a bug that needs to be fixed via a patch is identified, to the time it is delivered to production.

Both these needs – versioning environments and minimizing cycle time - can be addressed by approaching and managing the Infrastructure as code

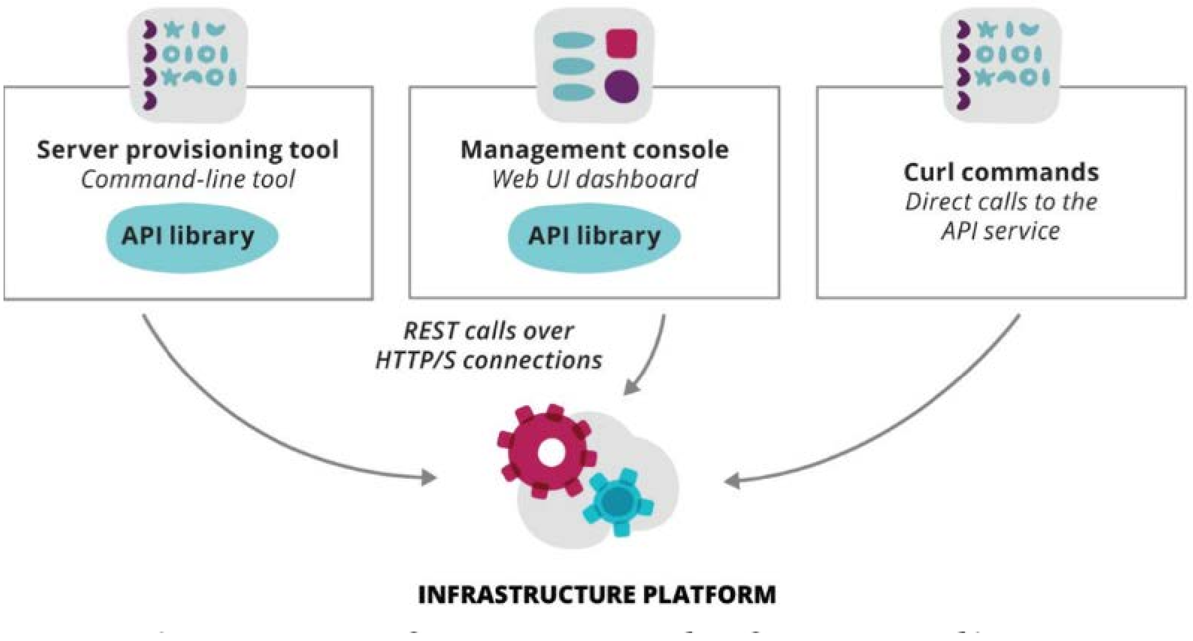


Figure 2-6 - Infrastructure platform API clients [4]

Bringing up new virtual environments or just new version of existing environments, simply requires the execution of a script. Versioning these scripts allows for proper configuration management. Releasing a new version of an environment simply requires checking out the proper script(s) and performing the necessary changes to the scripts – define the operating infrastructure, patching the OS, tweaking the application Server configuration or installing a new release of the application, and then checking the scripts back in as a new version of the environment.

Therefore, Infrastructure as Code becomes the stepping stone for the speed that DevOps demands and the management of multiple versions of multiple environments, to handle the CI builds.

2.6 Security

In this paragraph, the security requirements are briefly outlined, for a deeper discussion about the requirements see [1] and [6]. From a very high-level perspective, Security requirements include but are not limited to:

- Preserve, enforce, verify, Confidentiality, Integrity and Availability of Data and Applications
- Compliance to Security Standard/best practices
- Implement Security Controls
- Data classifications management
- Data retention policies management

2.6.1 Security Controls

IT security controls define safeguards and countermeasures that minimize, counteract or avoid IT security risks. NIST Special Publication 200 [6] provides the following table which identifies three security classes (technical, operational and management) and their associated family control types that will form part of the reference and technical architecture design.

While the controls are grouped into three categories, the underlying implementation of any control may blur these boundaries. For example, the operational controls within media protection may involve a technical control implementation of cryptography to achieve the control

Security Class	Class Description	Control Family Types
Technical	Controls that are implemented and executed by information systems primarily through security mechanisms contained in hardware, software and firmware components	• *Access Control* supports the ability to permit or deny user access to resources within the information system. • *Audit and Accountability* supports the ability to collect, analyze and store audit records associated with user operations performed within the information system. • *Identification and Authentication* supports the unique identification of users and the authentication of these users when trying to access information system resources. • *System and Communications Protection* supports the protection of the information system itself, as well as communications with and within the information system.
Operational	Controls include information system security controls that are primarily implemented through processes executed by people	• *Awareness and Training* supports the education of users with respect to the security of the information system. • *Configuration Management* supports the management of all components of the information system. • *Contingency Planning* supports the availability of the information system services in the event of component failure or disaster. • *Incident Response* supports the detection, response and reporting of security incidents within the information system. • *Maintenance* supports the maintenance of the information system to ensure its ongoing availability. • *Media Protection* supports the protection of information system media throughout their lifecycle. • *Physical and Environmental Protection* supports the control of physical access to information systems, as well as the protection of the ancillary equipment (e.g. power, air conditioning, wiring) used to support the information system. • *Personnel Security* supports the procedures required to ensure that all personnel who have access to the information system have the required authorizations, as well as the appropriate security screening levels. • *System and Information Integrity Controls* supports the protection of the integrity of the information system components and the data that it processes.

Management	Controls include security controls that focus on activities for the management of IT security and IT security risks	• *Security Assessment and Authorization* supports the security assessment and authorization of the information system. • *Planning* supports the security planning activities. • *Risk Assessment* supports the conduct of risk assessments and vulnerability scanning. • *System and Services Acquisition* supports the contracting of products and services required to support the implementation and operation of the information system.

Table 2-1 - IT Security Guidelines and Controls [6]

2.7 Business Continuity

High Availability (HA) protects the Datacenter service within and across the same geographic region, implementing different mechanisms like clustering, replication, storage synchronization, at the network, platform and storage layers.

Disaster Recovery (DR) refers to the protection of Datacenter service provided out of region and across DCs, using replication technologies (i.e. a platform-based replication and/or asynchronous storage-based replication).

Enterprise DCs must be deployed to provide both HA and DR solutions to the end systems that are deployed within.

Business continuity is a broader concept, which includes DR but actually refers to the ability of an organization to get the business (not only IT) back to full functionality after a disaster.

2.7.1 Disaster Recovery

Disaster recovery implementation is a matter of balancing the trade-off of the required compute resource investment versus the business loss. Disaster Recovery design implies having resources to bring up the systems with the required operations capacity level (compute/storage capacity), within a required timeframe (RTO – recovery time objective), with the needed business data (RPO – recovery point objective) to resume operations.

To protect the data, a mechanism must be implemented to extract the required data to a secure location. Typical mechanisms are either backups or storage replication. The parameter for the selection criteria is the RPO. To recover operations, you need compute resources either on standby or on a contract/process enabling compliance with RTO objectives. Low RTOs require equipment availability on premise, while high RTOs can absorb procurement time from the moment of disaster declaration. The operation capability levels drive the compute/storage capacity.

In multi-tenant environments, the Disaster Recovery design considerations must include the DR test process. This is key in shared infrastructure or services. Although the Disaster is a DC event, business units should regularly test and update their DR procedures for full functionality and correctness.

2.8 Other requirements

2.8.1 Government Regulations

Regulatory requirements usually depend on many external factors like the location of the datacenter, the type of data managed by the Enterprise, the context, and other external factors specific to the domain in which the Enterprise operates. This paragraph shows an example of the type of requirements that are not strictly business-related.

Usually the Regulatory requirements cover, at least

- Data protection: there are regulatory requirements covering who, when, how, (from) where the data is accessed and handled. One example is the General Data Protection Regulation [7], active in EMEA and applicable to data owned by EMEA citizens.
- Safety and Security of workplaces.
- Policies on Management of IT.
- Policies on providing electronic information under the Policy on Information Management
- Enable exchange of information under the Policy on Government Security

2.8.2 Regulatory and other compliance Requirements

Other policies, regulations, standards are usually applicable In the Enterprise context. As an example, Enterprise infrastructure handling Financial information could be required to

- Adopt standards required by the domain in which they operate (i.e. PCI-DSS [8] for the financial sectors)
- Adopt standards/regulations required from Auditing reporting to management and/or stakeholders (i.e. the board of shareholders)

2.8.3 Support to Legacy Systems

Sometimes some production workloads run on end of support/end of life/outdated systems for some of critical applications. Suppliers such as Hewlett-Packard, IBM, Microsoft, Tandem and Unisys still power important applications the Enterprise sector.

Many Enterprises rely on core systems built in the late 1970s and early 1980s. The transactions triggered in these systems pass through many legacy platforms which are hard to maintain but even harder to migrate.

Outdated platforms often survive because of the risk and cost of replacing them. The time and cost involved in system testing and the prospect of a massive end-user retraining program can prove intimidating too.

2.8.4 Restrictions on Encryption

In some cases, restriction apply on what could be encrypted, when, and where. These restrictions may vary; however, they could be summarized in three categories

1. Restrictions on use of cryptography (at rest, in transport, at all)
2. Restrictions on the export of Cryptography algorithms/devices
3. Restrictions on the import of Cryptography Algorithms/devices

2.8.4.1 Use of Cryptography

In some countries the use of cryptography is restricted, or anyway subjected to regulations. There are different scenarios, however from the Enterprise perspective it is important to know if a location chosen to host a datacenter restricts the use of encryption and/or if data flows coming from/going to a specific site must be encrypted using specific algorithm or not encrypted at all.

2.8.4.2 Import of Cryptography

Some countries may wish to restrict import of cryptography technologies for several reasons, as an example:

- Imported cryptography may have backdoors or security holes exposing sensitive data to the attacker organization; therefore, the use of cryptography is restricted to what is approved and verified by the regulation.
- Users can anonymously communicate, preventing any external party, including Law enforcing organizations, from monitoring them.

As stated in the previous paragraph, from an enterprise perspective, if such regulations exist in a datacenter location, they must be considered beforehand.

2.8.4.3 Export of Cryptography

Some countries have regulated the export of cryptography for national security reasons. As an example, as late as 1992, cryptography was on the U.S. Munitions List as an Auxiliary Military Equipment [9]

Due to the enormous impact of cryptanalysis in World War II, these governments saw the military value in denying current and potential enemies access to cryptographic systems.

Accordingly, regulations were introduced as part of munitions controls which required licenses to export cryptographic methods (and even their description); the regulations established that cryptography beyond a certain strength (defined by algorithm and length of key) would not be licensed for export except on a case-by-case basis. This policy was also adopted elsewhere for various reasons. Currently, many countries, notably those taking part in the Wassenaar Arrangement [10], have similar restrictions.

From an Enterprise, again, such restrictions, if applicable, must be considered to guarantee, for instance the feasibility of inter-region DR (see par. 4.6), and/or many other operations.

2.8.5 Restrictions on specific Data flows

Sometime, due to asymmetry in regulation policies, security concerns or other reasons, some specific data flows are simply not allowed to cross/rest specific sites and/or some users are requested to access the enterprise data only from specific locations/countries.

These requirements must be considered when designing DCN and related strategies (i.e. DR invocation

3 Datacenter network-specific Requirements

3.1 Abstract

Besides the high-level requirements presented in the previous chapter, there are also other technical/domain specific requirements that are amongst the main drivers of datacenter innovation and demand a new design approach.

This chapter starts presenting why the distributed networking model should be reconsidered in the context of datacenter networks, then introduces the current technological constraints driven by some of the mentioned technical requirements (i.e. MAC Address exhaustion, VLAN number space limitation). Other technical requirements are also presented, describing demands from the point of view of Software Development/innovation, Business Continuity, etc.

3.2 From Distributed to Centralized Networking in the Datacenter

3.2.1 Historical background

Until the first half of the 20th century, the biggest global telecommunication networks were the telephone networks. These networks were based on circuit-switching, essentially an evolution, both from the logical and design standpoints, of the telegraph networks. For the voice communication to happen, a circuit (connection) needed to be established; initially the circuit setup was provided through human intervention, later through signaling networks and protocols able to program the corresponding virtual circuit setup/teardown. Besides being circuit-switched, these global networks were centralized with a lot of users connected to large switching facilities.

In those years, during the Cuba Missile Crisis, a nuclear apocalypse seemed imminent between the two blocks and superpowers: The Western led by the United States (US) and the Communist block led by Union of Soviet Socialist Republics (USSR). Both the US and the USSR were building hair-trigger nuclear ballistic missile systems. Each country considered post-nuclear attack scenarios. US authorities considered ways to communicate in the event of a nuclear attack. How could "command and control network" continue? Paul Baran, a researcher at RAND, offered a solution to this question: *design a more robust communications network using "redundancy" and "digital" technology* [11]

Paul Baran, a Polish immigrant working in the 1960s as researcher at Rand Corporation, challenged the centralized design of the telecommunication infrastructure, saying that such approach would expose the network as an easy target in event of an enemy attack. In fact, the loss of a single switching center could potentially destroy the phone capability for a large section of the Country. Mr. Baran suggested solution to the issue was to send the voice in packets of data able to reach the destination and move independently from one other on the network and implement what he called a *Distributed Communication Network*. The core idea was that in case part of the path used for a specific communication were destroyed by the enemy, the communication itself was still working thanks to the ability of the voice packets to find their own way on different paths. In his paper, Mr. Baran demonstrated that the national voice network could still work even if fifty percent of its switches were destroyed. Baran proposed a network of unattended nodes routing information from one node to another to their final destinations. To exchange communications these nodes would use a protocol that Baran called "*hot-potato routing*”.

in the very same years, another cultural driver towards de-centralization was the need to share information and knowledge amongst different scientific entities: to enable this sharing, the experimental ARPANET program of the US Department of Defense (DoD) began to operate in 1969 as “*a mean to share data processing facilities at university, military bases and other DARPA-funded research sites*” [12]. ARPANET provided, amongst the others, two benefits responding to the two drivers mentioned above:

1. Enable the sharing of information, which was a key requirement in the Research world
2. Test, implement and improve the resiliency of the communication infrastructure, as envisioned by Baran.

This decentralized, connectionless network grew over the years and eventually became the Internet we know today. For decades after the born of the ARPANET, professionals fought wars around the advantages of connection-based vs. connectionless architectures and centralized vs. distributed architectures. The explosive growth of the Internet in the 1990s finally concluded these discussions: Internet, and its protocols, started to dominate, and their architecture was clearly connectionless, distributed. Older connection-oriented protocols like Frame relay seemed fated to the oblivion. All centralized designs were considered too vulnerable, not enough resilient. Even Asynchronous Transfer Mode (ATM) would eventually surrendered to the Internet that managed to handle the same throughput once conceivable only using ATM.

3.2.2 Distributed vs centralized networking in the datacenter

As its core, the Internet was connectionless and distributed: during its first cycle of expansion, the World Wide Web gave birth to ever-growing facilities housing large numbers of computers providing compute and storage capabilities. These facilities were protected against external factors as much as possible by being situated in disaster-unlikely locations, with redundant utilities and capacity.

Because of the large numbers of servers, these facilities were physically organized into structured sets of server racks. Over time the need for efficiency increased the density of these structures and pushed the adoption of server blades located in packed racks, instead of individual servers installed in loosely populated cabinets. Racks of compute-servers were hierarchically structured with Top-of-Rack (ToR) switches providing network both intra-rack and inter-rack (see Figure 3-1)

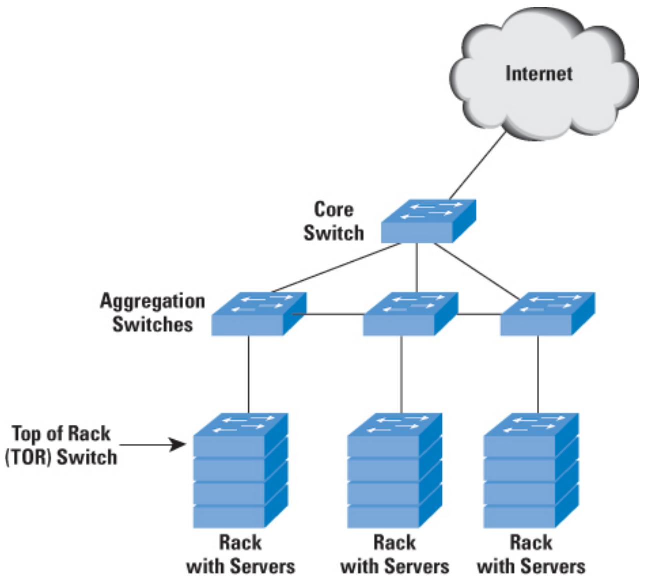


Figure 3-1 – ToR Switches organization

During this process, the total numbers of servers in the datacenter raised quickly. Modern Datacenters are currently able to accommodate over 120,000 physical servers, each one of them able to host more than twenty virtual machines (VMs) and/or 10 times more containers. This implies that the internal network in these datacenters would interconnect 2.4M VMs or 24M containers. This massive number of endpoints will need to communicate with each other via a set of protocols and networking devices in a highly controlled environment (the datacenter) where the type, amount and source of changes are barely comparable to those for which those protocols were designed. As explained in the previous paragraph, both the protocols and the networking devices were meant to work over a large, disparate geographical area with unreliable communications links. Obviously, a network where hundreds of thousands of IP endpoints located side by side in an isolated bubble with highly reliable communications links is a completely different scenario from the Internet (let alone ARPANET and before that the *Distributed Communication Network* imagined by Mr. Baran). Even though survivability, resiliency and availability in case of lost communications were not highly important in the datacenters, a huge amount of complexity designed to meet exactly those goals was making datacenter operations untenable.

3.2.3 The need of a new approach to network management

Besides the obvious difference in the stability of the network topology, and some technological constraints presented in the next paragraph, the other major difference between the Internet model and the current DCN is in the number of devices, links and interconnections: the massive number of these features creates a completely new network management challenge.

Management Networks designed for carrier public networks or large corporate intranets simply cannot scale to the size of a modern DCN. A new network management paradigm was needed.

3.2.4 The shift in datacenter traffic pattern: from North-South to East-West

The east-west traffic in the datacenter is composed of flows sent by one host in a datacenter to another host in that same datacenter, as an example see Figure 3-2

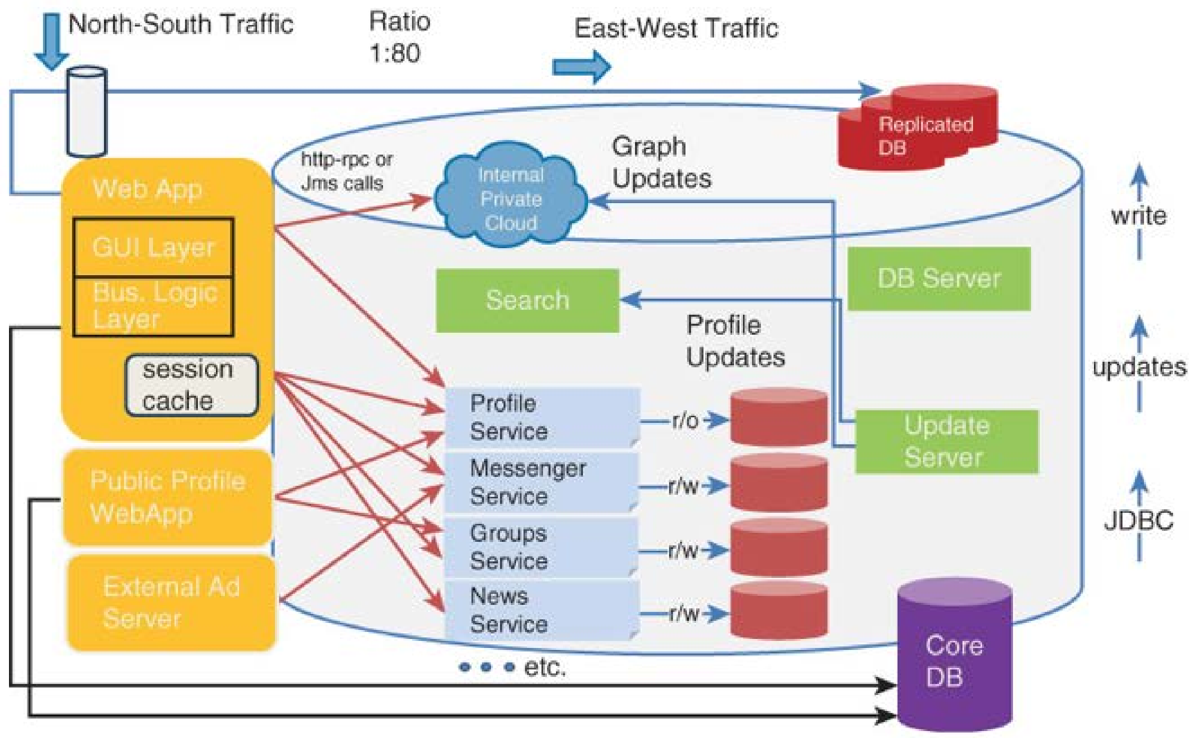


Figure 3-2 - East-west Dataflows in a social-network Application

Similarly, North-South traffic is the one arriving (leaving) the datacenter from (to) the outside world / end-users. As an example, a web browser's query to a search engine might be processed by a web server (North-South) which, before responding to the user (North-South), needs to retrieve data from one, often more, backend servers in the same datacenter (East-West). Quite often some of these flows are called *elephant flows* due to their sizable nature (compared to *mouse flows*, typically associated to north-south). These flows are characterized by being of relatively long duration yet having a discrete beginning and end. "*The protocols designed to achieve robustness in the geographically dispersed wide-area Internet today require that routers spend more than thirty percent of their CPU cycles [6] rediscovering and recalculating routes for a network topology in the datacenter that is highly static and only changed under strict centralized control* " [13]. While datacenters exist to support interaction with the outside world, studies and the previously mentioned high-level requirements imply that the predominant traffic in current datacenters is East-West.

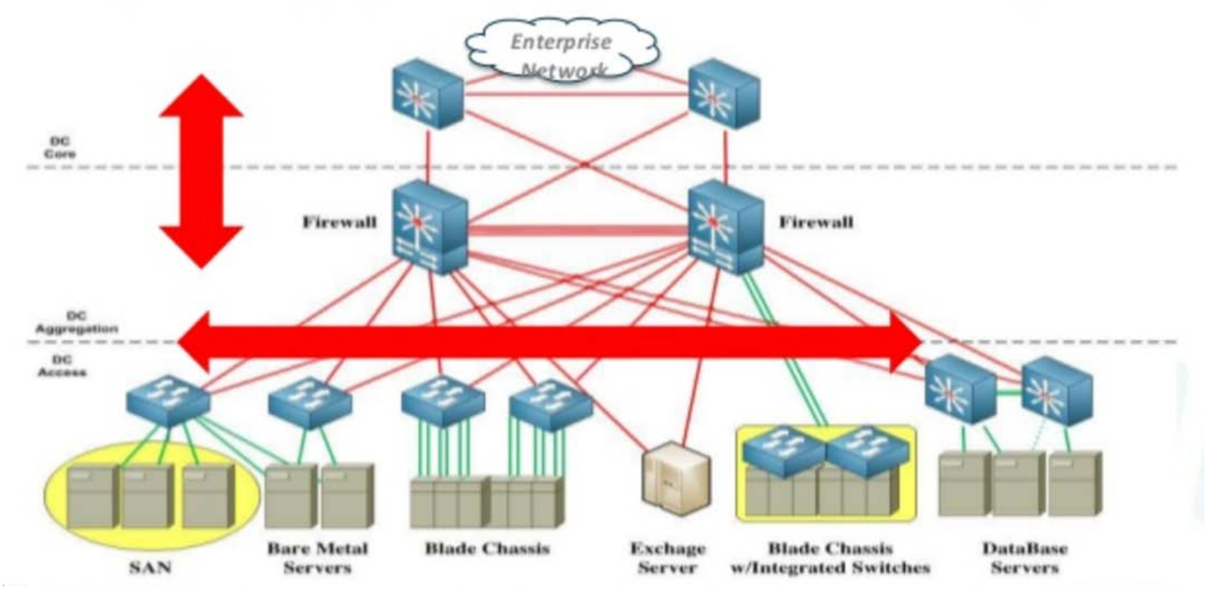


Figure 3-3 - Datacenter traffic flow shift: from North-South to East-West

The complex, distributed protocols and network infrastructure grown in traditional network switches to provide just the de-centralized survivability envisioned by Mr. Baran for the WANs of the past do not easily facilitate the increasing preponderance of East-West traffic. The mega-datacenters discussed in this section in fact are different from prior networks in many different core aspects:

- stability of the topology,
- traffic patterns
- pure scale.

Moreover, as shown in the previous paragraph, the services delivered by these datacenters demand frequent reconfiguration and require a level of agility not needed before. Traditional networking approach are simply unable to scale to the levels being required by the modern datacenter.

3.2.5 Centralize the control plane

A massive amount of unneeded control protocol occupies the control plane of the DCN:

- as already mentioned, in large scale datacenter networks thirty percent of the router's control plane capacity today is spent tracking network topology.
- The Internet-family protocols were also designed to handle routers joining or leaving the network.

These conditions simply do not represent the scenario of a modern DCN where there is virtually no unscheduled downtime. The base network is centrally provisioned even though there are links and nodes in the datacenter that occasionally might fail. In a datacenter, the endpoints simply do not magically appear or move: most of the time they change when the central orchestration software dictates so.

Therefore, most changes are done programmatically and intentionally, and this reduces the benefits of autonomous and distributed protocols, and the scale of the network is much larger than these protocols were originally designed for, requiring unacceptably long convergence times. To summarize:

1. The conditions for which the distributed protocols were designed to work simply do not exist in a modern DCN.
2. Using distributed protocols in this way overcomplicates the network management to an extent that is entirely avoidable. A simpler approach would be to remove the control plane from the datacenter switches and centralize it to program the forwarding tables of all the datacenter switches. The simplicity of this concept arises from the facts that:
    - The topology inside the datacenter is stable and under local control.
    - The knowledge of the topology is now centralized and controlled by the same administrators / orchestration system.
    - When a node or a link fails, this knowledge could be used to quickly provision a consistent set of forwarding tables.

3.3 Inadequacies in Today's Datacenter Networks

Current networking technologies seem to be unable to cope with some of the requirements presented in the previous chapter, especially in the context of Large Enterprises and/or mega datacenters. Several issues cause this problem:

- Technical inability to scale to the size of the modern mega-datacenters
- The cost of networking, compared to other equipment (storage, compute) in the datacenter, seems to be elevated.
- The innovation rate in the networking domain seems to be disconnected from the rate of innovation in other infrastructure areas like compute and storage.

The potential to easily switch on and off VMs is a radical departure from the physical world that network managers have traditionally faced. Networking protocols were coupled with physical ports, but this is not the case anymore: the dynamic nature and sheer number of VMs in the datacenter have placed demands on the capacity of network components that were earlier thought to be safe from such pressures. These areas include:

- MAC Address Table Size
- Number of VLANs
- Spanning Tree

3.3.1 MAC Address Explosion

Switches and routers forward frames and packets using the MAC Address Table, implemented in hardware, so its size has a physical limit. Equipment manufacturers define the maximum number of in the MAC table based on two different, opposites, drivers:

- control the overall product cost
- store an amount of entries *adequate* to the demand of the network.

In the past, the maximum number of MAC addresses that would need to be in the MAC Address Table at any given time was partly attributed to physical limitations of datacenters and to the maximum number of servers that would-be part of a single layer two / broadcast domain. The physical layout of the network also affected it. Moreover, physically separated layer two domains remained logically separated. Nowadays, network virtualization and the use of Ethernet technology across WANs, are stretching geographically the layer two networks as never before. Server virtualization has increased the number of servers in a single broadcast domain. The need of numerous virtual NICs on each virtual server, increases further the number of MAC addresses.

In the event of a MAC table miss, conventional L2 switches flood the frame causing the miss on all ports except the one(s) receiving the frame. In normal mode of operations, when receiving that L2 frame, the destination will respond. When the switch sees the response, it learns the port on which that MAC address was seen and populates its MAC table accordingly. This mechanism (known as *backward learning*) works well unless the MAC table is full, in which case the switch cannot learn the MAC address and must flood (transmit the frame on all the ports excluding the receiving one) all the frames sent to the destination causing the table miss. This is a very wasteful use of bandwidth and has significant negative performance impact. This problem is aggravated in the DCN core and/or in case of traditional non-overlay designs, where the pressure on the MAC address tables is intense.

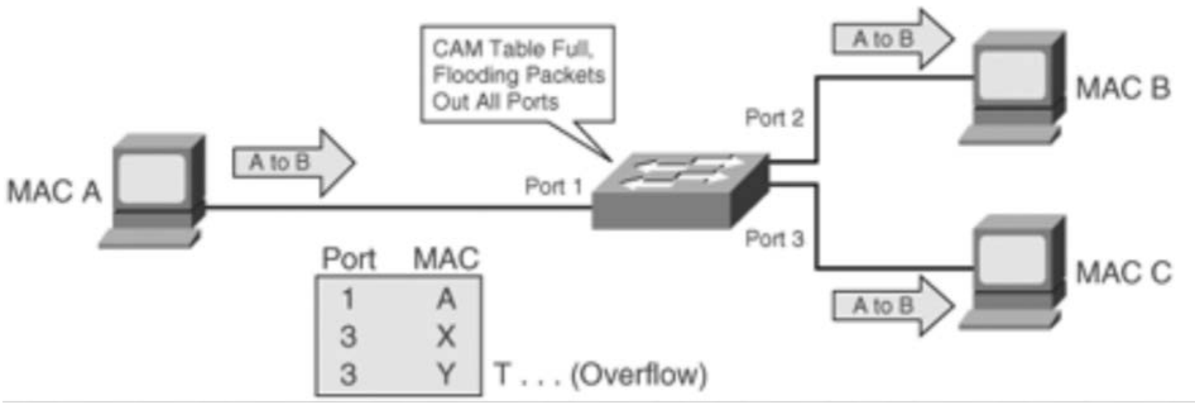


Figure 3-4 - MAC Address Table Overflow

In modern layer two networks VLANs are used extensively, and sometimes this is (wrongly) perceived as a way of mitigating the problem mentioned above: when a VLAN-tagged frame fails its match, it is flooded out only to all ports on that VLAN, in some way reducing the inefficiency of flooding. However, hosts might belong to multiple VLANs (i.e. hypervisors, network appliances, security appliances, etc.) and in this case, they will have one MAC entry for each VLAN interface (sub-interface), further increasing the chances of MAC address table overflow.

3.3.2 Number of VLANs

When the IEEE 802.1 working group standardized the 802.1Q [14] extension to the definition of local area networks, they thought that 4094 VLANs were enough, therefore they allocated 12 bits to store the VLAN ID. The IEEE 802.1Q Tag for VLANs shown in Figure 3-5 supports $2^{12} - 2$ (4094) VLANs (all zeros and all ones are reserved). When the 802.1Q tag was introduced (in the late 1990s), networks were smaller and the demand for multiple virtual broadcast domains sharing the same physical network was very small.

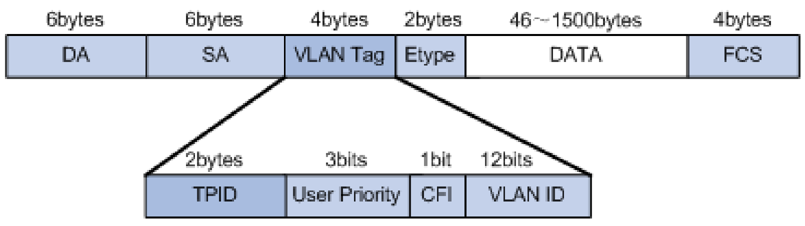


Figure 3-5 - 12 bits reserved for the VLAN ID (4094 VLANs)

The introduction, and the growth, of datacenters triggered the need to segregate traffic and enforce separation between the various tenants. 802.1Q/VLAN tagging is the technology responsible for providing this separation. If datacenters remained single-tenant, then 4094 VLANs seemed enough: expanding this field would have required that different tables in memory and in the ASICs had to be large enough to accommodate the new size.

However, with datacenters continuing to expand, especially with multitenancy and server virtualization, this number of required VLANs easily exceeds 4094. Sharing resources between tenants quickly becomes difficult when there are no more available VLANs. Since the size of the VLAN tag is hardcoded in CAMs, TCAMs, and in general network equipment has been built depending on that specific size, expanding this size to accommodate more VLANs is nontrivial and might have a massive impact on the market (not only switches but also NICs, OS drivers, etc. would be affected). Therefore, another solution is needed to overcome this limit[1].

3.3.3 Spanning Tree

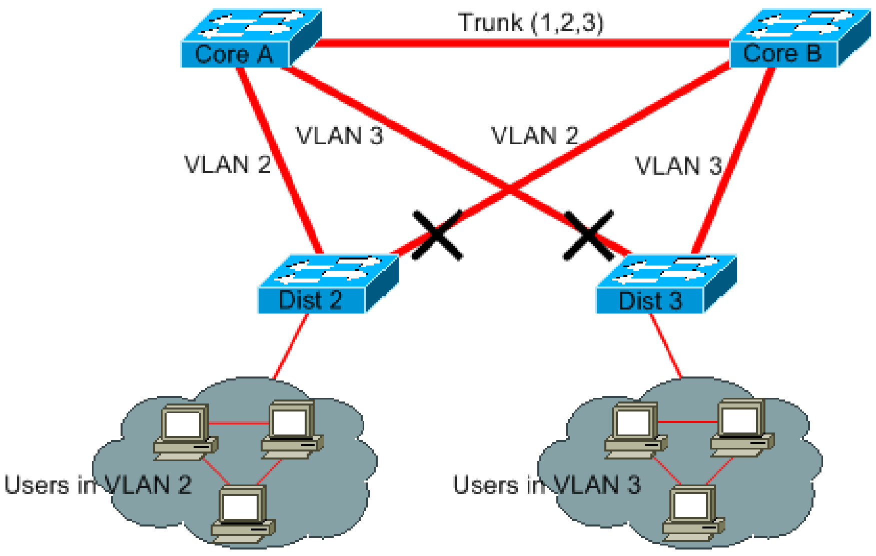


Figure 3-6 - Redundant links blocking by STP

[1] Another approach is the use of overlay techniques (see par. 7.6)

In the past, 802.1X bridges were built as transparent devices able to forward frames from one broadcast domain to another, without explicit configuration of forwarding tables. Bridges learned the forwarding tables by observing the traffic: they were able to determine if the network contained loops, break those loops and prevent broadcast storms from bringing the network down. Switches and bridges accomplished these goals cooperatively by building a spanning able to identify and enforce a loop-free network topology.

This tree was initially calculated using the Spanning Tree Protocol (STP) [15]. The early implementations of STP would require dozens of seconds to converge after a change on the network had taken place. Over time, through improvements to STP, the time needed for convergence has been reduced. Regardless these convergence improvements, by design, STP

- leaves completely functional links unused (STP "*blocked*" ports)
- enforces forwarding of frames to the root bridge, which is not always the optimal path.

Datacenters need to take advantage of the most efficient path between any two nodes without imposing a pre-defined hierarchy in the traffic patterns, and no using multiple 10G is a price too high in modern DCN. Moreover, datacenter virtualization has increased the frequency of changes and disruptions, thus requiring convergence to occur more often, this putting additional pressure to the already inefficiency of STP in the datacenter. To put it simply: *STP was not built for the modern DCN.*

3.4 Agility with Stability

Changes in modern datacenters are made much quicker than in the past. To stay in step with servers and storage, networks must be able to transform and change with at least the same speed. Automation, orchestration and agility are key requirements driven by the overall virtualization of resources and they are becoming a major requirement for the entire infrastructure, networking included.

On the other hand, a mistake in a network change would potentially impact the entire datacenter, so quite often in legacy network significant changes require days or week to be implemented: the amount of time needed to understand the potential impact of a change increase quickly with the complexity of the network, making the change/impact analyses process the most time-consuming part of the transformation.

The two factors mentioned above, need for quick changes and impact assessment, drive another transformation in the datacenter networks: Agility with Stability.

3.5 Failure Recovery

Due to their size and scale, recovering modern datacenters from failure is a difficult goal to be achieved, and the growing scale of datacenters further magnifies the impact of wrong recovery decisions/design.

Predictability and agility are two of the most important requirements when it comes to failure recovery. The distributed intelligence model of today's network failure recovery may result in erratic behavior. It is advisable that the infrastructure, and the network, move to a consistent, deterministic, stable state given a failure.

3.6 Dynamic infrastructure Delivery Model

The requirements of an improved efficiency, efficacy, security and all the other high-level requirements described in the previous paragraphs and chapter push for a better utilization of resources, more compact, resilient, and scalable.

These requirements have been implemented in other infrastructure domains (i.e. compute power, storage) by a massive use of resource (compute, storage, etc.) virtualization and resource-sharing, in a secure way, between different users. The same principle is now being applied to the datacenter networks, and to the datacenter concept itself with the colocation, cloud and CNF concepts. Essentially these requirements push for an infrastructure (compute, storage, network, etc.) able to dynamically adapt and transform in a secure and reliable way the datacenter minimizing the operating costs and maximizing the efficiency and efficacy.

3.6.1 Cloud computing Classifications

The NIST Cloud Computing Program defines Cloud computing as *"a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [16]

Just starting from the simple definition above, it should be clear why, from an enterprise perspective, adopting the cloud makes perfect sense: it basically moves further the already discussed virtualization, virtualizing not (only) the infrastructure inside the datacenter, but the datacenter itself (IaaS) and/or all the backend applications hosted (PaaS) and ultimately the services provided to the end-user (SaaS)

Cloud Computing could be classified in many ways, but essentially two classifications are quite popular nowadays (as an example, see [17]):

1. Classification based the service delivery model: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as Service (SaaS).
2. Classification based on the infrastructure deployment model: public, private and hybrid cloud.

3.6.2 Service delivery Models

Figure 3-7 - the three main Cloud service delivery models

3.6.2.1 IaaS

Infrastructure as a service (IaaS) is a cloud solution providing the basic infrastructure able to host virtual servers and/or virtual machines. In this type of solution, the Cloud team and/or the orchestrator platform, typically has direct access to the operating systems of the nodes hosted in the cloud, installs the operating systems and delegates choices on the use of the infrastructure to the users. Amazon Web Services (AWS), and Microsoft Azure (Private Peering) are two popular offering for modern IaaS solutions.

3.6.2.2 PaaS

With PaaS infrastructure, users can access the applications (i.e. a database server) without needing to have direct access and/or manage the underlying platform and operating system. Customers of PaaS manage only the application(s) they deploy and do not need to take care of the underlying infrastructure providing these applications. Microsoft Azure (Public Peering) is a popular offering for modern IaaS solutions.

3.6.2.3 SaaS

SaaS is another common, and quite popular solution which basically move to the latest stage the virtualization. Customers do not manage the infrastructure, not even the application, they just consume the services provided by the platform (for example, the MS-Office suite for Office365). SaaS users simply access the software over (typically) an Internet connection and do not need to manage, install, maintain the software/application or manage it on their local computer. Maintenance, upgrades, patches are completely managed by the platform provider.

3.6.3 Deployment Models

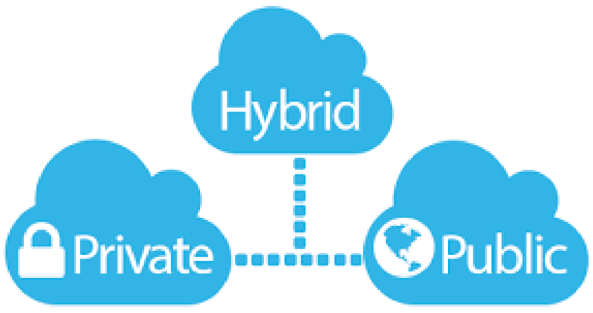


Figure 3-8 - Cloud Deployment Models

The main advantage of a cloud-hosted service is that the needed resources are pooled together and used as shared entities, and they can be upgraded, made accessible, managed and scaled regardless of any physical boundaries. Thus, cloud-based hosting could be available through basic services on small number of servers for limited group of users or through publicly offered services deployed in massive datacenters. This diversity drives different deployment and sharing models, that, as an example, could be based on the targeted users, type of application, type of service, and scope of access that the deployment should need. Three general deployment categories could be identified:

- Public Cloud
- Private Cloud
- Hybrid Cloud

3.6.3.1 Public Cloud

Public Cloud solutions offer small and medium companies the opportunity to use virtualization through public resources. These resources, offered by CSPs, can host computing power, storage, networks, applications, databases on a cloud infrastructure maintained, managed, and operated by the CSP. Essentially the choice between a CSP-based public cloud versus maintaining a private cloud is analogous to renting versus buying a car. With public cloud, the overhead of maintaining the hardware infrastructure and the operational costs to manage them, are completely left to the CSP. The user is just a tenant in the CSP's cloud and shares the resources with other tenants. Other advantages offered by the public cloud are usually a broader network access, vendor backed service guarantee and data backup. When overhead must be kept low, a public cloud is usually the best option.

Despite the mentioned advantages, public clouds present some level of hidden/hard to predict running costs and vulnerabilities, and from an enterprise perspective, pose some challenges on the overall security model.

Examples of public clouds include Amazon Web Services (AWS), Google Cloud, Microsoft Azure, Rackspace, and many others.

3.6.3.2 Private Cloud

A private cloud could be described as a virtualized infrastructure situated in one or multiple locations, directly or indirectly managed, operated, and utilized by the organization owning the infrastructure itself. This approach offers complete isolation, independence, and a security model easily compliant with the organization’s policies, but it also brings in the overhead necessary to acquire, manage, and operate the infrastructure. Large enterprises may find this model desirable, where the advantages of the cloud outweigh the overhead cost. Enterprises dealing with sensible (i.e. financial, medical, or military) data may require higher levels of security and isolation and avoid putting their data and compute in a publicly accessible domain. While private clouds offer much more control over the level of isolation, they do not have the cost benefit of offloading the management, maintenance, upgrade, and deployment of the infrastructure.

Several platforms available today offer implementation and management of privately hosted clouds. The choice between these could be based on licensing costs, roadmaps, availability of support, and ease of use. Some of the more commonly used platforms for deploying private clouds are VMware’s vSphere and OpenStack.

3.6.3.3 Hybrid Cloud

Hybrid clouds offer the best of both public and private offering: they can be setup to use resources hosted by Cloud Service Providers (CSP) for those services running in the cloud, whilst continuing to use private cloud services for more delicate applications. This implies that the hybrid cloud scales much better in comparison to private cloud architectures, yet it still allows management and control by keeping some portions of data and applications in the private cloud.

By using the hybrid cloud model, boundaries and differences are managed and defined basically by the cloud architects. The end-user sees the hybrid cloud as one environment where applications run, data is stored, and networks configured. Most private cloud management and deployment tools can interact with public clouds using APIs and present the hybrid cloud model to the user.

3.6.4 Containerization and Micro-segmentation

Containers virtualize the OS, splitting it up into virtual compartments to run isolated applications or workloads. This allows to run code in smaller, easily transportable pieces that can be hosted anywhere the host OS is running. The main appeal of containers is that they speed up the development process allowing applications to be distributed across the cloud and moved around in a virtual fashion. This has made them a key component of the DevOps approach.

μ-segmentation is a security technique that enables fine-grained security policies to be assigned to datacenter applications, down to the workload/container level. This approach enables security models to be deployed deep inside a datacenter, using a virtualized, software-only approach, possibly intent-based.

The combination of μ-segmentation and containerization allows the secure provisioning of virtualized, easy to move, containerized workloads in different environments.

3.6.5 Multitenancy

Consolidation and virtualization combined trigger a more efficient utilization of resources, which drives the need to host more clients “sharing” the same physical resources in the same space. These clients (also known as “tenants”) must be kept segregated one from the other and they must be able to utilize their chunk of resources, being them storage, compute power, network bandwidth and latency, efficiently and in compliance with the corresponding SLA.

In these multi-tenancy environments, it is vital to keep separated the resources belonging to each tenant. For storage, as an example, this implies implement proper segregation strategy in the vSAN by ACLs and other mechanisms as such. For networks, this implies for instance segregate the traffic using technologies able to virtualize the network (i.e. VLANs, VRFs, vDCs, dedicated Tenants) able to guarantee that packets from different tenants are kept separated one from another, each one with the amount of bandwidth and with the latency compliant with the SLA.

When the number of tenants in the datacenter was small, it was still possible to arrange bandwidth and latency per tenant, typically using QoS/traffic shaping strategies, using scripts or other manual/low level techniques.

Nevertheless, in a modern datacenter, as said, the number of tenants is continuously growing, and spans from tents (small datacenter) to hundreds (big datacenter) or thousands (cloud) or hundreds of thousands (large cloud providers). In this scenario, a more dynamic and reliable approach is required.

3.6.6 Resource-location driven network topology

Another constraint on modern network designs is that the physical location of the network services largely determines the topology of the network and have historically restricted the placement of the workloads.

End-to-end application traffic flows are often required to traverse various network service functions such as IDS/IPS, firewalls, WAN Optimizers, and load balancers along a predetermined path. This results in a static chains of network services that cannot flexibly cope with dynamic user and service requirements

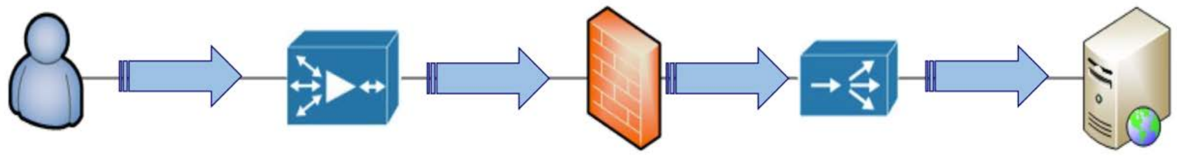


Figure 3-9 - Traditional Static Network Service path

Modern datacenters and cloud networks, however, need to be able to provide the required services and policies independently from where the workloads are placed and regardless of the availability of network service functions.

3.6.7 Infrastructure Automation and Orchestration

The need to adapt and transform triggers per se a requirement of an automation and orchestration platform able to dynamically plan, schedule, control, implement and maintain the infrastructure. The orchestration platform essentially must be able to communicate with all the infrastructure delivery components (Storage, computing, network) and to trigger the necessary changes depending on external factors (i.e. traffic load, change in operating costs during the time of the day, etc.).

It is worth noting that the automation and orchestration platform drives not only the local infrastructure but could also direct changes in delocalized, distributed centers (i.e. cloud services).

3.6.8 Zero Touch Provisioning (ZTP)

With Zero Touch Provisioning (ZTP) the infrastructure allows automatic provisioning and configuration of devices, eliminating most of the manual work required to add them to a network. ZTP reduces the provisioning time and the chances of misconfigurations

3.7 Carrier neutrality

An enterprise datacenter should allow interconnection between many co-location and interconnection providers. Carrier-neutral datacenters are not tied to any service provider (telecommunications, ISP, or other) and provide diversity and flexibility.

According to a study made by Gartner "*Carrier neutrality is an essential factor to look for when outsourcing the datacenter or seeking interconnection services, as in the end, it provides the benefit of both cost-efficiency and more connectivity than the Enterprise could find anywhere else.*" [18]

There are (at least) 4 important reasons to select a carrier neutral datacenter over a *carrier specific datacenter:*

1. **Redundancy** – Relying on only one carrier to connect to the Internet, or to the CSP. has inherent risks. IT best practices generally mandate that a minimum of two carriers be used to connect critical systems to the Internet just in case one fails. Choices of alternative carriers in a carrier specific facility are likely to be severely limited.
2. **Lower Pricing** – competition drives down prices: the availability of multiple carriers will enable the freedom of choosing between different carriers.
3. **Flexibility** – Each carrier's network is unique. Fiber and cables cover different routes and the equipment can provide different benefits. A datacenter with multiple carriers gives an intrinsic advantage of routing and feature sets offered by competing carriers at any time.
4. **Portability** – changing from one carrier to another will be inherently easier, due to the abundancy of carriers terminated in the facility [19]

SECTION II- DATACENTER NETWORK SPECIFICATIONS AND DESIGN TOOLS

4 Design Principles

4.1 Abstract

This chapter presents the main principles that drive the design of an Enterprise Datacenter Network. These principles descend directly or indirectly from the high-level requirements described in the previous Section and they will be applied on all the high-level designs presented in Section III. An introduction to network architecture, providing the theoretical foundation for the mentioned principles and specifications can be found in [20].

4.2 Architecture principles

The datacenter network must be designed keeping in mind the need to enable business and increase stability, the ability to change (agility), adapt and transform as fast as possible. The four principles of IT Architecture could be identified as:

- Business enablement
- Stability increase
- Cost reduction
- Security

Those stated above are high-level principles useful in evaluating/assessing if a specific design is fit for purpose. In the context of this paper (Enterprise Datacenter networks) the principles above and the requirements identified in the previous section, mandate the following features to be available in the design:

- Predictable, hyper-scalable, non-blocking network topology model
- Network and network functions virtualization, the centralization of the control plane, and the Services function chaining (DCN Virtualization)
- Support of Dynamic resource allocation, orchestration (Software-Defined Datacenter network)
- Well defined, tested, simple, Datacenter and Business Continuity model

4.3 A model for the Datacenter Network Topology

Most of the modern DCN architectures adopt a topology based on the Clos Network (see Chap. 5). Modern datacenter networks are comprised of top-of-rack switches and core switches:

- The top of rack (ToR) switches are the leaf switches and they are attached to the core/spine switches.
- The leaf switches are not connected to each other and the spine switches only connect to the leaf switches (or an upstream core device).

The current trend is to adopt Clos topologies using the Point of Delivery (Pod) concept as building block. The Pod is an atomic unit of compute, network and storage. It is designed as a unit, deployed as a unit, automated as a unit and retired as a unit.

4.4 The three pillars of a modern DCN

As already mentioned in Section I, the modern datacenter requirements together with the technological constraints and the DCN specifications push towards a:

- Virtualized network model able to securely share different flows and tenants on the same physical infrastructure (**Network is virtualized**)
- Dynamic, elastic, simplified, open and centralized network control plane where most of the complexity is moved from the networking equipment to a centralized controller (**Network is Software-Defined**)
- Virtualized network functions (i.e. firewalling, load balancing, etc.) provided through combination of physical and virtual services and chained together (**Network Functions are virtualized and chained**)

The three pillars above are the foundation elements of the solutions that will be presented in the next chapters of this section.

4.5 Orchestration and dynamic resource allocation

The three pillars identified in the previous paragraph provide a Software-Defined Datacenter (or DCNaaS, Datacenter Network as a Service) where, through a set of API/northbound protocols, an orchestrator platform can dynamically trigger the changes, allocating, updating, releasing the resources, defining the policies on the network infrastructure.

4.6 Distributed Datacenter and Business Continuity Model

In this paper we assume that the Enterprise operates globally in four regions:

- Asia-Pacific (APAC)
- Europe, Middle-East Africa (EMEA)
- Latin America (LATAM)
- North America (NA)

with 2 Datacenters in each region, connected to each other through a dedicated high-speed low latency regional Datacenter Interconnect (DCI). The Enterprise owns/operates a Wide Area Network interconnecting Corporate users, partners, and other networks (i.e. Public Cloud, Internet) to the Datacenters. DR capability is required on a region-based (inter-region DR is not required) with residual risk handled by offline copy of the data on a different region.

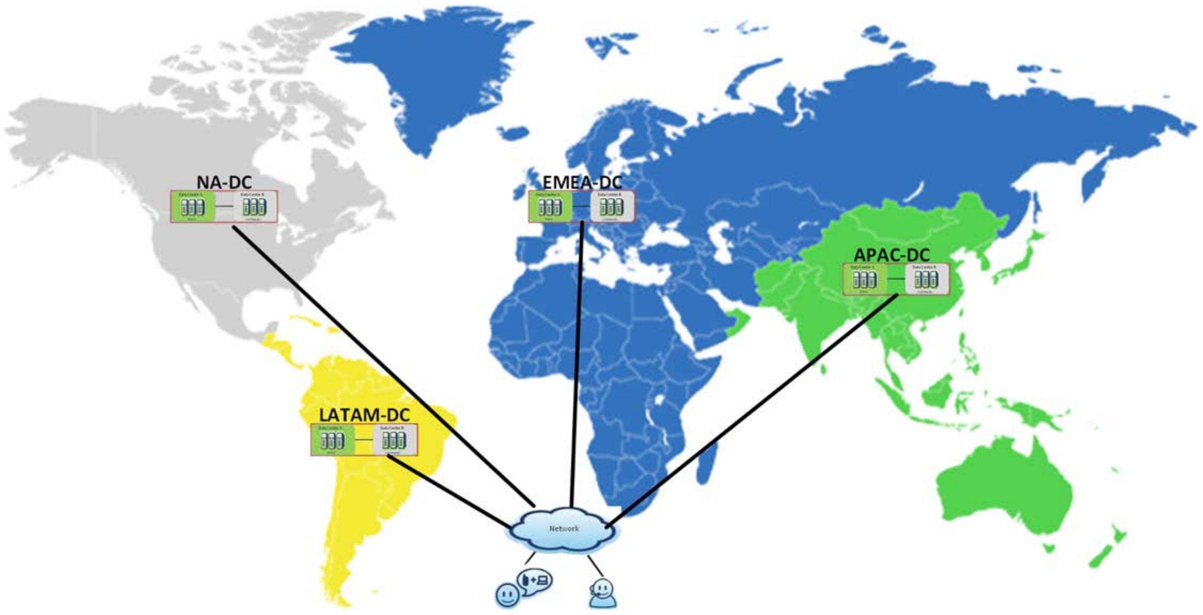


Figure 4-1 - Distributed Datacenter model and BC

Every region has two datacenters, the primary DC hosting PROD and SIT/DEV systems and the secondary for DR, DR Test and UAT. The two regional datacenters are, by design, very similar, provided that during real DR invocation the secondary datacenter must be able to host and handle all the PROD workloads running on the failed primary Datacenter (*Sacrificial DR model*). Inter-region DR is not required (region-based datacenter failure domains).

This paper focuses on the datacenter internal network, not the DCI link connecting the intra-region or inter-region datacenters, or the WAN network connecting the users.

5 Network Topologies for the DCN

5.1 Abstract

This chapter introduces a taxonomy for the datacenter networks, then an overview of the Clos Networks and POD-based designs is presented, and finally, using publicly available information, the network topologies of four massively scalable datacenters are briefly described.

Unfortunately, the details about the SDN implementations for most of the big companies are not public, however it is worth noting that the probable four biggest datacenter implementations (Amazon, Facebook, Google, Microsoft) share these design choices: Clos Pod-based design and a combination of SDN/NFV/NFC on the control and management plane.

For a more complete discussion on the theoretical foundations of this chapter see [21], [22] provides an introduction to different DCN topologies, and [23] presents a concise summary of the Clos theorem.

5.2 Datacenter network topologies

5.2.1 Taxonomy of DCN topologies

Several datacenter network topologies have been proposed by researchers, aimed at addressing many of the shortcomings of the current DCN. This paragraph introduces a taxonomy of several DCN topologies, then reviews their properties in terms of scale, performance and hardware redundancy. This paragraph is largely based on the work described in [22].

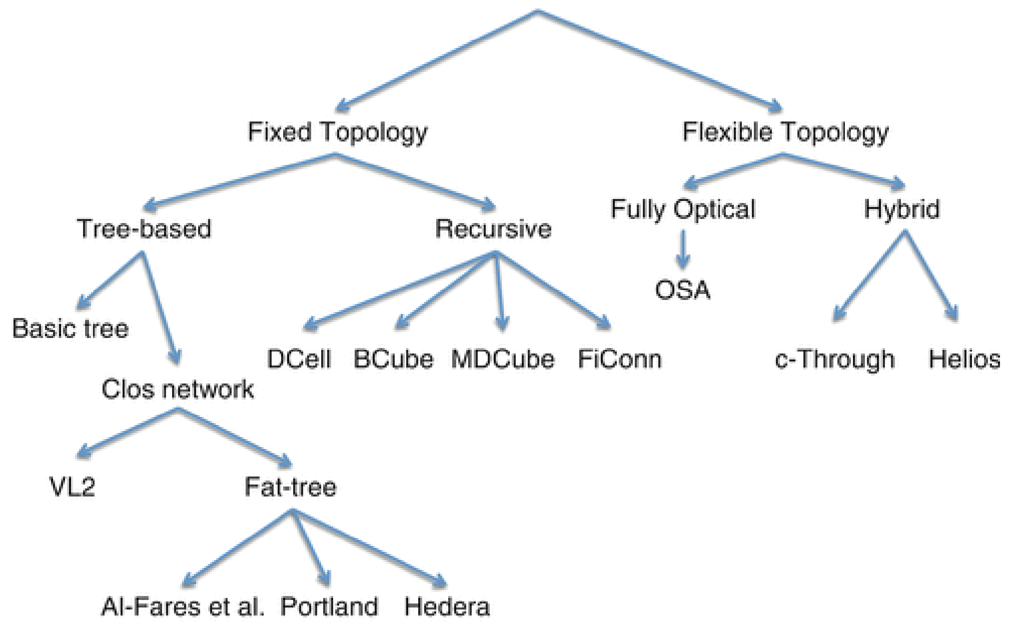


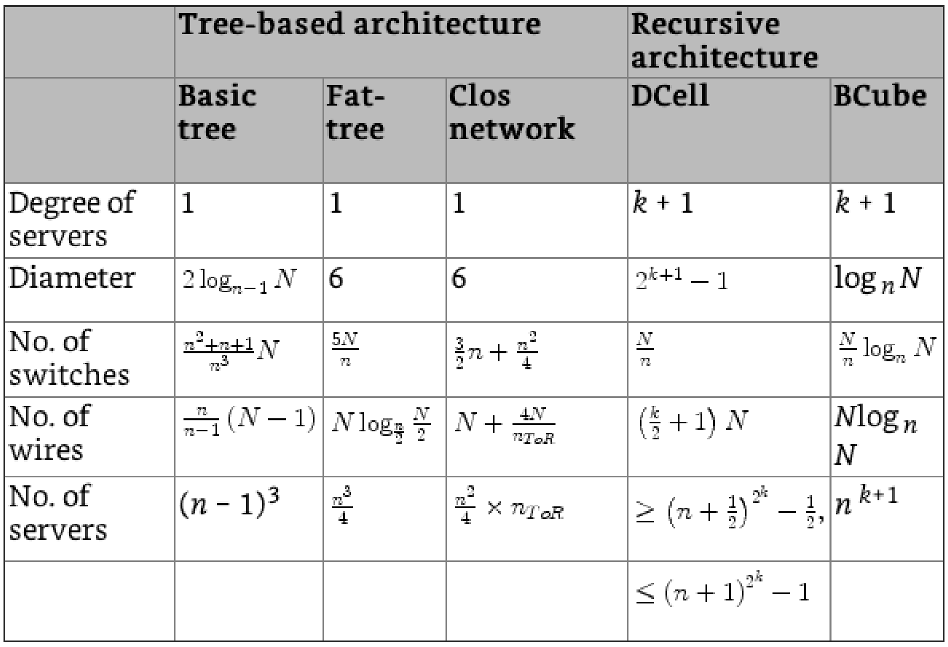
Figure 5-1 - Taxonomy of Datacenter network topologies (Fig. 3.1 in [22])

"DCNs can be classified as fixed-topology architectures and flexible-topology architectures, according to whether the network topology is fixed from the time it is deployed. Fixed-topology architectures based on Fat-trees include the architecture proposed by Al-Fares et al. [24]*, Portland and Heeder architectures* [25]*, Clos networks as represented by the VL2 architecture* [26]*, and Recursive topologies such as the DCell* [27] *and BCube architectures* [28]*. Flexible-topology architectures include c-Through* [29]*, Helios* [30] *and OSA* [31]*. Other noteworthy architectures include FiConn* [32]*, MDCube* [33]*, and CamCube* [34]*"* [22]. Figure 5-1 gives a taxonomy of the different datacenter network topologies.

Even though this paper focuses on folded Clos networks (aka spine-leaf topologies) which have become the de-facto standard for Enterprise Datacenter, the next two paragraphs (based on the work in [22]) present a brief comparison of different topologies.

5.2.2 Comparison of topologies

5.2.2.1 Comparison of Scale



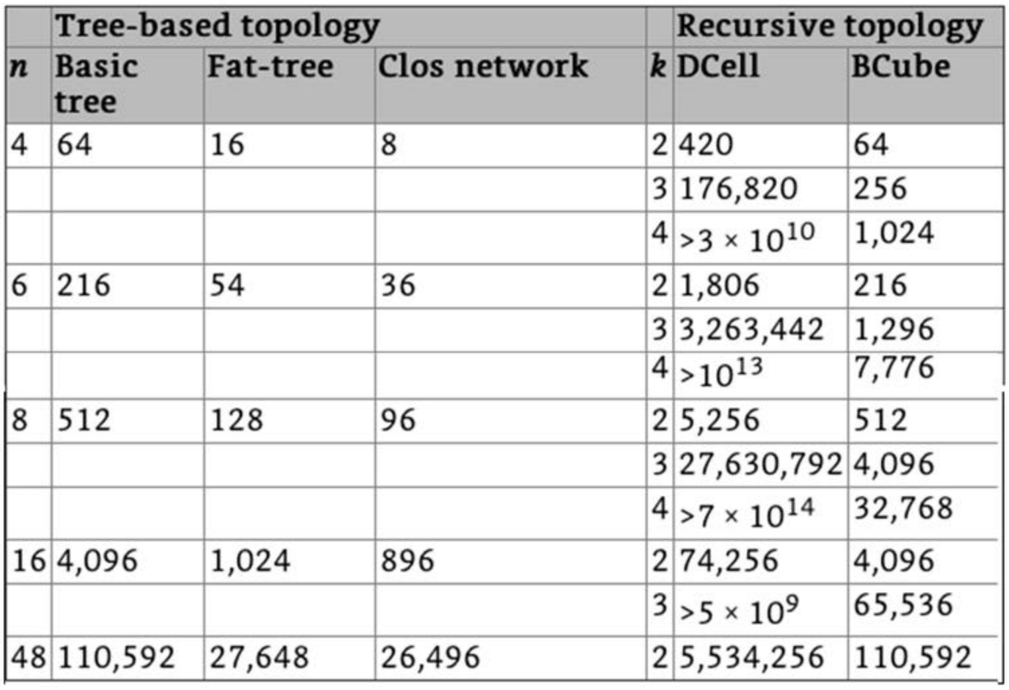
	Tree-based architecture			Recursive architecture	
	Basic tree	Fat-tree	Clos network	DCell	BCube
Degree of servers	1	1	1	$k + 1$	$k + 1$
Diameter	$2\log_{n-1} N$	6	6	$2^{k+1} - 1$	$\log_n N$
No. of switches	$\frac{n^2+n+1}{n^3} N$	$\frac{5N}{n}$	$\frac{3}{2}n + \frac{n^2}{4}$	$\frac{N}{n}$	$\frac{N}{n} \log_n N$
No. of wires	$\frac{n}{n-1}(N-1)$	$N \log_{\frac{n}{2}} \frac{N}{2}$	$N + \frac{4N}{n_{ToR}}$	$\left(\frac{k}{2}+1\right) N$	$N\log_n N$
No. of servers	$(n - 1)^3$	$\frac{n^3}{4}$	$\frac{n^2}{4} \times n_{ToR}$	$\geq \left(n+\frac{1}{2}\right)^{2^k} - \frac{1}{2},$	n^{k+1}
				$\leq (n+1)^{2^k} - 1$	

Table 5-1 - Summary of parameters (Tab. 3.1 in [22])

Table 5-1 presents a comparison of some of the parameters of the topologies introduced earlier. The flexible-topology architectures are not included. The following parameters are used when comparing the different topologies:

- Degree of the servers: The (average) number of network ports on the servers in the datacenter. For the tree-based topologies, only one port is needed on each server. However, in the recursive topologies, the degree of servers may vary according to the levels required.

- Diameter: The longest of the shortest paths between two servers in a datacenter. A smaller diameter leads to more effective routing, and lower transmission latency. In practice for most topologies, the diameter grows logarithmically with the number of servers.
- Number of Switches: It is assumed that all switches are the same in each topology. BCube uses the largest number of switches, which may lead to higher cost. The basic tree topology uses the fewest switches because there is little hardware redundancy in the architecture.
- Number of Wires: This metric shows the number of wires required when deploying a datacenter. BCube uses the most wires, followed by DCell. However, it should be noted that the number of wires only shows the wiring complexity; it does not show the accurate wiring cost, because not all the topologies use homogeneous wires. E.g., the typical bandwidth of a link between an aggregation switch and a core switch in fat-tree is 40/50 Gbps, while it is 10/25 Gbps on a link between a server and an edge switch.
- Number of Servers: All the metrics above are computed under the same number of servers (N), while this one shows the scalability of different topologies with the same n and k (for recursive topologies only). It is assumed in this row that the tree-based topologies use 3-level structure. Considering the data in Table 5-1, it is no doubt that DCell scales up much faster than other architectures for the number of servers in DCell grows double-exponentially with k.



	Tree-based topology				**Recursive topology**	
n	**Basic tree**	**Fat-tree**	**Clos network**	***k***	**DCell**	**BCube**
4	64	16	8	2	420	64
				3	176,820	256
				4	$>3 \times 10^{10}$	1,024
6	216	54	36	2	1,806	216
				3	3,263,442	1,296
				4	$>10^{13}$	7,776
8	512	128	96	2	5,256	512
				3	27,630,792	4,096
				4	$>7 \times 10^{14}$	32,768
16	4,096	1,024	896	2	74,256	4,096
				3	$>5 \times 10^{9}$	65,536
48	110,592	27,648	26,496	2	5,534,256	110,592

Table 5-2 - Number of Servers for different DCN topologies (Tab 3.2 in [22])

5.2.2.2 Comparison of performances

	Tree-based topology			**Recursive topology**	
	Basic tree	**Fat-tree**	**Clos network**	**DCell**	**BCube**
One-to-one	1	1	1	$k + 1$	$k + 1$
One-to-several	1	1	1	$k + 1$	$k + 1$
One-to-all	1	1	1	$k + 1$	$k + 1$
All-to-all	n	N	$\frac{2N}{n_{ToR}}$	$> \frac{N}{2^k}$	$\frac{n}{n-1}(N-1)$
Bisection width	$\frac{n}{2}$	$\frac{N}{2}$	$\frac{N}{n_{ToR}}$	$> \frac{N}{4\log_n N}$	$\frac{N}{2}$

Table 5-3 - Performance summary (Tab. 3.3 in [22])

Table 5-3 shows the comparison of some performance metrics of different topologies

- Bandwidth: The first four rows of Table 5-3 show the bandwidth that the topologies can offer under different traffic patterns. “One-to-one” means the maximum bandwidth that the topology can offer when one arbitrary server sends data to another arbitrary server, and so on so forth. “All-to-all” bandwidth means every server establishes a flow to all the other servers. The bandwidths are expressed as the number of links, which implies that each link in a datacenter has the same bandwidth. It is shown in the table that one-to-one, one-to-several and one-to-all bandwidths are in fact limited by the number of ports on the servers, or the degree of servers.
- Bisection Width: The minimum number of wires that must be removed when dividing the network into two equal sets of nodes. The larger the bisection width, the better fault-tolerance ability of a topology. A similar metric, the bisection bandwidth is also widely used, which means the minimum collective bandwidth of all the wires that must be removed to divide the network into two equal sets of nodes. Basic tree has the smallest bisection width, which means that it is the most vulnerable of all. A graceful degradation of performance implies that when more and more components fail in the datacenter, performance reduces slowly without a steep decline in performance. When building a new datacenter, it is usual that a partial topology will be built first, and more components are added based on future need. This partial network can be viewed as a network with many failures.

5.3 Clos Networks

In the mid-1950s Clos networks were created to switch telephone calls. From crossbar topologies, Clos networks architectures moved to chassis-based Ethernet switches and nowadays they are used in modern DCN to provide predictable latency, high performance and resiliency. The *Clos design* is now a key architectural model for datacenter networking.

5.3.1 The first appearance of Clos Networks: the telephony systems

Charles Clos in [35] described *"a method of designing arrays of cross points for use in telephone switching systems in which it will always be possible to establish a connection from an idle inlet to an idle outlet regardless of the number of calls served by the system"*. The concept is that there are multiple paths to "route" the call through the network so that the call will always be connected and not "blocked" by another call.

The key advantage of Clos networks is that the number of cross points (which make up each crossbar switch) required can be far fewer than would be the case if the entire switching system were implemented with one large crossbar switch.

The term fabric came about later because the pattern of links looks like threads in a woven piece of cloth.

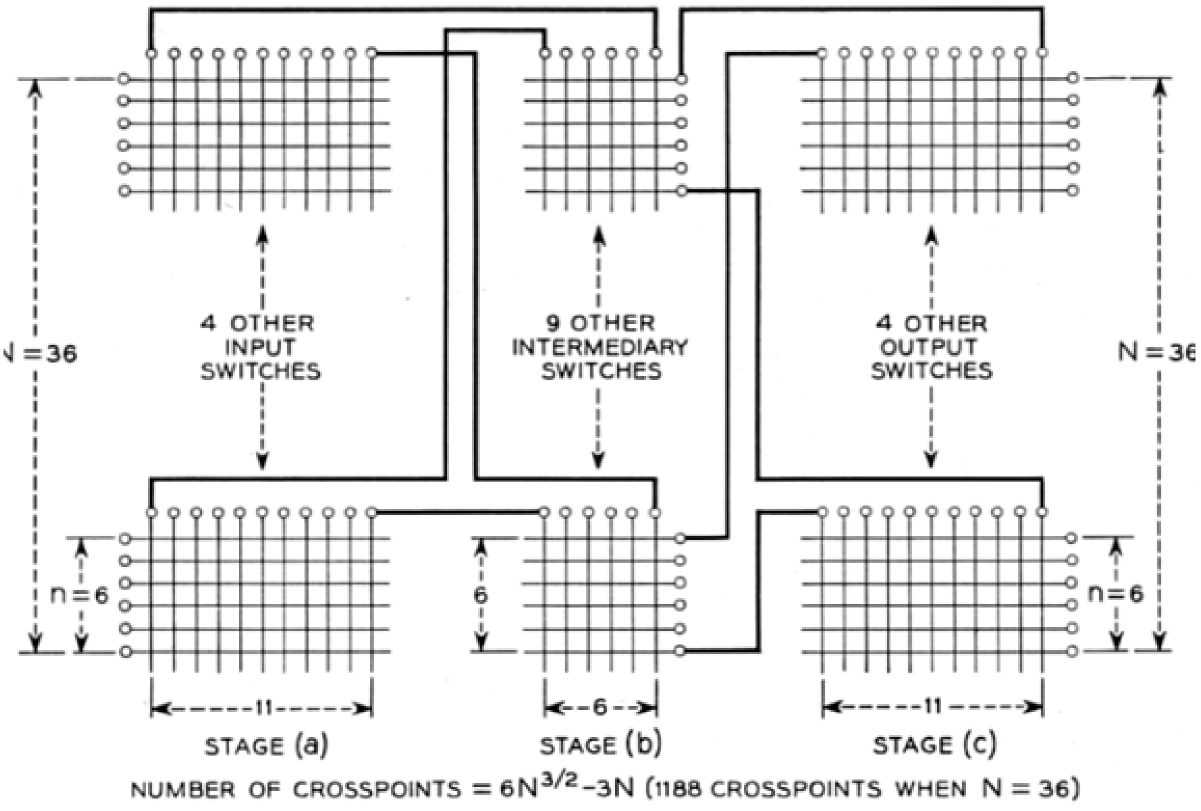


Figure 5-2 - "3 stage switching array" (Fig. 3 in [35])

5.3.1.2 Clos networks architecture

Clos networks are multistage switching networks. Figure 5-3 below shows an example of a 3-stage Clos network where *n* is the number of sources connected to each of the *m* ingress stage crossbar switch: there is precisely one link between each ingress switch and each of the *k* middle stage switch. And each middle stage switch is connected exactly once to each of the *m* egress stage switch.

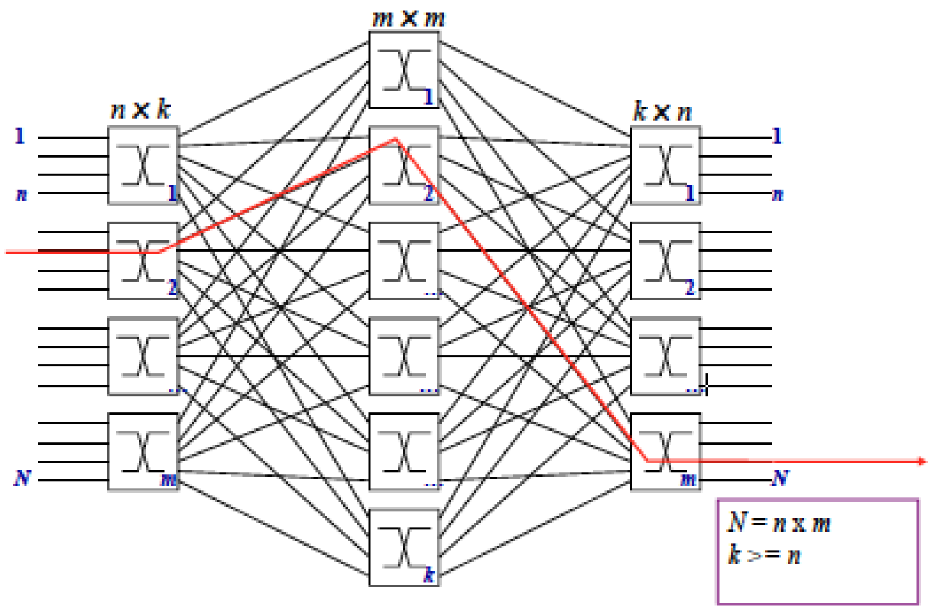


Figure 5-3 - 3-stage Clos network [35]

5.3.1.3 Clos Theorem

It can be shown that with k ≥ n, a Clos network is equivalent to a crossbar switch (it is non-blocking). This implies that, for each input-output matching, in the middle-stage there is a combination of paths connecting inputs and outputs. The Clos theorem shows that to add a new connection there won't be any need for rearranging the existing connections provided that the number of middle-stage switches is big enough.

Clos Theorem: If k ≥ 2n−1, then a new connection can always be added without rearrangement.

5.3.2 The second appearance of Clos Networks: within Network Switches

With the advent of the first Ethernet switches, in the 1990s, Clos networks came back to life. The switch needed to transport frames from any port to any other port, and this "simple" requirement could be implemented with a similar crossbar matrix of connectivity within the switch. The number of switch ports governed the size of the crossbar fabric. With the advent of modular switches, to accommodate faster interface speeds, the crossbar fabric needed to be expanded. The crossbar fabric, together with the interconnections between the line cards inside the chassis, were provided by the switch supervisor card.

Crossbar fabrics became unpopular because they were subject to Head of Line (HOL) blocking due to input queue limitations. Gradually, input and output queues were added to all the interfaces of the Ethernet switches. Today, Ethernet switches provide non-blocking performances with advanced fabric features like priority-based flow control, output queuing, and others.

5.3.3 The Datacenter Network Journey

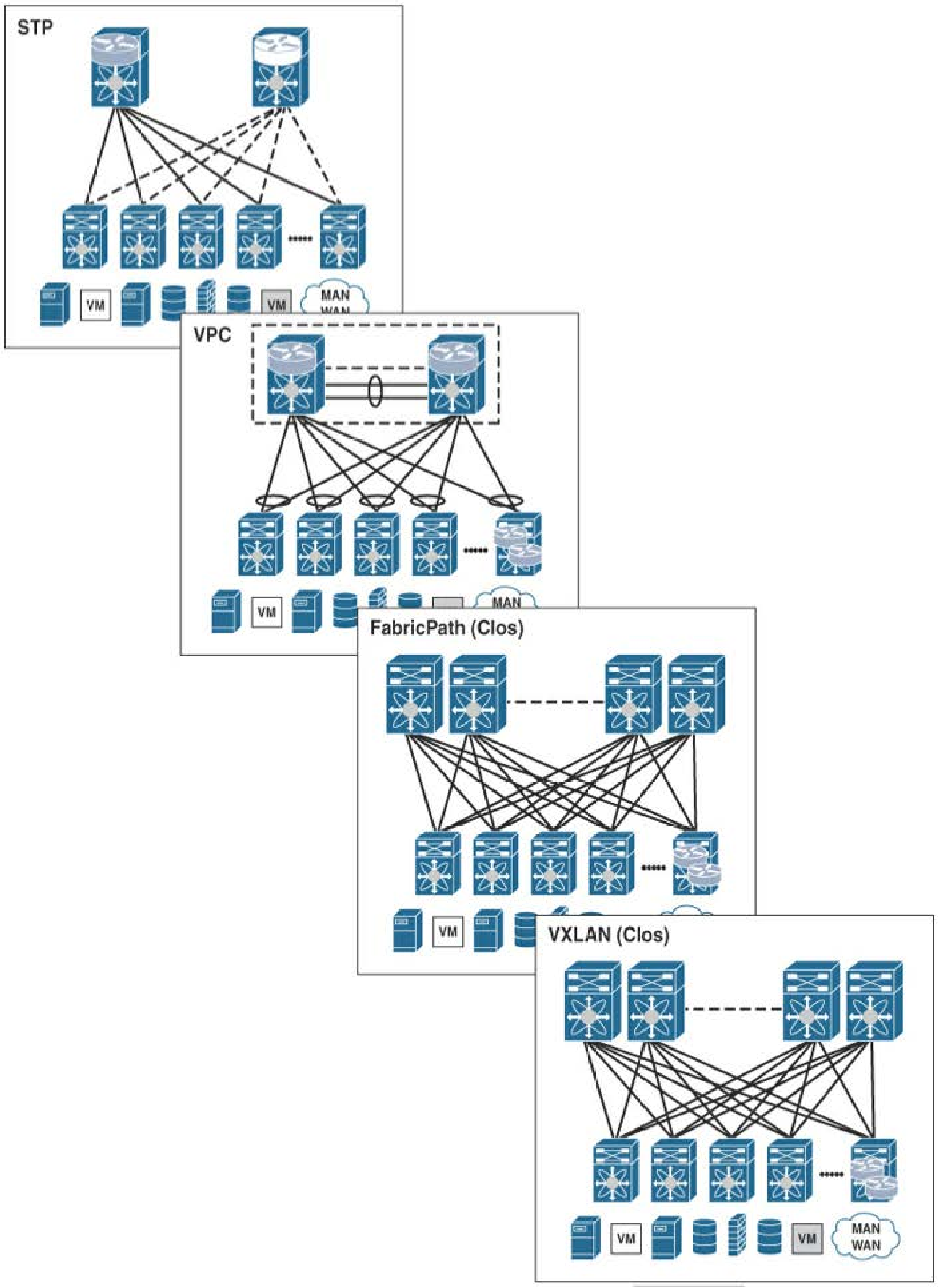


Figure 5-4 - The datacenter network Journey [36]

5.3.3.1 *Fat-tree topology*

Gradually, the "*fat tree*" model of connectivity using the core - distribution - access architecture has been adopted as a standard approach to network design. To prevent oversubscription, the aggregated throughput required at each layer became progressively higher coming closer to the core. As an example, in a server room the access layer could have been a 100Mbps Fast Ethernet link, the uplinks to the distribution layer could have been 1Gbps Ethernet links, and the uplinks from there to the core would have been 4X1Gbps port channels. The fat-tree topology however presented the challenges described in Par. 3.3.

Moving up to the Core Layer, traffic engineering in the datacenter becomes challenging using traditional Ethernet switches. While the links increase in bandwidth getting closer to the core, these links are usually greatly oversubscribed, and blocking can easily occur.

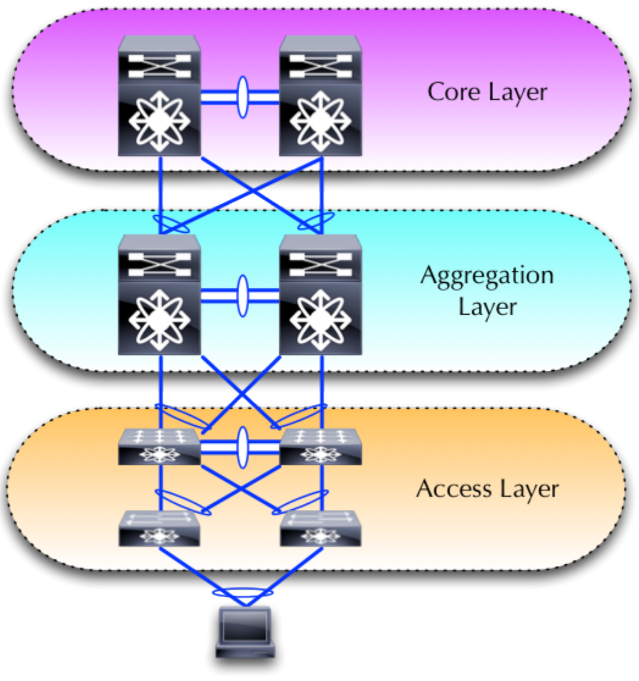


Figure 5-5 - Traditional Three-Tier Datacenter Design

5.3.3.2 *Design using active/active pair of links*

Around 2009, layer two multi-switch redundancy solutions (like Cisco vPC) were introduced: these technologies provide an active/active pair of links that doubled the throughput available compared to the STP active/standby solution[2]. These vPC-like technologies have been used most commonly in the datacenters in the past years, but the new trend of designing a datacenter is using a multi-tier spine-and-leaf design, taking full

[2] These topologies still use STP as a fail-safe mechanism.

advantage of all the links, completely removing the STP and reducing the complexity of the overall solution.

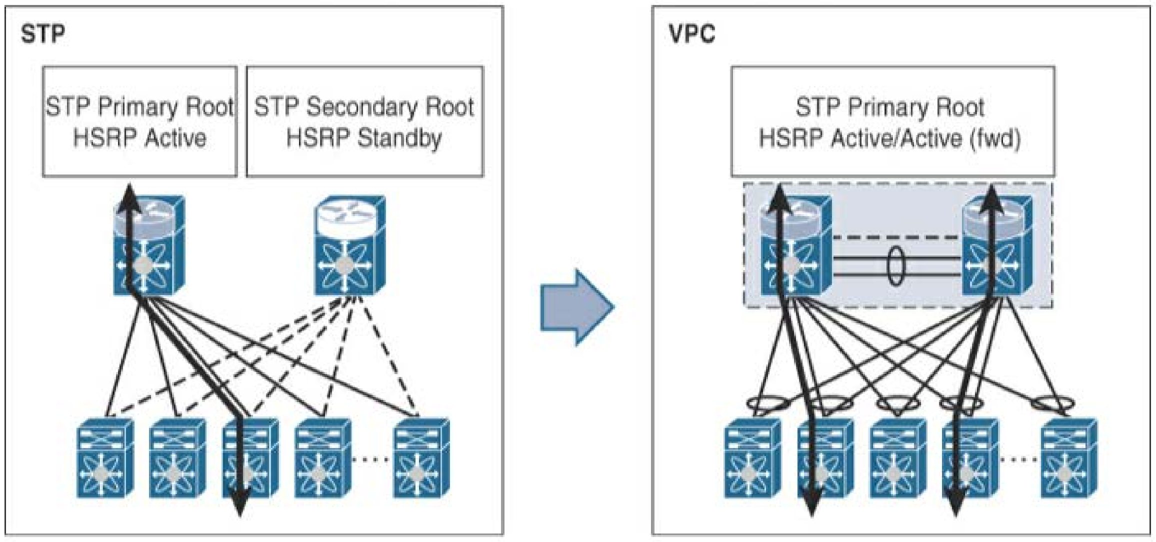


Figure 5-6 – STP to vPC improvement in the DCN [36]

Since 2003, with the introduction of virtual technology, the computing, networking, and storage resources that were segregated in pods in Layer 2 in the three-tier datacenter design can be pooled. This revolutionary technology created a need for a larger Layer 2 domain, from the access layer to the core layer, as shown in Figure 5-7.

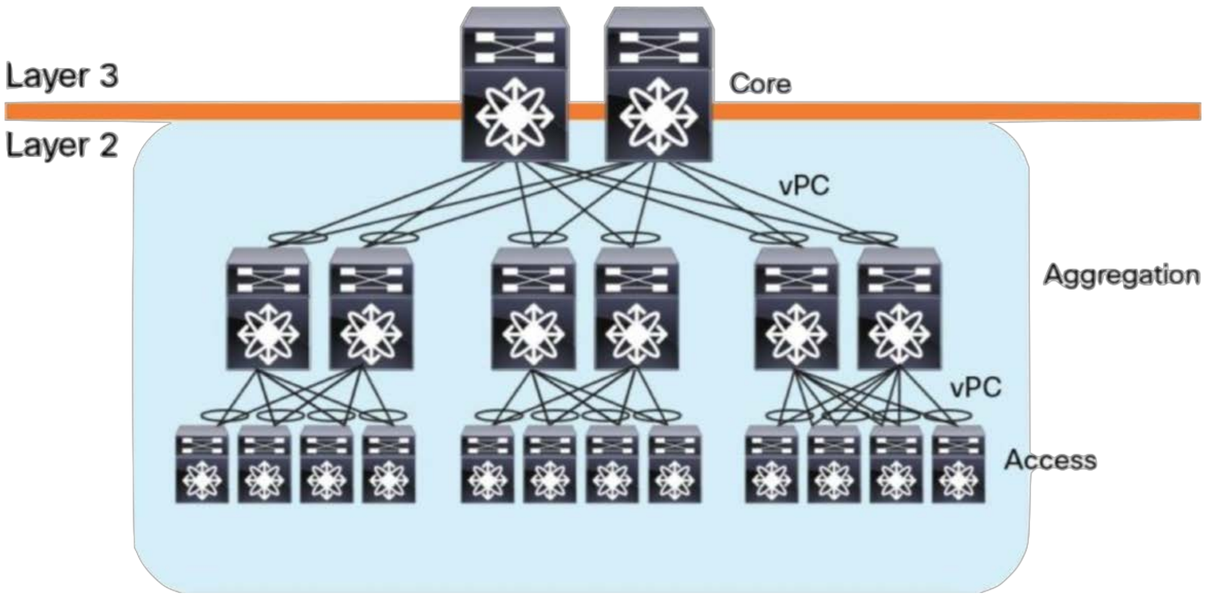


Figure 5-7 - Datacenter Design with Extended Layer 3 Domain [37]

5.3.4 The third appearance of Clos Networks: Spine-Leaf Architecture in DCN

With Layer 2 segments stretched across all the pods, the datacenter administrator can design a centralized, flexible infrastructure that can be reallocated based on needs. Virtual machines can move freely from one hypervisor to the other without changing the network configuration. With virtualized servers, applications are increasingly deployed in a distributed fashion, which further increases east-west traffic. This traffic must be handled efficiently, with low and predictable latency. However, vPCs provide only two active links, and so bandwidth becomes a bottleneck in this architecture. Another challenge is that latency is not predictable: server-to-server latency varies depending on the traffic path used.

To overcome these limitations, a new design approach is required: Clos networks come back to life in their third impersonation, this time as datacenter design called the spine-and-leaf architecture.

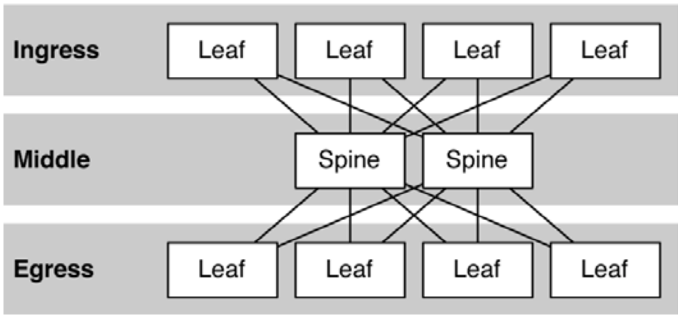


Figure 5-8 - Spine-Leaf Architecture (3-layers Clos)

In this (folded) two-tiers Clos architecture, every leaf switch is connected to each spine switch in a full-mesh topology. The leaf layer consists of access switches connecting hosts such as servers, load balancers, etc. The spine layer is the backbone of the network and interconnects all the leaf switches. Every leaf switch connects to every spine switch in the fabric. The path is randomly chosen (i.e. using ECMP) so that the traffic load is evenly distributed between the spines. In the event of a failure of one spine, it would only slightly degrade performance throughout the datacenter.

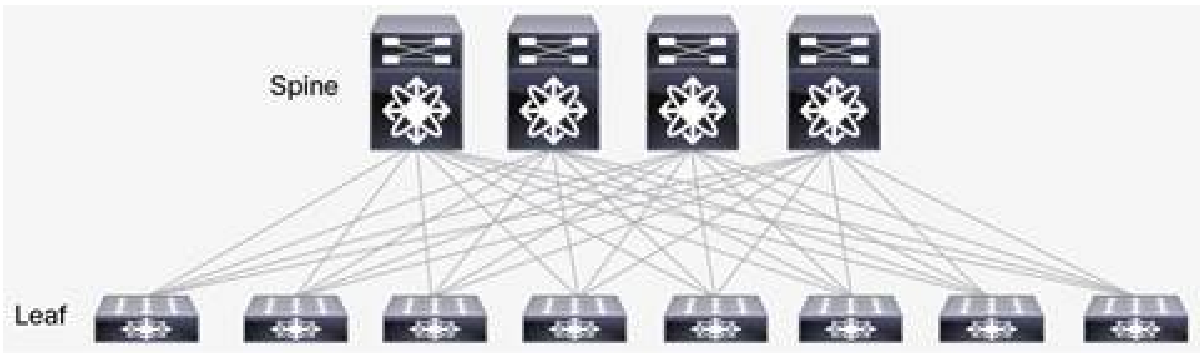


Figure 5-9 - Datacenter Design with Extended Layer 3 Domain [37]

In the event of oversubscription (i.e. there is more traffic than what can be aggregated on the active link at one time), the process for expanding capacity is easy: an additional spine switch can be added, and uplinks can be extended to every leaf switch, resulting in the addition of uplink throughput and reduction of the oversubscription. If device port capacity becomes a concern, a new leaf switch can be added by connecting it to every spine switch and adding the network configuration to the switch. The ease of expansion optimizes the IT department's process of scaling the network. If no oversubscription occurs between the lower-tier switches and their uplinks, then a non-blocking architecture can be achieved.

With spine-and-leaf architectures, no matter which leaf switch a server is connected, its traffic must always cross the same number of switches to get to another server. This approach provides predictable (low) latency: between two hosts in the DCN, unless they are connected to the same leaf, there are always 3 hops (source leaf – spine – destination leaf).

5.3.5 POD-based design

A single large workload can be split into smaller sub workloads, and their output can be combined, summarized, and reduced to find the overall result. Numerous kinds of workloads dealing with huge data volumes might benefit from this *Divide-and-conquer* approach.

Classic Core--Aggregation--Edge design maximizes north-south traffic, Pod designs is optimized for datacenter agility. They assume that the datacenter capacity will be built out over time rather than all at once, and during the lifecycle there will be important changes in price/performance of the infrastructure.

The core building block is the Point of Delivery (PoD): an atomic unit of compute, network and storage. It is designed, deployed, automated and decommissioned as a unit. Pods cannot be "half built" and they cannot be "half upgraded." In practice, pods are often marked with a version number (sometimes called a 'generation').

The very same concept of Pod can be applied to the spine-leaf Clos network topologies, where basically a Pod is the "building block" for the full datacenter network.

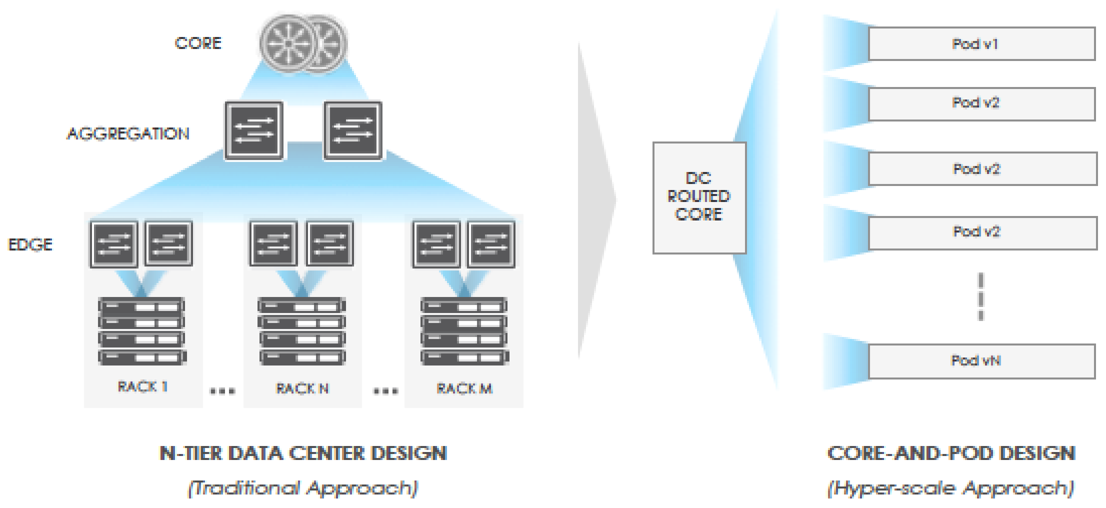


Figure 5-10 - Pod Design vs traditional [38]

5.3.6 Top-Of-Rack (ToR)

The datacenter's access layer presents the biggest challenge to architects because they need to choose the cabling architecture to support the ever-growing needs of the DCN [39] .

The spine-leaf design is per se independent from the physical location of the leaf, however the de-facto standard for this architecture is the ToR model (see Figure 5-11) where endpoints are connected to Network Access Modules (NAM)[3] located in the same racks. These NAMs are then connected to the leaf or spine switches

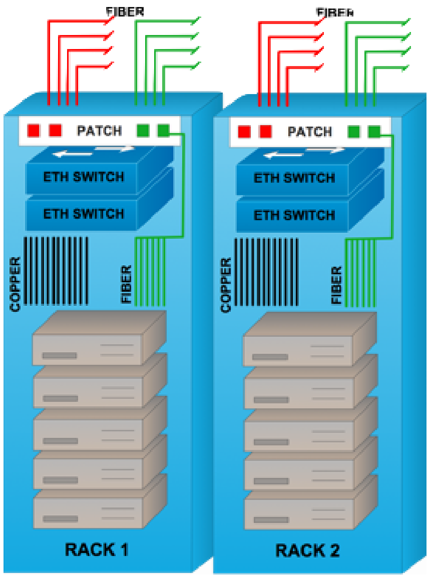


Figure 5-11 - ToR model.

With the ToR approach, leaf, spine, or aggregation can still be connected at the End-of-Row or Middle-of-Row while drastically reducing the amount of cabling and providing a scalable model at the rack level.

5.4 Modern Massively Scalable Datacenters examples

In this paragraph five massively scalable datacenter (MSDC) network topologies are presented: Amazon, Facebook, Google, Microsoft Azure and Oracle Cloud are briefly introduced and compared. Facebook and Google provide public information for their network, Amazon topology is based on the public information extracted from a AWS conference (see [40]), the information for Microsoft Azure is deducted by information provided on MSDN, and the one for Oracle is based on public sources.

It is worth noting that all five MSDCs are based on rearrangeable Clos networks, pod-based, and make extensive use of SDN and virtualization.

[3] NAM could be provided by Leaf Switches, remote line cards, or Fabric deployed as ToR switches controlled by leaf switches deployed as MoR or EoR.

5.4.1 Amazon Datacenter Network Topology

The topology information presented in this paragraph is deducted from [40] during which James Hamilton, Vice President and Distinguished Engineer on the Amazon Web Services (AWS) team, provides some insights about how the Amazon Datacenter network is organized

5.4.1.1 Design parameters

- AWS provides 25Gbps access to the hosts and 50Gbps links to the backbone.
- All hosts are dual connected (2x25Gbps)
- It uses ASICS Tomahawk with 128P@50Gbps Rearrangeable in non-blocking topologies
- It hosts between 50k and 80k servers in a single datacenter
- It uses SDN *(“it is super important (…) we have been using it since the beginning*”)

Assuming the DCN connects 65k dual-homed hosts, the amount of 25G ports is around 130k

5.4.1.2 Deployment Unit

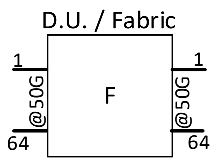


Figure 5-12 - Amazon basic deployment unit

The basic switch used in AWS datacenter is a 128P@50G switch based on Tomahawk ASICS. The “Fabric” layer switch provides 128P@50G (see Figure 5-12), 64 Ports to the ToR switches and 64 Ports to the Spine Switches.

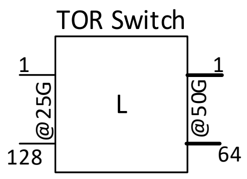
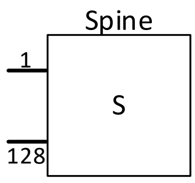


Figure 5-13 - Amazon TOR switch (assumption)

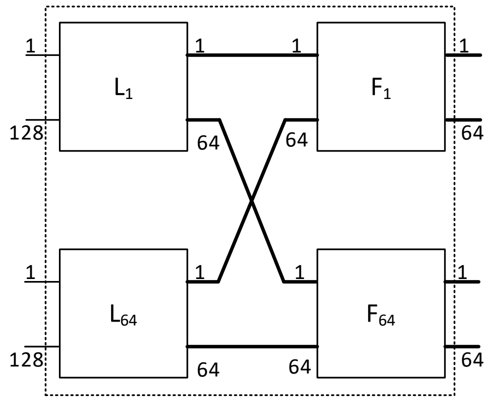
The access layer/ToR switch provides 128P@25G to the hosts, and 64P@50G to the aggregation (see Figure 5-14). The Aggregation/Spine (see Figure 5-13) switch provides 128P@50G connected to the Aggregation switch



.

Figure 5-14 - Fabric Switch Configuration (Aggregation Layer)

5.4.1.3 Amazon POD



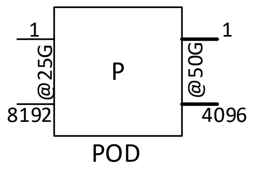


Figure 5-15 - AWS POD

The AWS PoD provides 8,192 access ports (128x64) and is connected to the spine layer with 4,096 ports. The PoD has the following main features

- Non-blocking topology
- Switch number per pod S_{pod}=64+64=128

5.4.1.4 AWS DCN

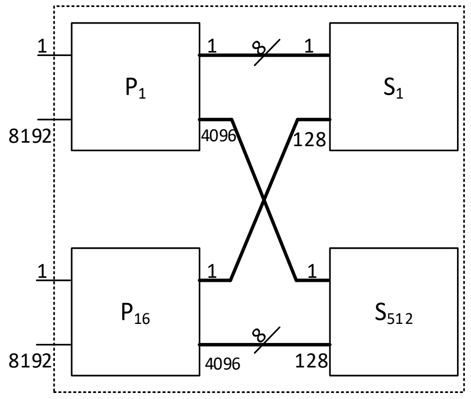


Figure 5-16 - AWS DCN Topology

The required number of pods to connect 65k hosts is 16 (see below).

- 25G Access Ports number P_{acc}=8192 x 16 = 131,072 → connected hosts = 65,536
- Spine switches number: $S_{spine} = \frac{4096 \times 16}{128}$ = 512.
- Each spine is connected to each pod with an 8 x 50Gbps link (128/16=8)
- Bi-section bandwidth: 16 x 4096 @50G = 3,276,800Gbps= 3,276Tbps=3.27Pbps
- Total switch number: S_{tot}=16xS_{pod} + S_{spine} =2,560
- Total 25G Equivalent ports $P_{tot,eq}$ = 256 x S_{tot}=655,360[4]
- Infrastructure Efficiency (25G Access/25G Equivalent Ports) = $\frac{P_{acc}}{P_{tot,eq}} = \frac{131{,}072}{655{,}360} = 20\%$

[4] Every building block switch provides 128P@50G, or 256@25G

5.4.2 Facebook Datacenter Network Topology

Facebook introduced in 2014 their network switch, Wedge [41], along with FBOSS, Linux-based software for controlling switches. Thousands of Wedges have now been deployed throughout Facebook datacenters, and the Wedge design was recently accepted by the Open Compute Project

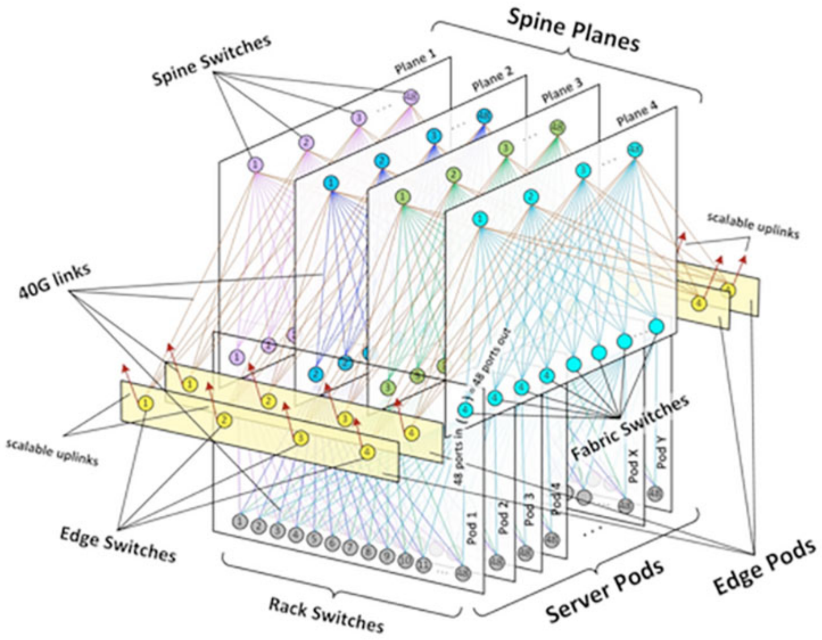


Figure 5-17 - FB Datacenter 3D Representation [41]

5.4.2.1 Deployment unit

The basic deployment units are two switching modules, one for the Access (TOR) the other for Fabric (Aggregation layer) and Spine.

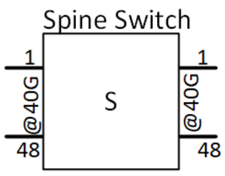


Figure 5-18 - FB DC Deployment unit – Spine and Fabric configuration

Each spine switch provides 96P @ 40G non-blocking

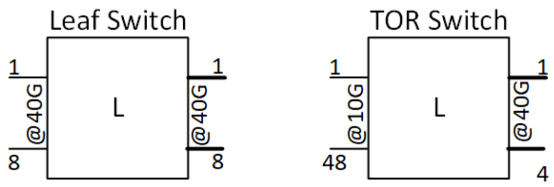


Figure 5-19 - FB DC Deployment unit - TOR configuration

Each Leaf Switch is configured in TOR Mode with 48P @10G for server access and 4 x 40G uplinks, providing 160G total bandwidth to the fabric

5.4.2.2 FB Spine switch equivalent

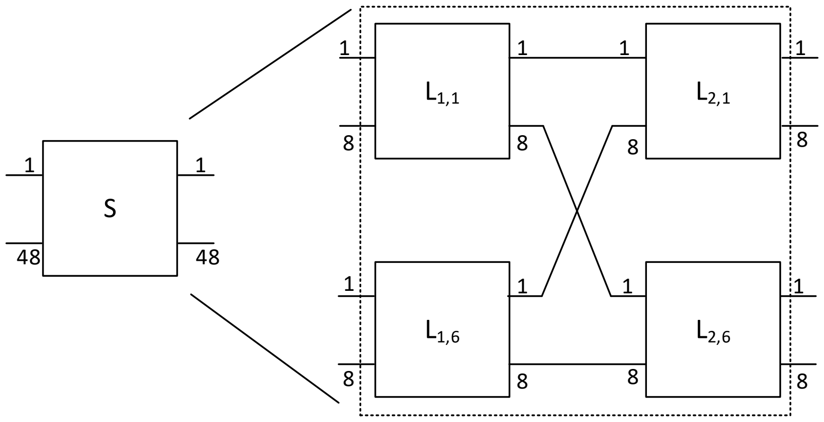


Figure 5-20 – FB spine – equivalent

To better evaluate the complexity and the efficiency of the network, it is worth noting that a 96P spine switch providing 96P@40G can be built using a 2 stages Clos network made by 16P@ 40G building blocks (see Figure 5-20). In terms of network complexity, a FB spine is equivalent to a 2-stage folded Clos network made by 12 x 16P@40G Switches.

5.4.2.3 FB POD

A pod is like a layer3 micro-cluster and it is not defined by any hard-physical properties. it is simply a standard “unit of network” on the fabric. Each pod is served by a set of four fabric switches.

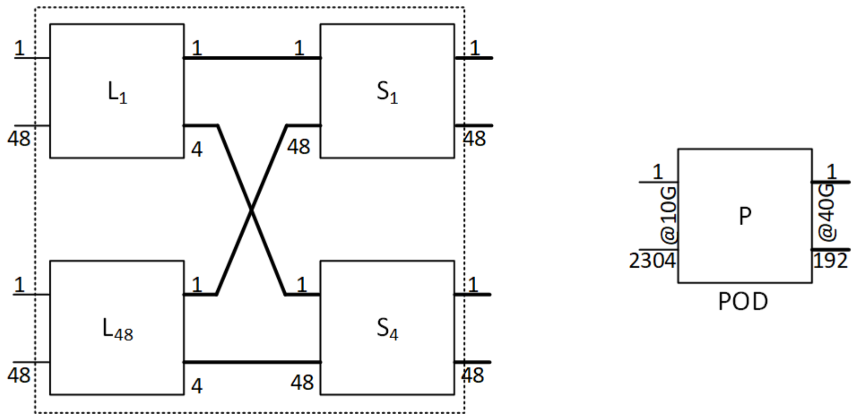


Figure 5-21 - Facebook DC POD

Each pod has 48 server racks, and this form factor is always the same for all pods. It’s a building block that fits into various datacenter floor plans, and it requires only basic mid-size switches to aggregate the TORs. Each PoD has the following characteristics

- blocking topology (Oversubscription 1:4)
- Switch number per pod $S_{pod}=48+4S_{spine,eq}=48+4x12=96$

Each pod is equivalent to a Blocking Clos-like two stages topology made using 96 16Ports switches.

5.4.2.4 FB DCN Topology

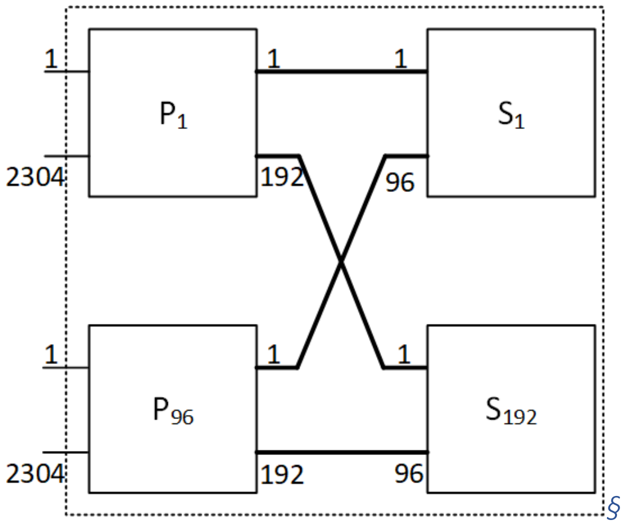


Figure - 5-22 - Facebook DCN Topology

- Total number of connected hosts = $Ports_{hosts}$ = 221,184
- Bi-sectional bandwidth = 96 x 192@40G = 737280G=0.73Pbps
- Total 16P switch number: S_{tot}= 96 x S_{pod} + 192 x $S_{spine,eq}$ = 11,520
- Total 40G port number P_{tot} = S_{tot} x 16 = 184,320
- Total 10G Equivalent Port Number $P_{tot,eq}$ = 4 x P_{tot} =737,280
- Infrastructure Efficiency (10G Access ports/Total 10G equivalent Ports)=$\frac{P_{acc,eq}}{P_{tot,eq}} = \frac{221{,}184}{737{,}280} =$ 30%

It is worth noting that the number above could be used just as an indicative measure, not as an absolute value, because the assumption of Spine switches made by Leaf Switches is purely theoretical.

5.4.3 Google Datacenter Network Topology

5.4.3.1 Google DCN

Jupiter is the name given to the Year 2015 iteration of Google DCN [42] and needed to scale more than 6x the size of previous existing fabric (see Table 5-4). Unlike previous iterations, there was a requirement for incremental deployment of new network technology because the cost in resource stranding and downtime was too high. Upgrading networks by simply forklifting existing clusters stranded hosts already in production. With Jupiter, new technology would need to be introduced into the network in situ. Hence, the fabric must support heterogeneous hardware and speeds

Datacenter generation	First deployed	Merchant silicon	ToR config	Aggregation block config	Spine block config	Fabric speed	Host speed	Bisection BW
Legacy network	2004	Vendor	48x1G	–	–	10G	1G	2T
Firehose 1.0	2005	8x10G 4x10G (ToR)	2x10G up 24x1G down	2x32x10G (B)	32x10G (NB)	10G	1G	10T
Firehose 1.1	2006	8x10G	4x10G up 48x1G down	64x10G (B)	32x10G (NB)	10G	1G	10T
Watchtower	2008	16x10G	4x10G up 48x1G down	4x128x10G (NB)	128x10G (NB)	10G	nx1G	82T
Saturn	2009	24x10G	24x10G	4x288x10G (NB)	288x10G (NB)	10G	nx10G	207T
Jupiter	2012	16x40G	16x40G	8x128x40G (B)	128x40G (NB)	10/40G	nx10G/nx40G	1.3P

Table 5-4 - Google DCN generations [42]

5.4.3.2 Deployment unit – Centauri reconfigurable chassis

The unit of deployment is a Centauri chassis, a 4RU chassis housing two line-cards, each with two switch chips with 16x40G ports controlled by a separate CPU line card. Each port could be configured in 4x10G or 40G mode. There are no backplane data connections between these chips; all ports are accessible on the front panel of the chassis.

Centauri units are used in three different configurations, Main Mode, TOR Mode and Middle Mode.

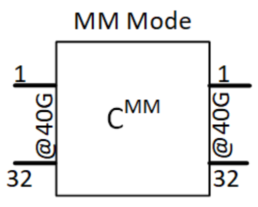


Figure 5-23 - Centauri in Main Mode

Centauri in Main Mode provide 64 ports @40 Gigabit per seconds and they are used as building blocks on the aggregation and spine layers.

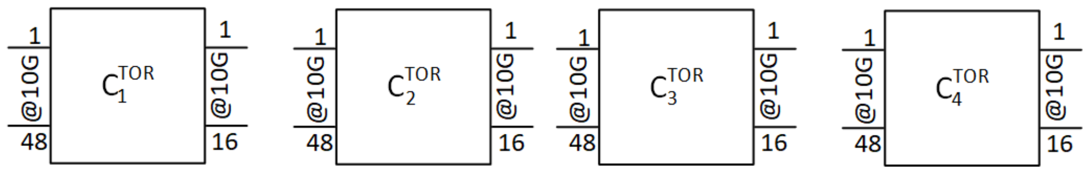


Figure 5-24 - Centauri in TOR Mode

Centauri in TOR Mode are connected to the hosts (provide 48P@10G with reconfigurable speed @40G) on one side, and to the fabric on the other side (16 ports @10G connected to 8 Main Blocks with double redundant links)

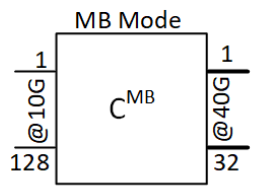


Figure 5-25 - Centauri in Middle Mode

Centauri in Middle Mode provide 128P@10G (connected to the TOR Switches) and 32P@40G (connected to the Spine) and are used in the construction of the medium block, needed in the Aggregation layer.

5.4.3.3 Middle Block

The logical topology of a Middle Block is a 2-stage blocking network, with 256x10G links available for ToR connectivity and 64x40G available for connectivity to the rest of the fabric through the spine

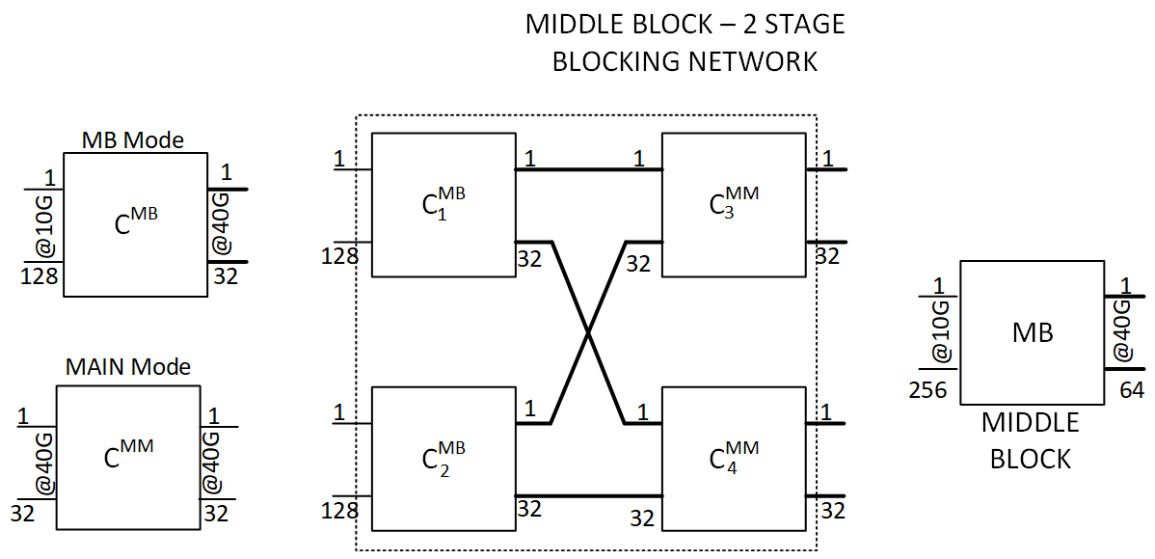


Figure 5-26 - Middle Block topology

- Each middle block is made by 4 switches a 2-layer non-blocking Clos network
- S_{MB}=4

5.4.3.4 Spine Block

Jupiter employs six Centauris in a spine block exposing 128x40G ports towards the aggregation blocks. We limited the size of Jupiter to 64 aggregation blocks for dual redundant links between each spine block and aggregation block pair at the largest scale, for local reconvergence on single link failure.

The available documentation does not specify the internals of the spine block, it states however that the spine switch provides 128P@40G non-blocking using 6 Centauri. This leads to the assumption of a 3-layers rearrangeable Clos topology (See Figure 5-27)

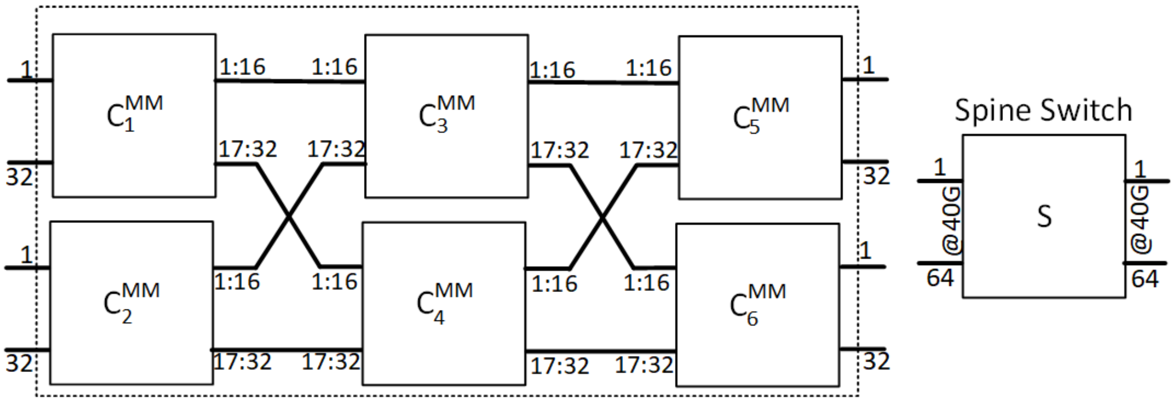
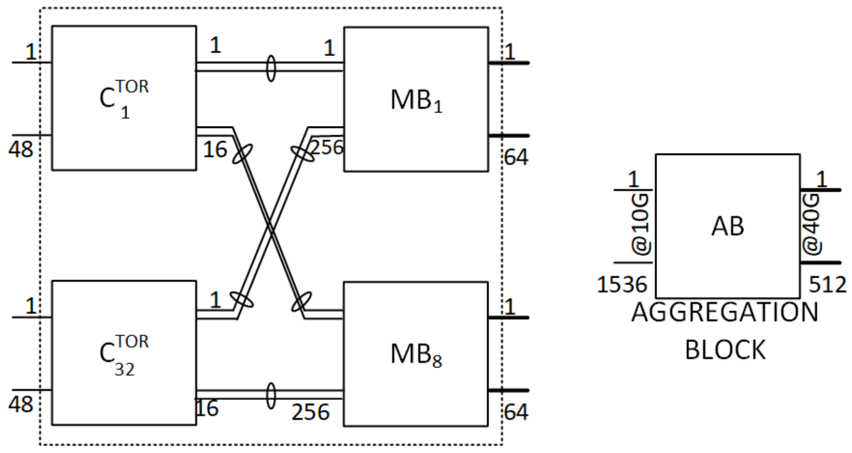


Figure 5-27 - Spine Block topology diagram

- S_{SB}=6

5.4.3.5 Aggregation Block - Tier 1

In the aggregation block, Each ToR chip connects to eight MBs with dual redundant 10G links. The dual redundancy aids fast reconvergence for the common case of single link failure or maintenance. Each aggregation block exposes up to 512x40G links towards the spine blocks



- S_{AB}=32+8 x S_{MB}=64

The Aggregation block is made by 64 Centauris

Figure 5-28 - Jupiter Aggregation Block

5.4.3.6 The Jupiter DCN Topology

The size of Jupiter is limited to 64 aggregation blocks for dual redundant links between each spine block and aggregation block pair at the largest scale, for local reconvergence on single link failure. In its largest configuration, Jupiter supports 1.3Pbps bisection bandwidth among servers.

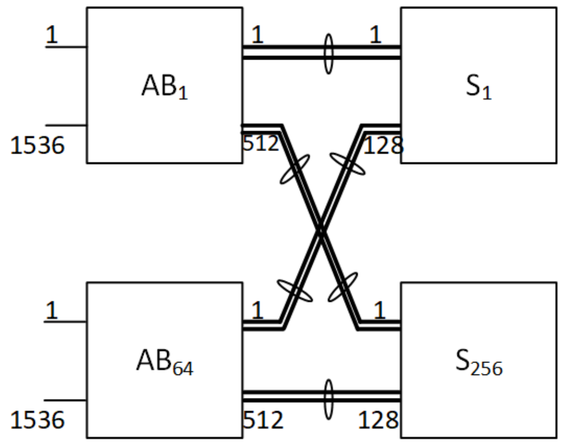


Figure 5-29 - Jupiter DCN Topology

- Bi-section bandwidth: 64*512*40G = 1,310,720G=1.3Pbps
- Max number of connected Hosts: P_{acc} =1,536 x 64 = 96,768
- Total number of Centauri switches S_{TOT} = 64 x S_{AB}+256x S_{SB}= 5,632
- Total number of 40G ports P_{TOT} = 64 x S_{TOT} = 360,448
- Total 10G Equivalent Port Number $P_{tot,eq}$ = 4 x P_{tot} = 1,441,792
- Infrastructure Efficiency (10G Access ports/Total 10G equivalent Ports)=$\frac{P_{acc}}{P_{tot,eq}} =$

  $\frac{96,768}{1441792}$=6.71%

5.4.4 Other topologies

5.4.4.1 Microsoft Azure Network Topology

This paragraph presents a brief summary of the information available in [43] and [44]. The design of Microsoft Azure DCN took place to scale up to 10M endpoints and hundreds of millions of VMs and VNFs. Other design requirements are: use commodity HW (white box switches), scale at low operational and computational capacity providing elasticity and service velocity.

The network technologies in use are MPLS + SDN on a physical Clos POD-Based network topology.

- MPLS:
  - IP lookup only at the edge (Leaf layer) and unify forwarding
  - Built-in network Virtualization and ease of scale-up
- SDN
  - Decouple control plane from data plane, take ownership of control plane
  - Reduce number of protocols, use commodity switches

The solution was to use Hierarchical SDN (HSDN) and it is briefly described in Par. 16.3

5.4.4.2 Oracle Cloud

From the available documentation [45] it seems that Oracle is using a non-blocking Clos network based on interconnected PODs, providing 10Gbps access and using white box switches with SDN.

5.4.4.2.1 Datacenter network (Availability Domains)

An Availability domain (AD) is a completely independent datacenter, connecting 1M hosts using a non-blocking full-mesh Clos Topology. The access is provided through

ASICs are used in the AD's both on the host layer (as Converged Network Adapters, providing I/O virtualization and network access) and in the switches.

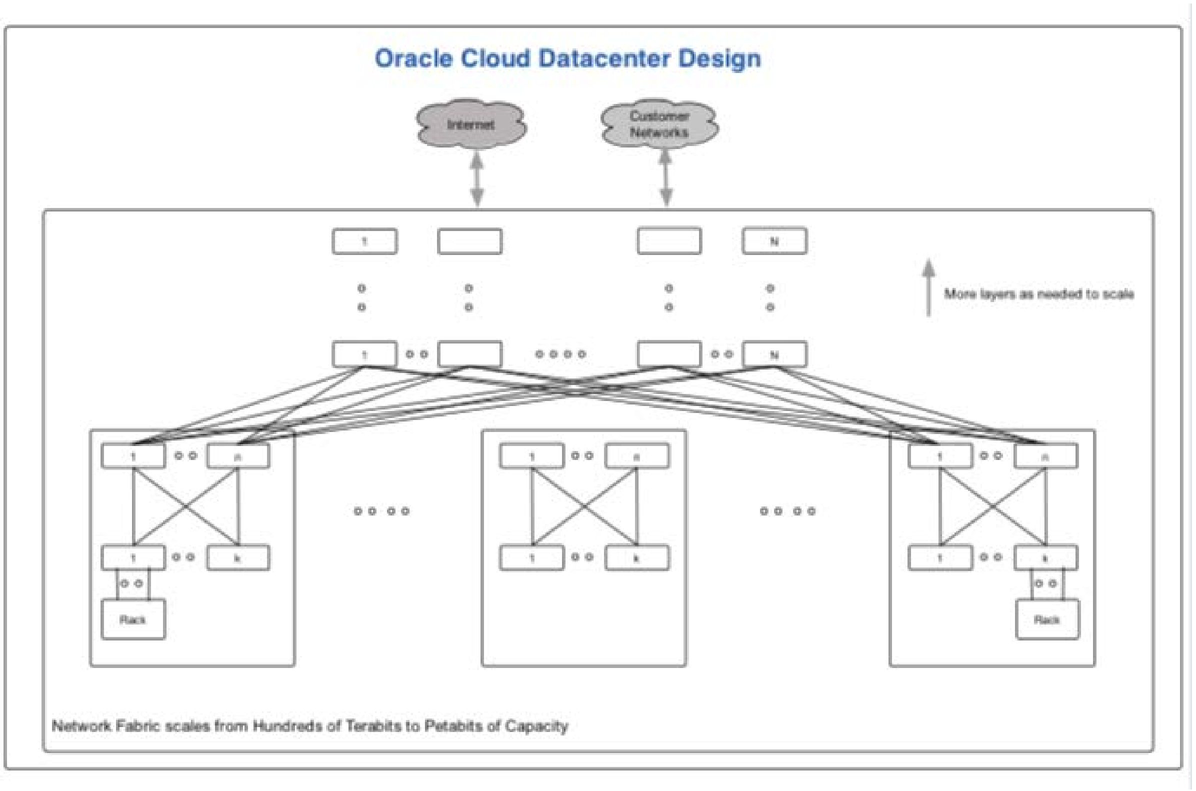

Oracle Cloud Datacenter Design
Internet
Customer Networks
More layers as needed to scale
Rack
Rack
Network Fabric scales from Hundreds of Terabits to Petabits of Capacity

Figure 5-30 - Oracle DCN Clos Topology [45]

5.4.4.2.2 Regions

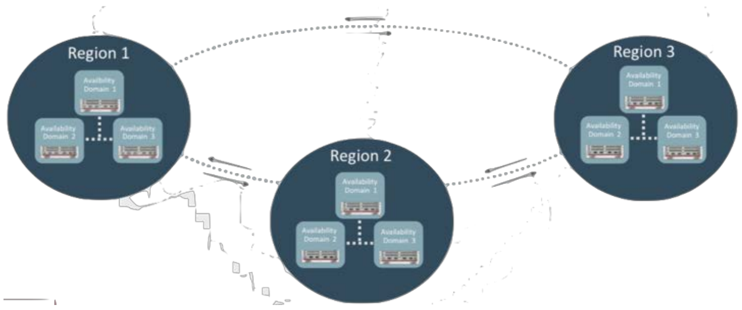


Figure 5-31 - Regions in Oracle Cloud [45]

Each region contains 3 fault independent AD's (availability domains) with a maximum distance of 40km, and provides <5ms RTT latency, 1Tbps throughput for inter-region traffic.

5.4.5 Comparison of the MSDC topologies

This paragraph briefly compares the MSDC networks previously shown. Due to the lack of publicly available information about of Microsoft Azure and Oracle network topologies, these two have been excluded from the comparison. It is worth noting that AWS and Google provide PaaS, IaaS and SaaS services, but Facebook provides only SaaS services: this implies that FB has better knowledge and control over applications and dataflows, which in turn allows FB to better "clusterize" their network.

The comparison parameters have been grouped in the following three categories:

- Architecture:
    - Topology Type: blocking or non-blocking.
    - Oversubscription ratio: how many access ports are connected to a single uplink port in the access block. In case of non-blocking topology, this parameter is always 1:1 or better (leaving room to future expansion).
    - Stages of the Clos topology: how many stages build the Clos Topology
    - Base Building Blocks: physical characteristics of the base deployment unit
    - Access and Uplink speed, and Access/ Uplink speed ratio
    - Pod size
- Size:
    - Total number of switches: how many switches build the network
    - Number of Access ports: how many hosts can be connected to the network
    - Number of Normalized Access ports: the access ports number normalized using the access-to-uplink speed ratio
    - Total Number of Normalized Network ports: a number describing how many "normalized" ports exist in the network.
- Performance:
    - Access Throughput: the total throughput available for the hosts at the access
    - Bi-Sectional Throughput: total throughput between spine switches and Pods
    - Network Efficiency: the ratio between normalized access ports and total normalized ports. This parameter describes the percentage of ports connected to hosts, and it is equal to 1.0 when 100% of the ports are connected to hosts.
    - Infrastructure Overhead: dual of the efficiency above (1- network efficiency), this is the percentage of network ports not directly connected to hosts, whose purpose is to move packets from one intermediate stage to the other.

Description	AWS	FACEBOOK	Google
Type	Non-Blocking	Blocking	Non-Blocking
Oversubscription (OS) ratio	1:1	3:1	1:1
Stages	3	4	6
Base Building Block Switch	128P@50G	16P@40G	64P@40G
Access (Gbps)	25	10	10
Uplink (Gbps)	50	40	40
Access /Uplink speed ratio	2:1	4:1	4:1
Pod Size	8192x4096	2304x192	1536x512
Total Access Ports	131072	221184	96768
Tot Switch number	2560	11520	5632
Normalized Access Ports	131072	221184	96768
Total Normalized Ports	655360	737280	1441792
Access Bandwidth (Pbps)	3.28	2.21	0.97
Bi-sect Throughput (Pbps)	3.28	0.73	1.31
Infrastructure Overhead	80.00%	70.00%	93.29%
Network Efficiency	20.00%	30.00%	6.71%

Table 5-5- Comparison of AWS, Facebook and Google Jupiter DCNs

5.4.5.1 Architecture

From an architecture perspective, both AWS and Google adopt a non-blocking design with an Oversubscription ratio equal to 1:1 or better, and Facebook adopt a blocking design with a 3:1 Oversubscription. Nevertheless, FB has full control over infrastructure, applications and dataflows inside the MSDC because it does not provide IaaS/PaaS services to customers. This implies that FB ability to localize applications and to "*segregate*" their dataflows inside the pod, avoiding the need to traverse the uplinks, is way better than AWS and Google: probably this is one of the reasons why FB DC architects decided to adopt a blocking design.

AWS uses the biggest building block switch (128Ports) of the three: this design choice, together with the smallest stages number of its Pod topology (3 stages for AWS), reduces the overall number of switches. On the other hand, AWS uses non-standard link speeds (25G/50G) with the biggest access/uplink ratio (2:1): this indirectly implies[5] that AWS is the solution with the biggest amount of wires connecting the access and the fabric layer.

AWS is the solution with the biggest Pod size: it is worth noting that a Pod is also a failure domain: a major failure in a Pod would, potentially, impact all the hosts connected to it, and the bigger the pod, the bigger the impact of its failure.

[5] In a non-blocking topology, the access/uplink ratio defines how many access links are supported by one uplink, and (indirectly) how many wires connect Pod to distribution.

5.4.5.2 Size

FB is the “biggest” DCN in terms of access ports and sheer switches number, however the total number of normalized ports for FB-DCN is nearly half that of Google and AWS: this is because FB DCN is a blocking architecture, so its overall network “size” is smaller than AWS and Google.

Google, due to its choice of adopting a 6-stages architecture, its smaller building block, and the redundancy between main block and ToR switches, has the greatest network overhead, the smallest efficiency and biggest network in terms of normalized ports number.

5.4.5.3 Performance

AWS provides the greatest bi-sectional throughput, 3.28Pbps, with 20% of the overall normalized ports connected to services (so, an 80% infrastructure overhead). Google DCN is the only solution providing a bi-sectional throughput greater than the access bandwidth: this leaves room for redundancy and future growth, but of course decreases the overall network efficiency (some ports are, strictly speaking, not required). FB, finally, provides 2.21Pbps at the access stage but “only” 0.73Pbps Bi-Sectional Throughput: this “small” value (and the difference between access and bi-sectional throughput), again, is due to the blocking design. Due to the 3:1 Oversubscription, the bi-sectional throughput is 33% of the access throughput.

The infrastructure efficiency (ports connected to services against overall ports) of a non-blocking network is a direct function of the building block size and an inverse function of the number of stages: from a high-level perspective, the bigger the building block, the better the infrastructure efficiency, and the bigger the number of stages, the worse the infrastructure efficiency (more stages would require more ports not directly connected to hosts). So, the combination between the biggest deployment unit (128P) and the smallest number of stages provides, for AWS, a DCN more efficient than Google DCN[6]. The tradeoff, though, is again that a failure of a base deployment unit will impact, in case of AWS, twice the number of hosts that would be impacted in Google DCN (64P), and a smaller number of stages provides less resiliency and redundancy in the distribution layers, expanding their failure domains.

Facebook, due to its blocking design, is the solution with the greatest network efficiency (and the smallest network overhead): again, a blocking topology implies a “smaller” network.

[6] The overall Datacenter efficiency is a complex concept, involving power, cooling, and many other infrastructure components, and it goes way beyond the concept of Datacenter Network efficiency defined in this paragraph.

6 Network Virtualization

6.1 Abstract

This chapter presents the first virtualization technologies that have been introduced in the networking domain: overlays, tunneling, VRFs and vDCs. These technologies represent well established tools in the current datacenter designs and are the founding elements of datacenter networks. A framework for datacenter network virtualization is introduced in [46], and [47] presents a thorough introduction to the topic.

6.2 Rise of virtualization

The scale and fluidity introduced by server and storage virtualization in the datacenter pushed forward the need for automation, multitenancy, and multipathing. The general principle of virtualization consists in creating a higher-level abstraction that runs on top of the actual physical resource that is being abstracted (storage, compute, etc.).

In case of the network services, Network Virtualization (NV) refers to a solution where a virtual abstracted network runs on the top of the actual physical network. With virtualization, network administrators can create a network anytime and anywhere they want, as well as expand and contract existing networks.

6.3 Properties of network Virtualization in the Datacenter

NV in the datacenter could be seen from many different perspectives; in this paper we summarize NV with the following features:

- Switch Systems Virtualization
- Link Virtualization
- Broadcast Domain Virtualization (VLAN)
- Virtualization by Overlays and encapsulation
- Virtualization of the internal network processes

6.3.1 Switch systems Virtualization

In network switches the main/core component is often called Supervisor, and it is the part (combination of software and dedicated ASICs) of the switch managing the core services. Some network switches support the virtualization of the supervisor. This feature enables redundancy of the control plane, and is currently implemented using two techniques:

- VSS, Virtual Switching System
- Stacking

6.3.1.1 Virtual Switching System (VSS)

Virtual Switching System (VSS) [48] is a system virtualization technology available on the datacenter-grade Cisco Catalyst switches (i.e. 65xx, 68xx, etc.)[7]. This solution connects two switches with a dedicated redundant link called Virtual Switch Link (VSL) and then pools the pair into one logical virtual switch called VSS. A VSS allows two physical Cisco Switches to operate as a single logical virtual switch acting a single management unit, supporting multiple-chassis port-channel and In-Service Software Upgrade (ISSU). In a VSS the control and management planes of the switches are shared.

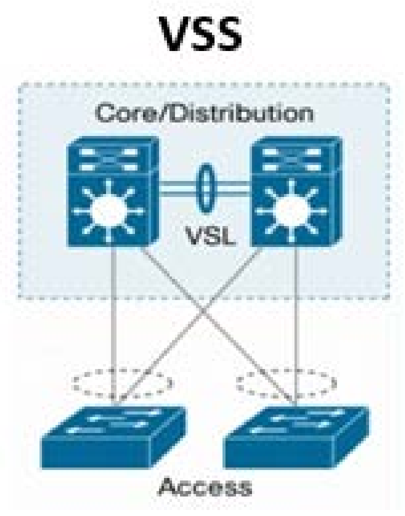


Figure 6-1 - Cisco VSS and VSL [49]

6.3.1.2 Stacking

Multiple switches could be connected in a (double) chain to create a single switching unit through a stack interconnect (see Figure 6-2). Configuration and routing information are shared by every switch in the stack, creating a logical single switching unit.

Switches can be added to and deleted from a working stack without affecting performances. The stack is managed as a single unit by a master switch, which is elected from one of the stack member switches.

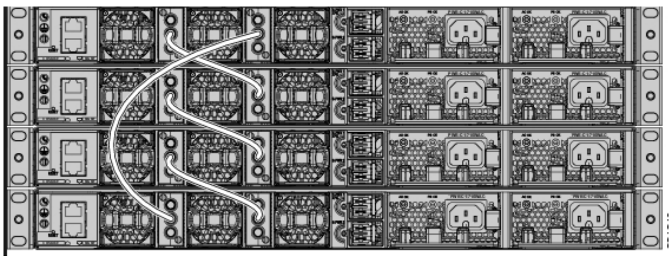


Figure 6-2 - Stacking of 4 switches

[7] Other manufacturers offer an equivalent feature but use a different name to define it.

6.3.2 Access-Link Virtualization

The ability to aggregate multiple interfaces in one logical port could be part of the virtualization process (it abstracts network resources from physical devices). In this chapter, two link virtualization techniques are briefly described:

- LACP, to aggregate multiple ports on the same logical switch
- Virtual Port Channel (vPC), to aggregate multiple links on two different switches.

6.3.2.1 LACP

The IEEE802.3ad Link Aggregation Control Protocol (LACP) [50] defines a method to control the bundling of multiple physical ports to form a single logical channel. LACP allows a network endpoint to negotiate an automatic bundling of links by sending LACP frames to the device directly connected that also supports LACP.

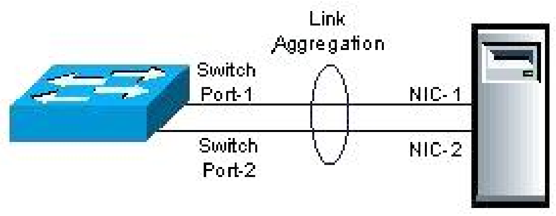


Figure 6-3 - LACP aggregation

6.3.2.2 Virtual Port Channel

Virtual Port Channels (vPCs) [51] allow links physically connected to two different Cisco® Nexus switches to appear to be coming from a single device and a single Port Channel. Technologies as virtual Port Channel (vPC), Multichassis EtherChannel (MCEC) and virtual switching system (VSS) also allow a downstream device to attach to a pair of switches. The main difference between VSS and vPC, though, is that VSS unifies the control and management planes of the two-member switches, vPC instead keep the control/management planes of the two switches separated: they just synchronize the required information.

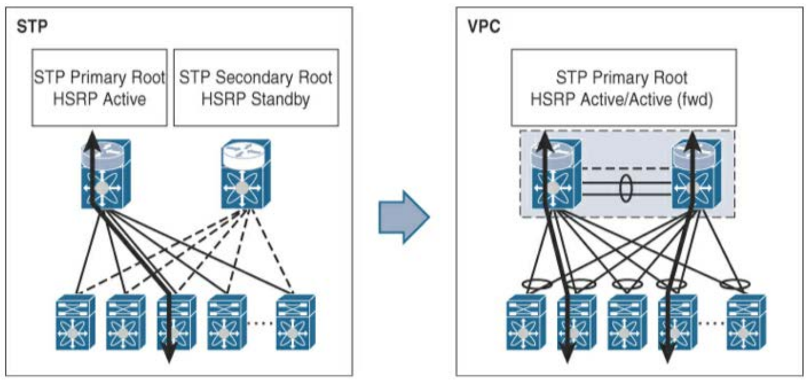


Figure 6-4 STP to vPC improvement [51]

6.3.3 VLANs

Instead of wiring a separate physical infrastructure for each group, VLANs [14] can be efficiently used to segregate the hosts based on the business needs, with a unique identifier allocated to each logical network (VLAN id). VLANs de-couple Broadcast domains and network switches, providing the ability, as an example, to configure multiple LANs on the same physical switch.

In 802.1Q VLANs, usually the access switch enforces the VLAN ID as well as other security and network settings (e.g., quality of service). The VLAN ID is a 12-bit field, which allows a theoretical limit of 4094 unique logical networks (tag 0 and tag 4095 are reserved).

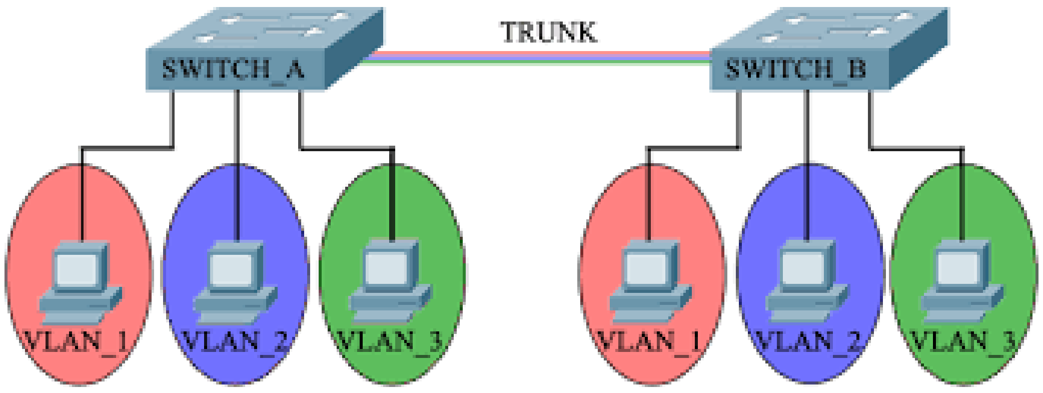


Figure 6-5 - VLAN Access and Trunking

6.3.4 Network overlays

Network Overlaying is a mechanism to abstract (decouple) specific resources from the network devices providing them[8]. Overlays use abstraction and encapsulation to run multiple separate, discrete virtualized network layers on top of the physical network that provides them (often called underlay).

Overlays create a virtual path between multiple endpoints connected to the same physical network, logically placing them in different segments. This virtualization can be achieved using routing (L3 or L4 overlays) or switching (L2 overlay, MAC in MAC) protocols that can apply software tags, labels, and/or encryption to create virtual tunnels running across the underlay. If encryption is used (i.e. IPSEC), the data can be secured between the endpoints so that the end-users must be authenticated to use the connection.

By separating host and topology address spaces, overlays provide a level of abstraction that allows both to grow independently.

[8] VLANs can also be a special case of Layer-2 network overlay, but this paragraph focuses on overlay technologies that provide “full” L2 or L3 encapsulation.

6.3.4.1 L2 in L2 - Cisco Fabricpath

Cisco FabricPath [52] is a L2-in-L2 (MAC-in-MAC, or L2 routing) encapsulation which removes the need of STP in Layer 2 networks. Fabricpath is the Cisco “version” of the TRILL IETF standard [53] using Layer 2 IS-IS to share topology information between the network switches. FabricPath switches act “like” routers, building reachability tables and getting all the advantages of Layer 3 like ECMP. In addition, all the available links are used and provides optimal forwarding. Figure 6-6 shows the FabricPath header.

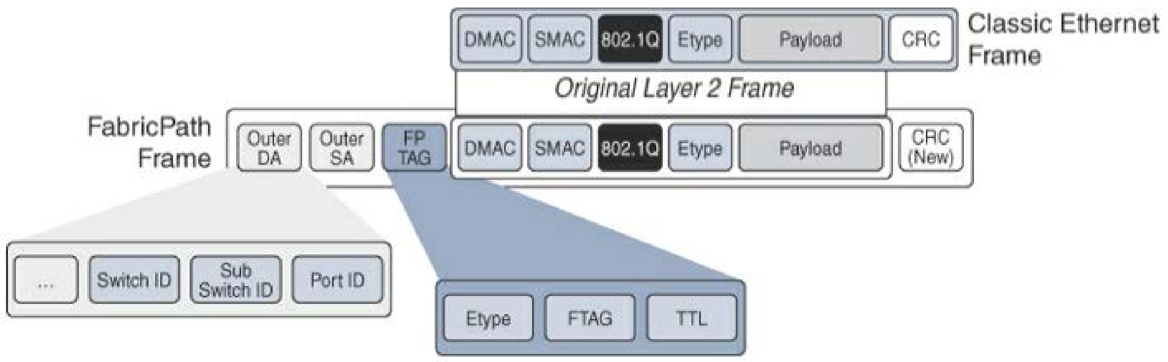


Figure 6-6 - FabricPath Header

While FabricPath has been adopted by thousands of Cisco customers, it has been challenged because it is a Cisco proprietary solution and lacks multivendor support. In addition, with IP being the de facto standard in the networking industry, a push for an IP-based overlay encapsulation occurred. As a result, VXLAN was introduced.

6.3.4.2 L2 in L4 – VXLANs

Virtual eXtensible LAN (VXLAN) is a L2 in L4 (MAC-in-IP/UDP) encapsulation protocol and it is currently the prevalent overlay encapsulation in use in the datacenter space. Networking vendors have extensively adopted VXLAN as an open standard [54]. Just like FabricPath, VXLAN addresses all the STP limitations. However, with VXLAN a 24-bit number identifies a virtual network segment, thus enabling support for up to 16 million broadcast domains, overcoming the traditional limitation of 4k VLANs imposed by 802.1Q.

Because VXLAN runs over an IP network (L3/L4 underlay), the ECMP feature of Layer 3 networks is innately available for use. In general, an overlay such as VXLAN running on top of Layer 3 can use hierarchical addressing with IP and any transport.

With VXLAN and its use of IP, DCNs have moved from being transport-dependent to transport-independent: this implies that the previous flat MAC-based addressing scheme for the underlay has moved to a hierarchical IP-based addressing scheme.

The border switches in a VXLAN network are called edge devices, and they host the VXLAN Tunnel Endpoint (VTEP). The edge switches are responsible for encapsulation and decapsulation of the VXLAN header. The core switches interconnecting the various VTEPs are regular IP routers. Remarkably, these do not need specialized hardware or software. Moreover, the switches within a VXLAN network learn about each other using regular routing protocols such as OSPF, Layer 3 IS-IS, and so on.

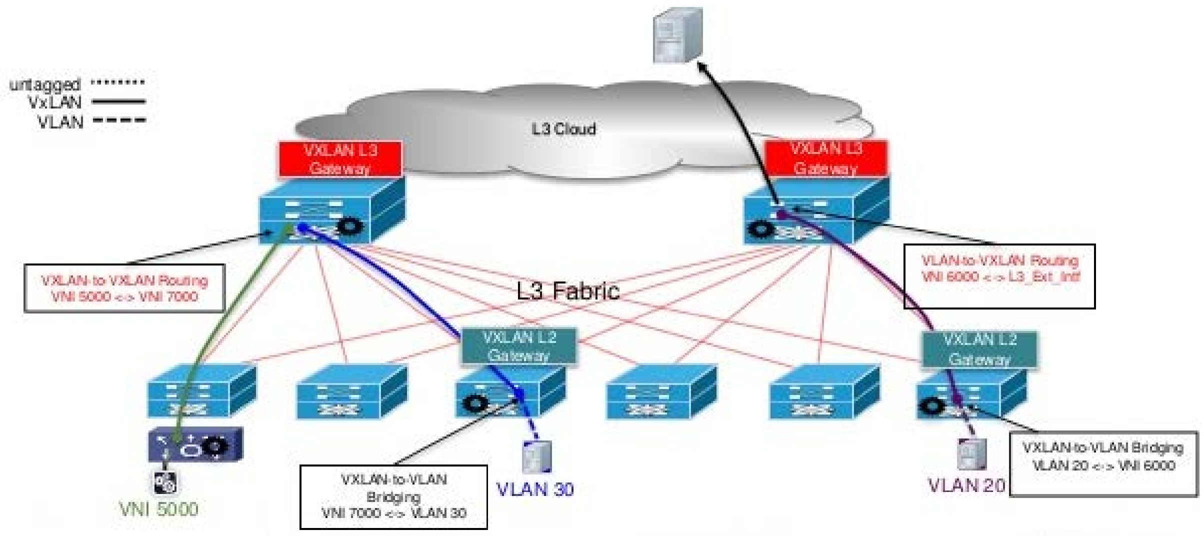

untagged
VxLAN
VLAN
L3 Cloud
VXLAN L3 Gateway
VXLAN L3 Gateway
VXLAN-to-VXLAN Routing VNI 5000 <-> VNI 7000
L3 Fabric
VLAN-to-VXLAN Routing VNI 6000 <-> L3_Ext_Intf
VXLAN L2 Gateway
VXLAN L2 Gateway
VNI 5000
VXLAN-to-VLAN Bridging VNI 7000 <-> VLAN 30
VLAN 30
VLAN 20
VXLAN-to-VLAN Bridging VLAN 20 <-> VNI 6000

Figure 6-7 - VXLAN communication

VXLAN is one of the few protocols that can provide both host or a network overlay. This implies that the encapsulation/decapsulation of the VXLAN header can be performed both from VXLAN-capable network switches and/or the server hosts themselves (vSwitch or hardware VTEP in the NICs). This enables highly flexible implementations with transparent physical-to-virtual integration.

6.3.4.3 Other overlays

Many other overlays/encapsulation protocols exist/are used in the datacenter networks, notably

- Generic Protocol Encapsulation (GPE) [55]
- Network Service Header (NSH) [56]
- Multi-Protocol Label Switching (used, amongst others, in Azure Cloud) [57] (See par. 16.3)
- Overlay Transport Virtualization (OTV) [58]

However, currently, VXLAN is the de facto (software or hardware based) standard overlay protocol for datacenter deployments[9].

[9] It should be noted, though, that MPLS is gaining popularity in the Datacenter space.

6.4 Virtualization of the network processes

All the software functions processed by the switch CPU (found in the supervisor) are included in the switch control plane providing crucial software functions such as the routing information base, running of various Layer 2 and Layer 3 protocols, and many others. The control plane runs many different process, manages the data plane and enables many hardware-accelerated features. These software processes could be "virtualized" to extend their scope and application domain. The two main techniques to virtualize these processes are virtual Device Contexts (vDCs) and Virtual Router and Forwarding (VRFs) [10]

6.4.1 vDCs

Some DCN switches support virtual device contexts (VDCs). In its default state, the switch control plane runs a single resource pool, called device context (VDC 1) within which, in a typical DCN switch, approximately 80 processes will run. Some of these processes can have other threads spawned, resulting in as many as 250 processes actively running on the system at a time depending on the services configured [59].

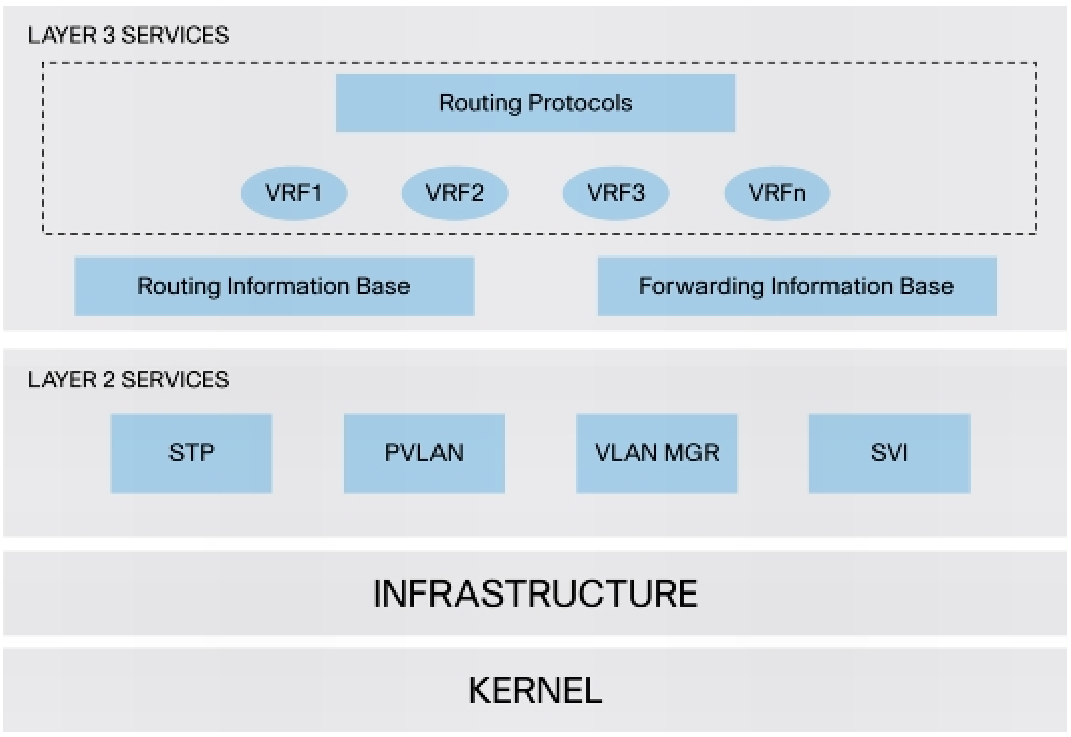


Figure 6-8 - Default Operating Mode with Single Default VDC [59]

This set of processes constitutes what is the control plane for a single physical device. At the higher level, the virtual Device Contexts are defined, within every VDC, multiple VRFs and VDCs can be defined.

[10] Equivalent features might be called with different names by other manufacturers.

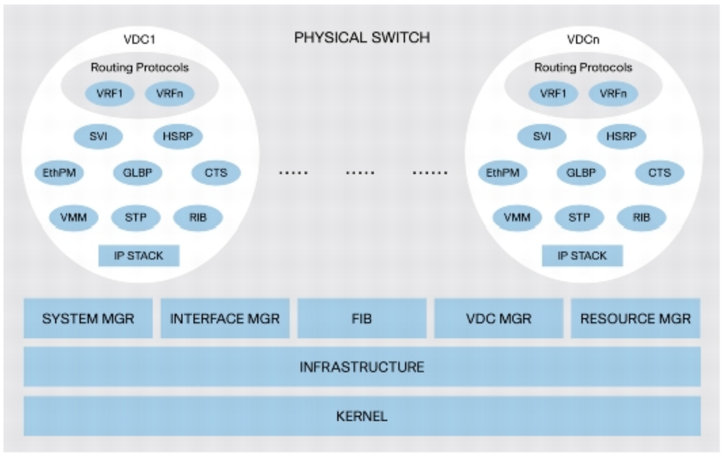


Figure 6-9 - vDC Mode [59]

A VDC can be used to virtualize the device itself, presenting the same physical switch as multiple logical devices. Every VDC can contain its own unique and independent set of VLANs and VRFs. Each VDC can have assigned to it physical ports, thus allowing for the hardware data plane to be virtualized as well. Within each VDC, a separate management domain can manage the VDC itself enabling the virtualization of the management plane itself.

6.4.2 VRFs

Virtualization of the routing/forwarding processes is supported through virtual route forwarding instances (VRF) (see [60]). A VRF can virtualize forwarding and routing tables allowing the co-existence of multiple instances of a routing table in a device (router, L3 switch). This allows network paths segmentation without using multiple devices

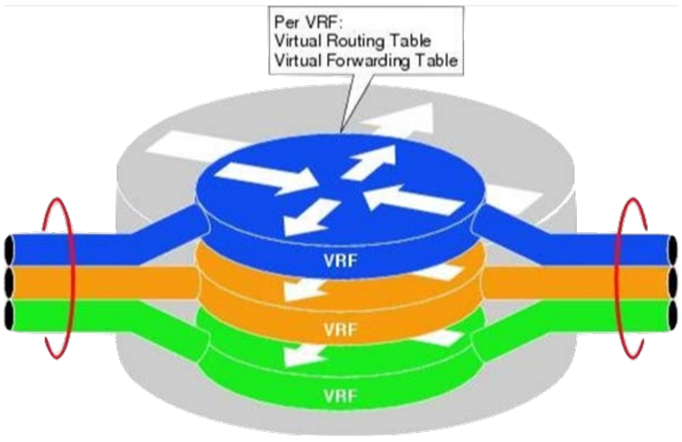


Figure 6-10 - Multiple VRF instances inside the same physical Router

VRFs behave like logical routers, but a VRF instance uses a single routing table, whereas a logical router includes several tables. Moreover, VRFs require a set of rules and routing protocols controlling the packets forwarding and a forwarding table designating the next hop for each data packet. These tables enforce the logical segregation inside the single VRF and keep out traffic that should remain outside the VRF path.

7 Software Defined Networking

7.1 Abstract

As mentioned in the previous chapters, the modern datacenter has pushed traditional networking to the breaking point. This chapter describes the virtualization and the centralization of the control plane and its separation from the data plane, presents the classical and alternative definitions of Software Defined Networking (SDN), briefly introducing three SDN Architectures (Big Switch Big Cloud Fabric® , Juniper Contrail® , Cisco ACI® that will be used in Section III to build different designs for the Enterprise DCN (for an introduction on SDN, see [61])

7.2 Data, control and management planes in network switches

A brief introduction of the architecture model of network device is useful to understand the key features and the core concepts of SDN

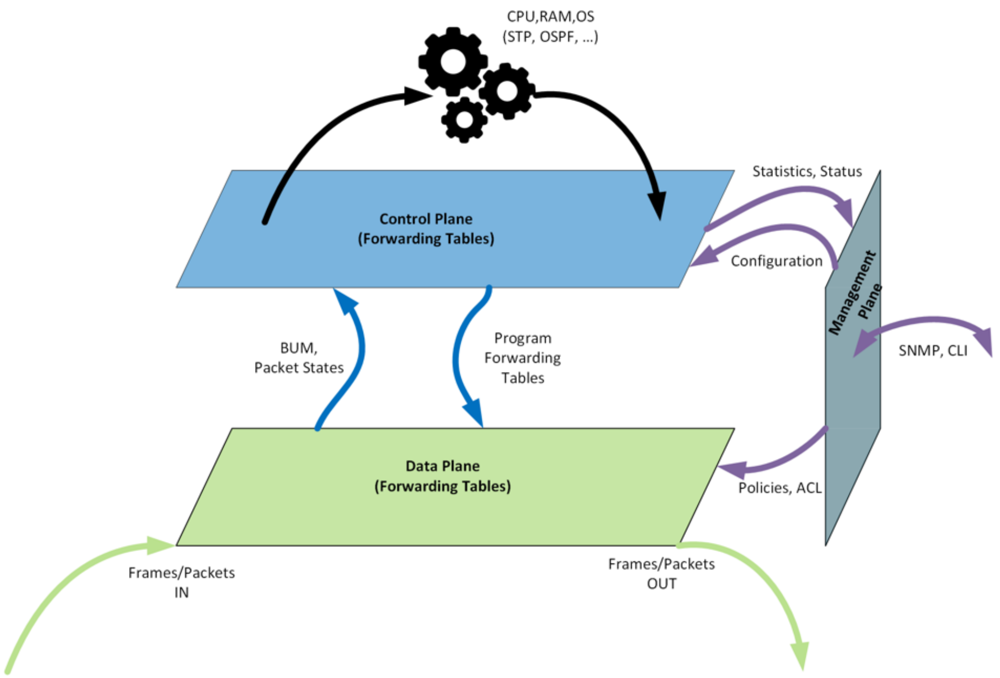


Figure 7-1 - Roles of control, management and data plane

As shown in Figure 7-1, a network device can be represented by three planes, with different roles and responsibilities:

- Data (aka forward) Plane: handles autonomously most of the frames/packets. The data plane consists of various ports that receive/transmit frames and packets, and contains the forwarding table needed to perform this communication without the other planes.
- Control Plane: not all packets can be handled only by the data plane, sometimes simply because the forwarding table doesn't contain all the required information, or because

they belong to a control protocol that must be processed by the control plane. The control plane main responsibility is to update the forwarding table to maximize the amount of traffic that the data plane can autonomously handle.

- Management Plane: the switch is managed, configured and monitored through this plane, which also extracts information from or modifies data in the control and data planes.

7.3 Why SDN?

The traditional model with the control plane distributed on all the switches fails to scale to the size and complexity of many modern deployments and introduces (extra) complexity that was originally needed to provide redundancy and distributed intelligence required improve the overall resiliency. The most important reasons for this failure have been discussed in Chapter 3 and could be summarized as:

1. Inability to scale up to the size of modern mega-datacenters
2. The TCO of networking equipment is elevate when compared to other equipment
3. A disconnect between the rate of innovation in the areas of compute and storage virtualization as compared to networking.

The datacenter has been the place where these failings have emerged first and most severely, and in the DCN the pressure to provide a different, modern, approach drove the design and implementation of different solutions overcoming the presented issues.

The separation between Control and data planes has been a recurring topic during the past, not only in the Datacenter area, so in the next paragraph some forerunners of SDN will be briefly introduced. The purpose of this brief list is to show how SDN is not just a sudden trend on the market, or a nice catch/buzz word used in pre-sale meetings, but It is actually the result of a historic process which emerged more than twenty years ago.

7.3.1 Forerunners of SDN

Even before the advent of the expression *Software Defined Networking*, avant-garde researchers were planning radical changes to the networking model. Table 7-1 shows some of those early implementations of what was to become SDN.

Year	Project	Description	Ref.
1997	DCAN	Separation of forwarding and control planes in ATM	[62]
1999	Open Signaling	Separation of forwarding and control planes in ATM	[63]
Late 1990s	Active Networking	Separate control and programmable switches	[64]
Late 1990s	IP Switching	Control layer two switches as a layer three routing fabric	[65]
Late 1990s	MPLS	Separate control software establishes semi-static forwarding paths for flows in traditional routers	[66]
2003	ForCES	Separating the forwarding and control planes	[67]
2005	4D	Control plane intelligence located in a centralized system	[68]
2007	Ethane	Complete enterprise and network access and control using separate forwarding and control planes, and utilizing a centralized controller	[69]
2008	Orchestration	Use of SNMP and CLI to help automate configuration of networking equipment	
2010	RADIUS, COPS	Use of admission control to dynamically provision policy	[70] [71]
2011	Virtualization Manager	Use of plug-ins to perform network reconfiguration to support server virtualization	[72]

Table 7-1 - Forerunners of SDN (summary from [61])

Just looking at the timeline it is quite clear that, during the last twenty years, the ideas around advancing networking technology steady progressed to what we call SD today.

One interesting point is that the idea of separating control and data plane has been brought back to life during the early development of ATM, and it is intimately connected to the circuit (cell) switching approach adopted by ATM. The same approach has been later applied by the precursors of L3 Switching solutions (based on ATM concepts as well).

The separation of control and data planes is part of the evolution of the control plane itself, and this evolution had also another main branch: the evolution of the network equipment, which is briefly presented in the next paragraph. Eventually, these two processes converged to what we call Classical/SDN with control and data plane separation.

7.3.2 The Evolution of Network Switches

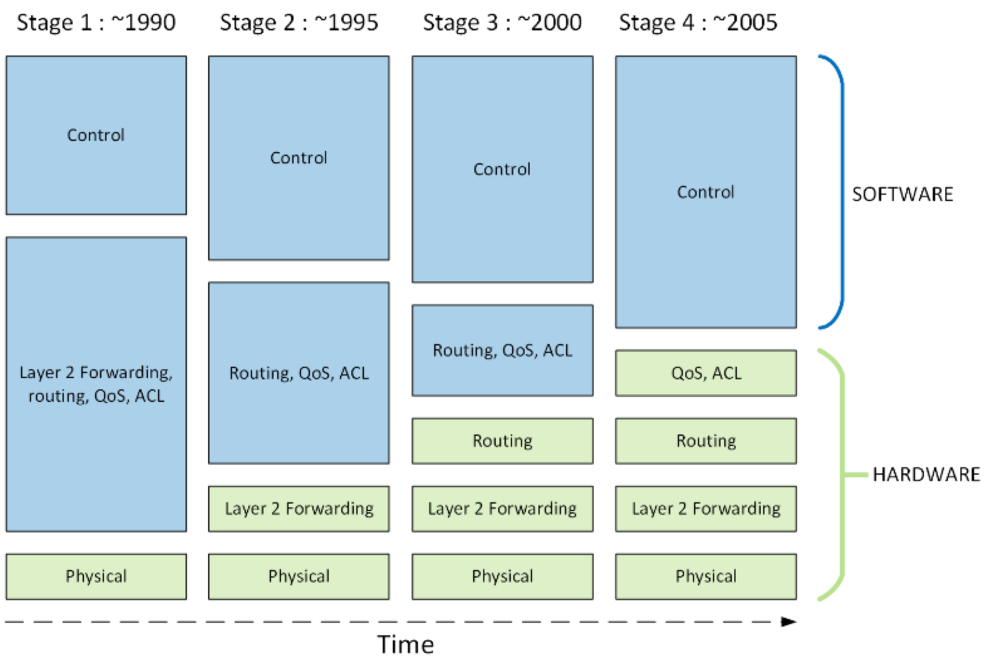


Figure 7-2 - Switching and networking functions migrating to Hardware [61]

Different components of the network switches moved to ASICs in the past 20 years. Four stages can be identified in this migration:

1. Early stage: even the simplest of tasks, such as MAC-level forwarding decisions were performed by software running inside bridges, switches. This was true even through the early days of the Internet explosion in the early 1990s.
2. Stage 2: Layer 2 Forwarding moves to Silicon. Switches started to be composed by ASICs, FPGAs and TCAMs. The combination of these innovations made possible moving Layer 2 forwarding decisions into hardware, massively improving the forwarding capabilities.
3. Stage 3: Routing becomes L3 Forwarding. In the late 1990s, Ipsilon Networks utilized GSMP (General Switch Management Protocol) to manage ATM connections within their IP Switch product [73]. The Ipsilon IP Switch showed “standard” Internet router interfaces externally, but its internal switching fabric utilized ATM switches for persistent flows. Flows were defined as a somewhat persistent stream of packets between peers identified by the 4-plet (Source, Destination IP addresses and Source/Destination TCP/ UDP port).
4. Stage 4: with the further evolution in ASICs, TCAMs, classifying features needed for QoS management and frame/packet evaluation at line speed started to be available, so both QoS and ACL management started to be implemented in Hardware.

At the end of this process it is quite clear that the separation between control and data plane inside the switch is mature enough to allow the virtualization of the control plane, which already happens in many traditional high-end switches. The supervisor on these switches (i.e. Cisco Nexus, 68xx) can already be virtualized and it is usually located on dedicated modules of the chassis. The next, natural, move would be to completely re-locate the supervisor on a different box and share it amongst different devices.

7.4 What is SDN?

in the traditional approach the Control Plane is the logic, implemented by software running on all the switches, that determines optimal paths and responds to outages and new networking demands. In the previous paragraph, we have seen that the push to centralize the control plane, and to virtualize the supervisor, has been in the market for the last 20 years.

SDN emerges in this context with the main goal of moving the control software from the devices to a centrally located resource, called SDN Controller, capable of seeing the entire network and making decisions which are optimal, given the holistic view of the situation. Basically, SDN is about moving from a distributed control plane co-resident with the data plane to a centralized control plane located on a system that connects, manages, and programs the distributed data planes located on simplified (White box) switches.

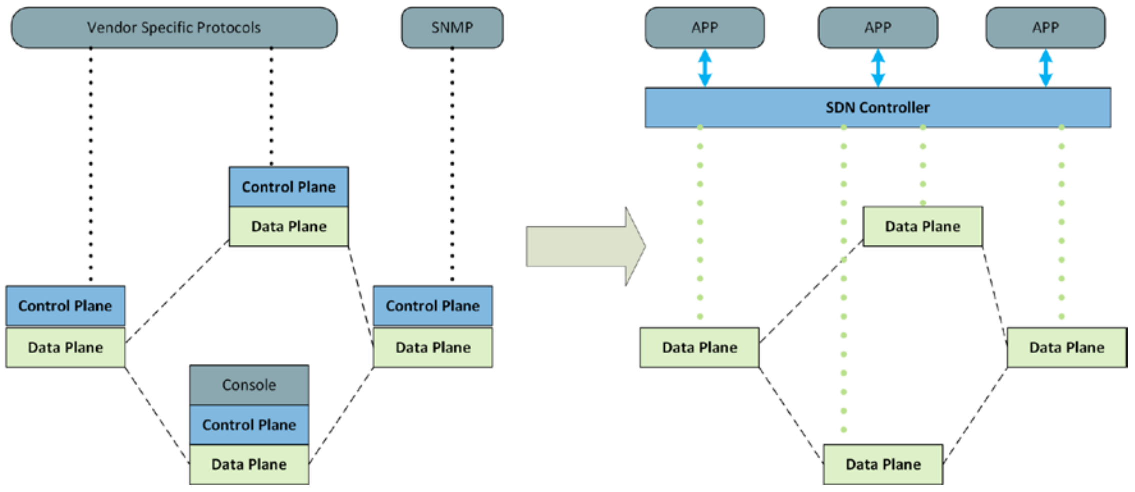


Figure 7-3 - From Traditional Networking to Classic SDN

Classic SDN attempts to separate network activities in the following manner:

- Forwarding, Filtering, and Prioritization responsibilities, implemented in hardware tables, remain on the Data Plane of the device. Other features, like Access Control Lists to prioritize data flows, are also enforced on the device and usually implemented in hardware.
- The Control software is removed from the networking device and placed in a centralized controller, which has a holistic view of the network and it is capable to perform optimal routing decision and correspondingly program the distributed data planes of the devices.

With the control plane no longer embedded, closed, strictly coupled with the hardware, the forwarding hardware can be programmed by external software on the controllers: this triggers a “*migration to a programming paradigm for the control plane* “ [61]

- The network applications run on the SDN controller, implementing higher-level functions and participating in managing and controlling packet forwarding within the network.

7.4.1 The Five Goals of SDN

The authors, in [61], identify five main goals for SDN:

1. Separate Control and Data Planes: the control plane leaves the switches.
2. Centralize the Control: the control is now centralized in the SDN Controller.
3. Simplify the Device: the switches do not have any more a control plane, they just need to communicate with the SDN Controller through Southbound protocols.
4. Virtualize and Orchestrate the Network: the network resources are fully virtualized: their availability is independent from the location and the physical device providing them. Moreover, the Datacenter Orchestrator platforms, using northbound protocols, could interface with the SDN controller and drive the necessary changes on the network.
5. Ease the development (Openness): northbound and southbound protocols used in SDN should be open, to facilitate the interaction with Orchestrator platforms and eliminate/reduce the risk of vendor lock-in.

The five goals above are somewhat general across all the SDN Solutions, however the degree by which they are implemented/achieved vary in each solution. For these reason, a possible classification of SDN is presented in the next paragraph

7.4.2 SDN Classification

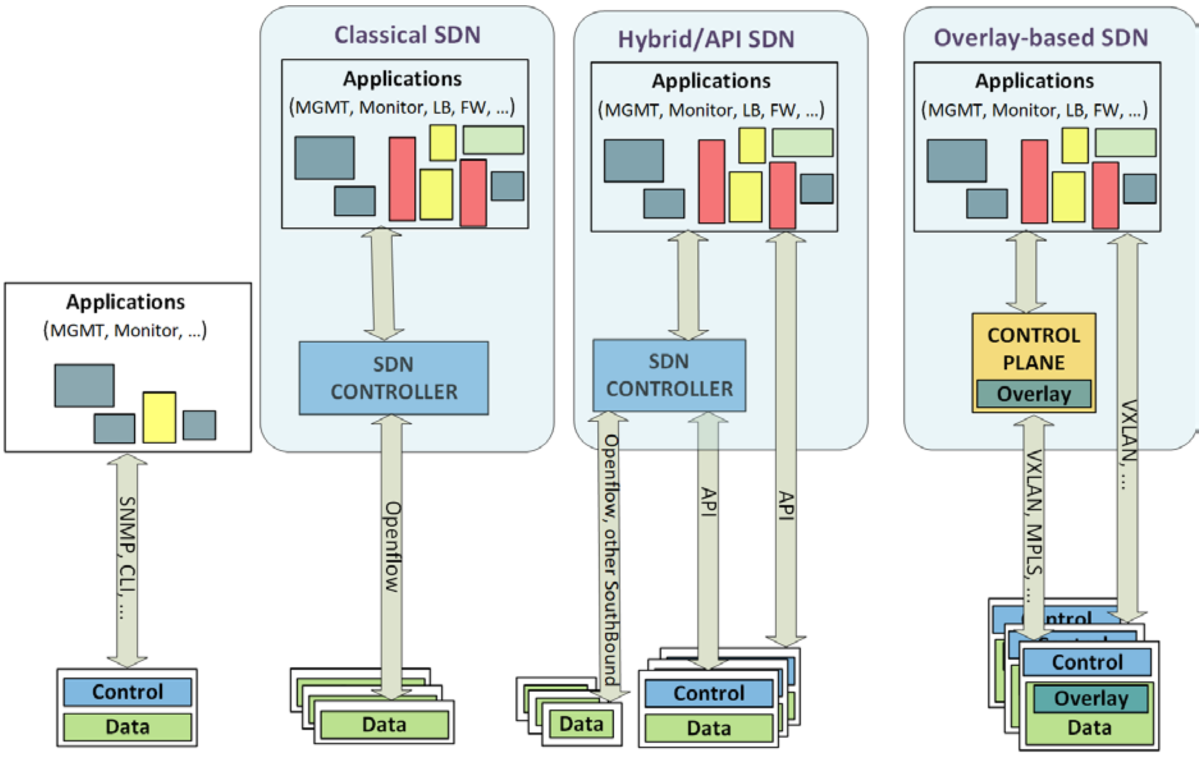


Figure 7-4 - Classification of SDN

There are different approaches to SDN that accomplish, to different extents, the goals introduced in Par. 7.4.1. and there are different ways to classify them. The classification shown in this paragraph is focused on the specific needs of the Enterprise market for DCN and primarily based on the work presented in [61]:

- Classical/Open SDN: a flow-based, imperative, approach with a control plane fully centralized on the SDN controller, using OpenFlow as Southbound protocol. Classical/Open SDN is described in in par 7.5
- SDN via Overlay Networks (Hardware based, or hypervisor based): the control plane is distributed, and Applications invoke directly functions on the Network devices via traditional protocols, such as SNMP, CLI, or NETCONF. Alternatively, newer mechanisms such as RESTful APIs or other Southbound protocols are used (VMWare NSX uses OpenFlow, Cisco VXLAN uses BGP/EVPN, etc.) may be used. SDN Via Overlay is described in Par. 7.6
- Hybrid SDN/Programmable API: a class of SDN solutions where different Southbound Protocols coexist with OpenFlow, and where some functions of the control plane are still distributed on the network devices. Hybrid SDN is introduced in par. 7.7

7.5 Classical SDN

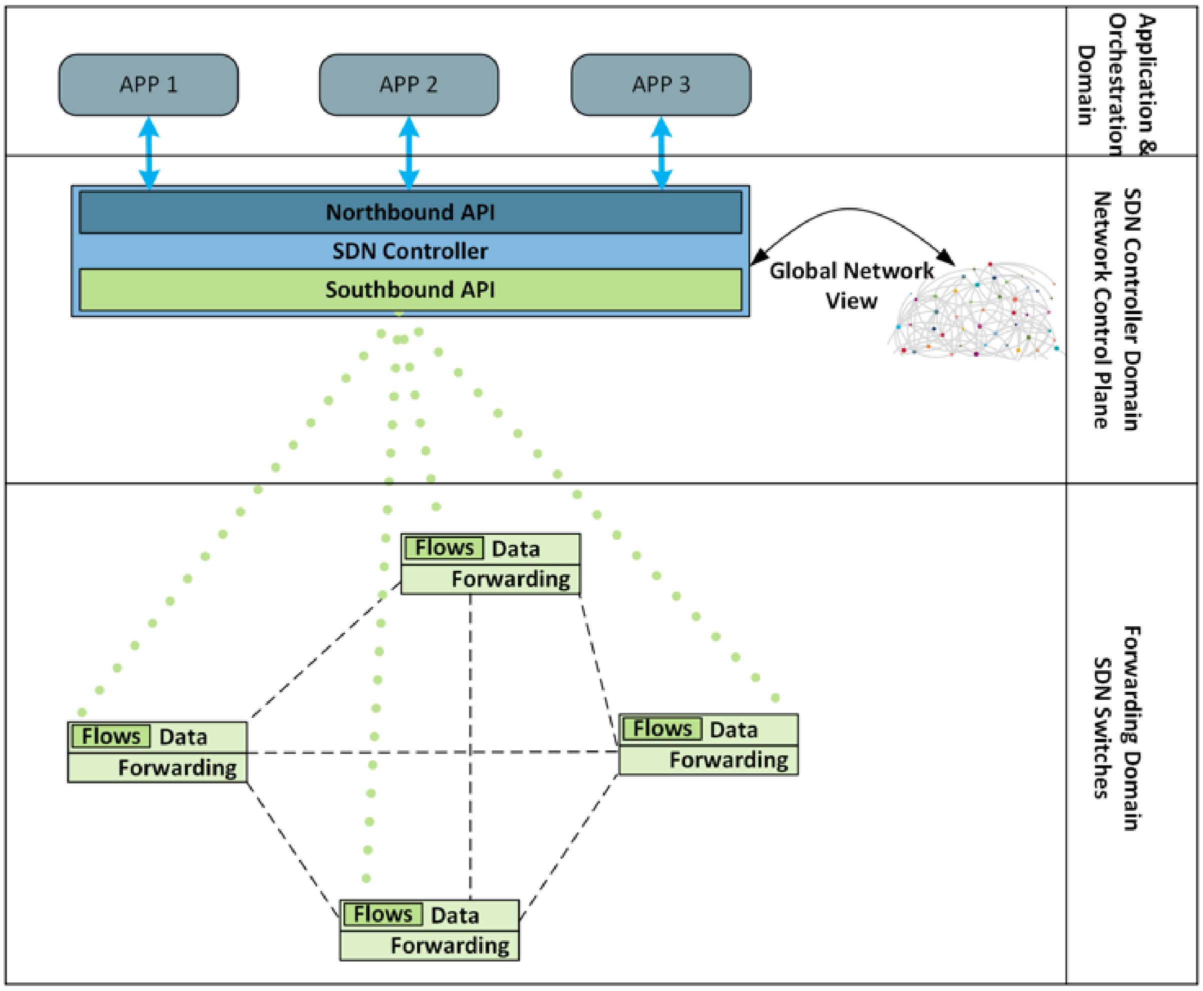


Figure 7-5 - Overview of SDN Operations

From a high-level perspective, the operations on a SDN could be categorized in three different domains:

- Forwarding Domain
- SDN Controller Domain
- SDN Applications and Orchestration Domain

7.5.1 Forwarding Domain: SDN Switches (using OpenFlow)

The SDN Switches forward the frames based on the content of its Data Block, containing the Flow Tables populated by the SDN Controllers using OpenFlow as a Southbound protocol [74]. A SDN Switch is composed by three blocks (see Figure 7-6):

- Forwarding Block, responsible of forwarding frames between two ports
- Abstraction Layer, containing the Flow tables populated by the SDN Controller
- Southbound API, the module in charge of communicating with the SDN Controller using OpenFlow as a Southbound protocol

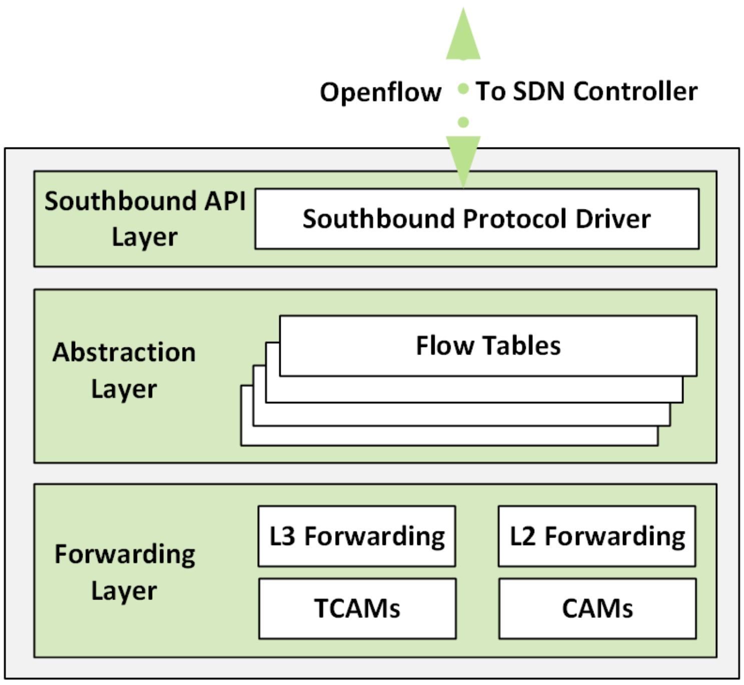


Figure 7-6 - SDN Hardware Switch Block Diagram

7.5.1.1 Southbound API Layer

The Southbound API Layer is responsible of the communication with the SDN Controller, and the update/creation of the Flow Tables is based on the commands received from the SDN Controller.

7.5.1.2 Abstraction Layer and Flow Tables

The abstraction Layer contains one or more Flow tables (populated by the SDN Controller through the Southbound Protocol Driver). The abstraction layer translates the Flow tables into commands for the underlying forwarding layer (machine code for a hardware switch and API calls for a Soft Switch).

A flow is defined by a stream of packets transferred by one endpoint (or a group of endpoints). The definition of endpoint may vary with the Southbound protocol in use, for instance an endpoint could be a 4-tuple (Source, Destination IP Addresses and Source, Destination TCP Ports), or a VLAN tag, a hostname, LLDP information, or many other metadata extracted from the packet (i.e. some controllers provides the ability of defining a flow using Application-level information).

Flow tables are the core data structures in a SDN switch: they allow the device to evaluate incoming packets and take the appropriate action (i.e. drop, flood, forward, mark) based on the contents of the packet that has just been received.

Flow tables contain many prioritized flow records, each usually containing of two entries:

- match fields: the inbound packet is compared against the match field chain, and the first full match is selected
- actions: Actions are the directives that the switch should implement if the inbound packet matches the match fields specified for this entry

Match Field	Action

Figure 7-7 - Flow Table Entry

If a match is found, the incoming packet is processed locally, unless the action associated with the match requires that the packet is explicitly forwarded to the controller.

When no match is found, the packet may be sent to the controller for additional processing. This scenario is defined as the controller *consuming* the packet. In the case of a hardware switch (i.e. white box switches discussed in Chapter 7.4) the specialized hardware implements these mechanisms. In the case of a software switch (i.e. available on a hypervisor) these same functions are mirrored by software.

7.5.1.3 Forwarding Layer

As already noted, the forwarding layer could be implemented either in hardware or software. Its responsibility is to forward the frames/packets. The forwarding actions are implemented by the Packet processing module in case of a vSwitch, or by the switch logic connecting the TCAM/CAM/ASICs, whose status is updated by the Abstraction layer, in a hardware switch.

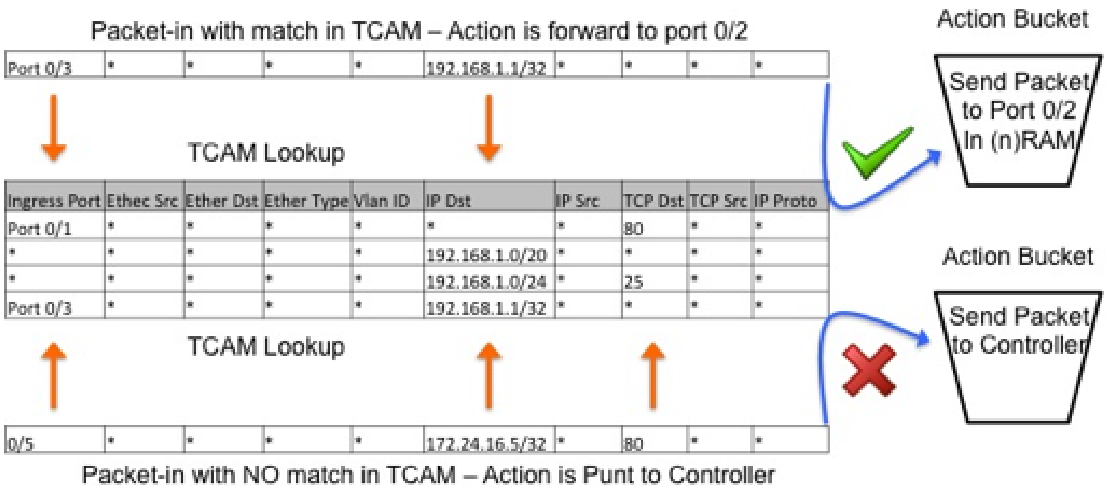
In its early life, OpenFlow provided just a way to use the TCAM tables in existing switches to implement any flow a programmer wanted. Now, using OpenFlow on any of the 10 fields can define a match and the conforming action space can be selected between an ever-increasing set (DROP, forward, process, copy, etc.).

7.5.1.4 Network Abstracted model

The mechanism described in the previous paragraph could be a sort of enhanced, centralized, policy-based routing/forwarding system, decoupling control and data plane, and pushing a set of flow policies from outside the switch, through a TCP socket, using a messaging protocol that defines the policy and flow rules that should be put in the TCAM.

Modelling the switch as a set of flow tables, match field and actions, and programming these objects using a standard protocol, the network can be defined abstracting from its inherent complexity: this level of abstraction, control, centralization is one of the key element of the Classical SDN approach.

7.5.1.5 Proactive vs reactive Flows


Packet-in with match in TCAM – Action is forward to port 0/2
TCAM Lookup
Action Bucket
Send Packet to Port 0/2 In (n)RAM
Action Bucket
Send Packet to Controller
TCAM Lookup
Packet-in with NO match in TCAM – Action is Punt to Controller

Port 0/3	*	*	*	*	192.168.1.1/32	*	*	*	*

Ingress Port	Ethec Src	Ether Dst	Ether Type	Vlan ID	IP Dst	IP Src	TCP Dst	TCP Src	IP Proto
Port 0/1	*	*	*	*	*	*	80	*	*
*	*	*	*	*	192.168.1.0/20	*	*	*	*
*	*	*	*	*	192.168.1.0/24	*	25	*	*
Port 0/3	*	*	*	*	192.168.1.1/32	*	*	*	*

0/5	*	*	*	*	172.24.16.5/32	*	80	*	*

Figure 7-8 - TCAM Matching [75]

When using OpenFlow to populate the switches' TCAM there are essentially three modes of operation:

1. Reactive flow instantiation: when the switch identifies a new flow, the OpenFlow Driver performs a flow table lookup, either a search on a ASIC (in case of a hardware switch) or a table lookup (in case of a Software Switch). If there is no match for the flow, the switch contacts the controller for instructions: in this mode the switch reacts to traffic, consults the controller and creates a rule based on the instruction. The problem with reactive mode is that today's hardware does not have enough CPU power. HW switches outside of NPUs or general-purpose CPU vSwitches don't do this over a few thousand packets per second.
2. Proactive flow instantiation: instead of reacting, an OpenFlow controller might populate the flow tables proactively. Using this approach, the packet-in with NO match event does not occur: when the flow table is in the TCAM, all packets can be forwarded at line rate just by a lookup in the flow tables. The proactive instantiation removes the latency caused by consulting the controller on every new flow.
3. Hybrid flow instantiation: the combination of proactive and reactive instantiations provides the flexibility of reactive for specific cases and, at the same time, preserves low-latency forwarding for other pre-populated flows. As an example, in the financial sector, markets operate in nanoseconds: in this case a reactive flow will be simply not applicable. However, granular security might be a core feature in some sections of an Enterprise DCN (i.e. DMZs, sensitive and segregated backends, etc.): in this case the flexibility of reactive flows might be needed.

7.5.2 SDN Controller

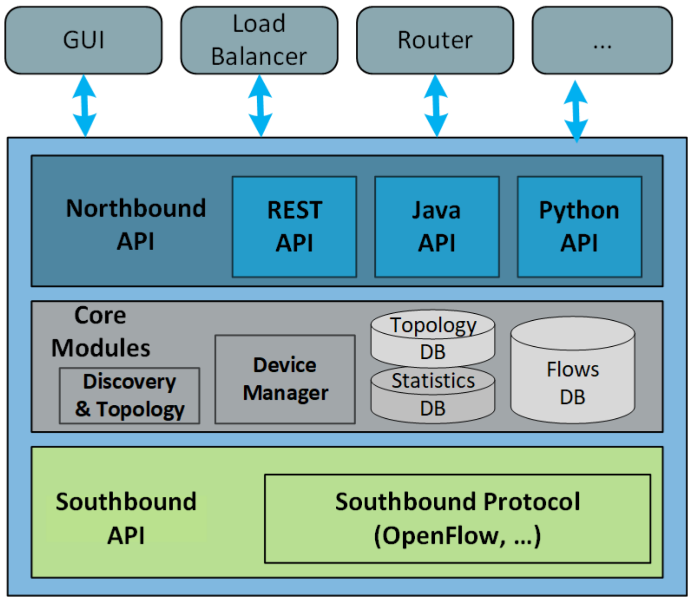


Figure 7-9 - Block diagram of a SDN Controller

The SDN controller abstracts the network of SDN Switches it controls and presents this abstracted network to the SDN applications running above it. The controller allows the SDN applications to define flows on SDN Switches helping the application to respond to packets forwarded to the controller by the SDN devices. As shown in Figure 7-9, the SDN controller in its Topology DB keeps a view of the complete network it controls. In this way, the controller can calculate optimal forwarding solutions for the network in a predictable manner.

Since one controller can control many network devices, these calculations are normally performed on a (cluster of) high-performance system(s), with an order-of-magnitude performance advantage over the CPU and memory capacity than is typically afforded by the network devices themselves.

7.5.2.1 SDN Controller Core operations

Figure 7-9 shows the different blocks composing a SDN Controller:

- Southbound layer, connecting to the SDN Switches
- Core Modules
- Northbound layer

The southbound API is used to interface with the SDN devices. This API is OpenFlow in the case of Classical SDN or some alternative such as XMPP+OVSDB in Juniper Contrail, or OPFlex in case of Cisco/ACI. In principle, the same SDN Controller can support more than one southbound API: for instance, OpenDayLight (ODL) supports different southbound protocols (OPFlex, NETCONF, BGP, OpenFlow, CLI, SNMP, etc.).

The Open Source SDN Community (http:// www.opensourcesdn.org) is proposing a northbound equivalent to the southbound OpenFlow standard. While the absence of a standard for the controller-to-application interface is considered a current deficiency in SDN, organizations like the Open source SDN group are developing proposals to standardize this.

Aside the absence of a standard, northbound interfaces have been implemented in several forms. For example, the Floodlight controller includes a Java API, and a Representational State Transfer (RESTful) API. The OpenDayLight controller provides a RESTful API for applications running on separate machines. The northbound API represents an outstanding opportunity for innovation and collaboration amongst vendors, developers and the open source community.

7.5.2.2 Northbound protocols

A crucial feature provided by the SDN controller to the SDN Application is the API to access the network. Two different interfaces usually provide this key feature:

- A low-level API, providing access to the network devices in the usual and consistent manner. In this case, the application is aware of individual devices, but is shielded from their differences.
- A high-level API providing an abstraction of the network, so that the application developers do not need to concern with discrete devices but see the network as a whole.

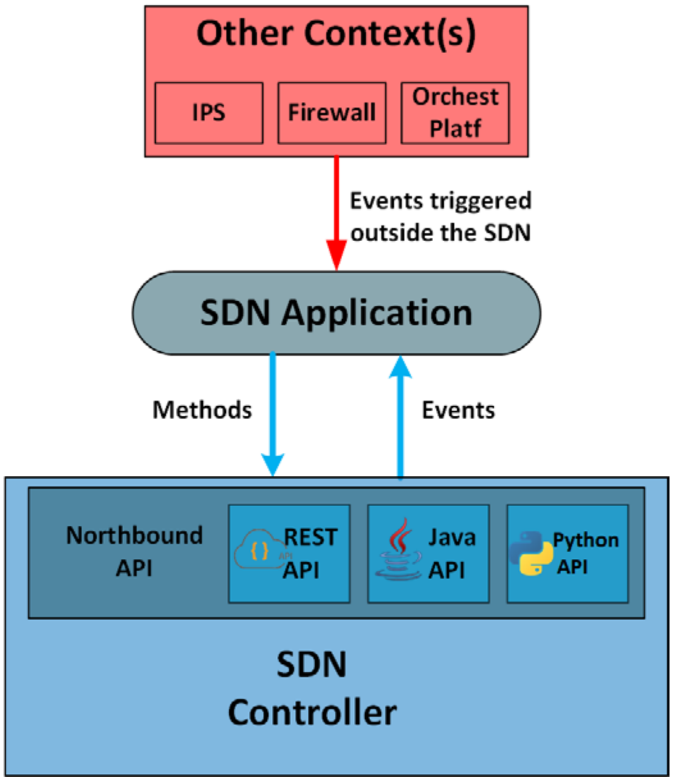


Figure 7-10 - Northbound API on Classic SDN controller

As shown in Figure 7-10 the controller interacts with the applications with two different mechanisms: triggering Events, for which the application is registered, and exposing methods used by the applications. The workflow could be summarized as follows:

- Applications subscribe to an Event (or a set of events) on the controller, and when this happens, the controller informs the application. Events may be related to a specific packet received by the controller or to a change in the network topology (i.e. a link going down).
- Upon the receipt of an event, applications may invoke different methods to affect the network operation, causing the packet to be dropped or forwarded, and/or a flow to be added, deleted or modified. Applications may also invoke methods independently without the spur of an event from the controller. The applications could receive these events from other contexts (Red box, in Figure 7-10)

Even though most of the controllers currently available in the market expose a REST API, there is currently no standardized Northbound API (also called Northbound Interface, NBI). From a market perspective, a standard NBI will have many benefits, amongst which:

- SDN application developers would no longer need to support multiple proprietary interfaces across different SDN controllers.
- SDN applications could be deployed and plugged into different controllers, in the same way different SDN switches could be used across many different controllers.
- Investments in customizing SDN applications to differentiate their own end services would be more flexible because of the ability to plug-n-play SDN controllers.

The three benefits above would have, combined, the effect of opening the market for SDN applications, pushing forward the evolution of SDN.

7.5.3 Applications and Orchestrations Domain

SDN Applications are created on top of the controller. The SDN Applications communicate with the controller using the NBI that is usually installed through some sort of driver/plugin on the application. The SDN Application interacts with the environment with two types of Flows:

1. Proactive Flows: applications using the NB-API, or the controller itself, can set proactive flows on the SDN Switches and/or receive packets forwarded to the controller. This kind of flow is known as a static flow.
2. Reactive Flows: these flows could be defined in response to a packet forwarded to the controller. Upon receiving a packet forwarded to the controller, the SDN application can

configure the controller how to respond and, if needed, could establish new flows on the switch to allow a local response in the event of other packets will be recognized on the switch. Using this method, it is possible to develop network software to implement load balancing, firewalling, routing and other functions. Other sources can also create or modify reactive flows: as an example, other data sources such as Intrusion Prevention Systems (IPS) or traffic analyzers can trigger events on the Controller causing the creation of (reactive) flows

7.5.4 Big Switch Big Cloud controller (BCF): an example of Open/SDN Solution

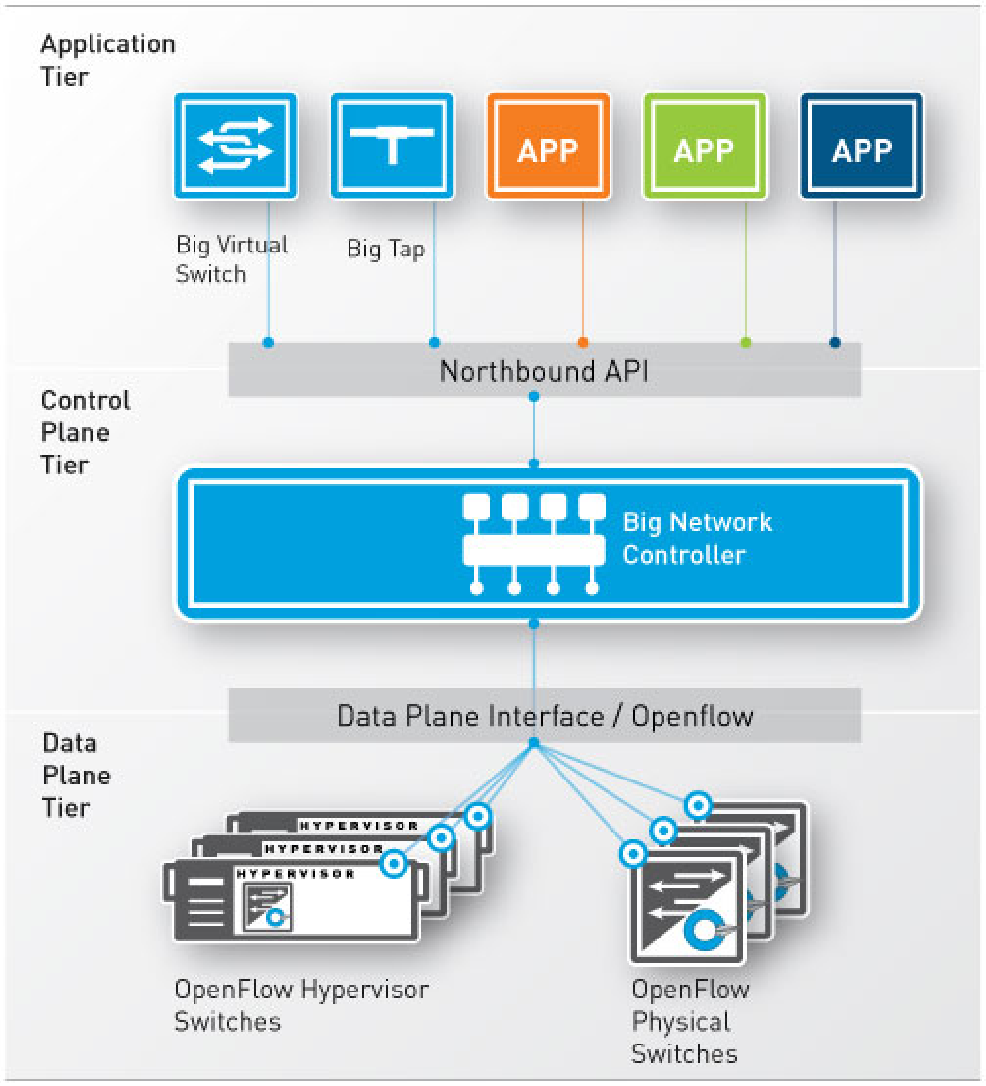


Figure 7-11 - Big Switch – Classical SDN Architecture [76]

Big Switch Big Cloud Fabric (BCF) [76] is a commercial SDN controller based on Project Floodlight [77]. BCF uses a custom version of OpenFlow as southbound Protocol, supporting both White box and vSwitches. It uses REST API as NBI: these APIs expose the learned network state to applications and enable the applications to program the network dynamically and automatically.

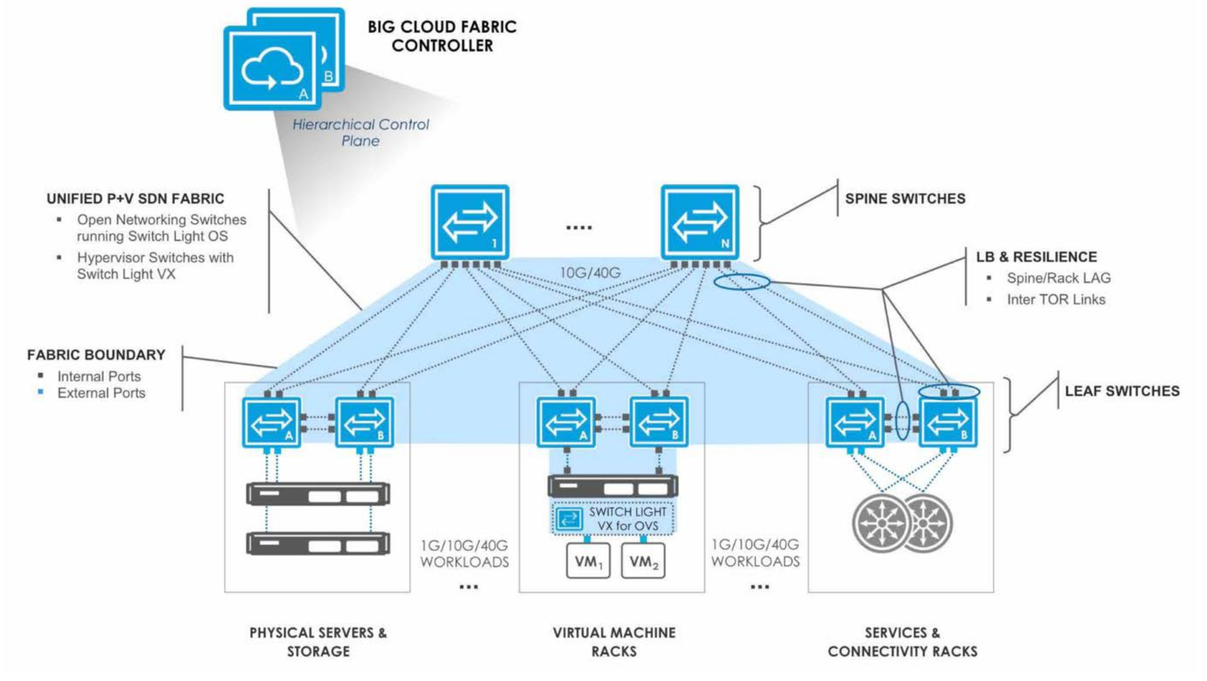


Figure 7-12 - Physical topology of a DCN based on Big Switch SDN [76]

BCF is essentially a pure SDN Controller using OpenFlow with extensions as southbound protocols and leveraging a spine-leaf architecture made with white box switches running Big Switch Light OS, that have Broadcom ASICs with several types of TCAMs. The OS running on the switches is Big Switch's Indigo OpenFlow Agent (Open Network Linux on x86 or ASIC-based hardware).

7.6 VXLAN and SDN based on Overlays

7.6.1 Introduction

Overlays add a level of indirection, abstracting the existing network technologies and extending classic network capabilities. The Fundamental theorem of software engineering states "*All problems in computer science can be solved by another level of indirection, except of course for the problem of too many indirection*s." [78]. This principle supports the concept of network virtualization overlays, briefly described in Par. 6.3.4: *an overlay is a static or dynamic network layer that runs over a physical network layer (underlay).*

In the 1990s GRE- and MPLS- encapsulations started to gain popularity and meanwhile other solutions like as 6in4, IPsec, and L2TPv3 also became popular, typically across the WAN. These encapsulations were employed either to improve security, simplify routing, or in the case of 6in4 for example, to create an IPv6 network on top of a standard IPv4.

With overlays, the original PDU (packet or frame, depending on the overlay type) is encapsulated at the source by an edge device adding an outer header, and dispatched toward the underlay/transport network to an appropriate destination edge device. The intermediate network devices in the *underlay* blindly forward the packets based on the outer header and are completely unaware of the inner payload. At the destination, the edge device strips the outer header, and forwards the packet based on the inner payload.

7.6.1.1 Main features of Overlays in the DCN

Overlays have been used in datacenter environments for about 10 years now and they are often called *network virtualization overlay*. Some specific features must be considered when considering network virtualization overlays:

1. The first and most important feature of overlays is the ability to separate the location and the identity of a host. Host Identity defines a specific host, and could be its MAC address, its IP address, and so on. Location identifies the edge device responsible of the encapsulation/de-encapsulation of the traffic for that end point[11]. The outer header of the overlay usually contains to the source and destination *locations*, and the inner header to the source and destination endpoint *identities*.
2. The other important feature is the service provided by the overlay, and this defines the overlay type and its header. Overlays are normally offered as either Layer 2 (bridging) or Layer 3 (routing) services; many modern overlays, though, provide both Layer 2 and Layer 3. The original PDU can be encapsulated into another PDU on the same layer (i.e. MAC in MAC) or on a different layer (i.e. VXLAN encapsulates L2 in UDP). This provides potentially four combinations, depending on whether a packet/frame is carried in another packet/frame, the most common combinations are
   i) If the outer header is a Layer 2 frame, the overlay approach is referred to as *frame encapsulation*. Examples of overlays that employ frame encapsulation are TRILL, and Cisco FabricPath.
   ii) if the outer header is a Layer 3 packet, the overlay is referred to as *packet encapsulation*. Examples of overlays that employ packet encapsulation are LISP, VXLAN

[11] The end points could be virtual machines, bare-metal servers, containers, or any other workload

3. With Overlays defined with different data-plane encapsulations, there is a need to define a transport service to move the data across the physical network: this service is called an underlay transport network (or simply the underlay). To define the underlay, the layer where the encapsulation occurs must be also defined. To a certain extent, the overlay type determines the transport network. As an example, in VXLAN environments, the underlay is a Layer 3 network, which transports the UDP VXLAN-encapsulated packets between the source and destination tunnel edge devices.

7.6.1.2 Classification of overlays

Network virtualization overlays can be initiated from

1. Physical servers (Host overlays)
2. Network switches connected to the servers (Network overlays)
3. Both (Hybrid overlays)

The physical servers (see Figure 7-13) are typically hypervisors running a (distributed) virtual switch/router (i.e. Distributed Virtual Switch in VMWare vSphere) that has enhanced capability of encapsulating and de-encapsulating the overlay header. This model requires the network switches to only provide connectivity between the hypervisors, which in turn permits transport of data between the virtual hosts

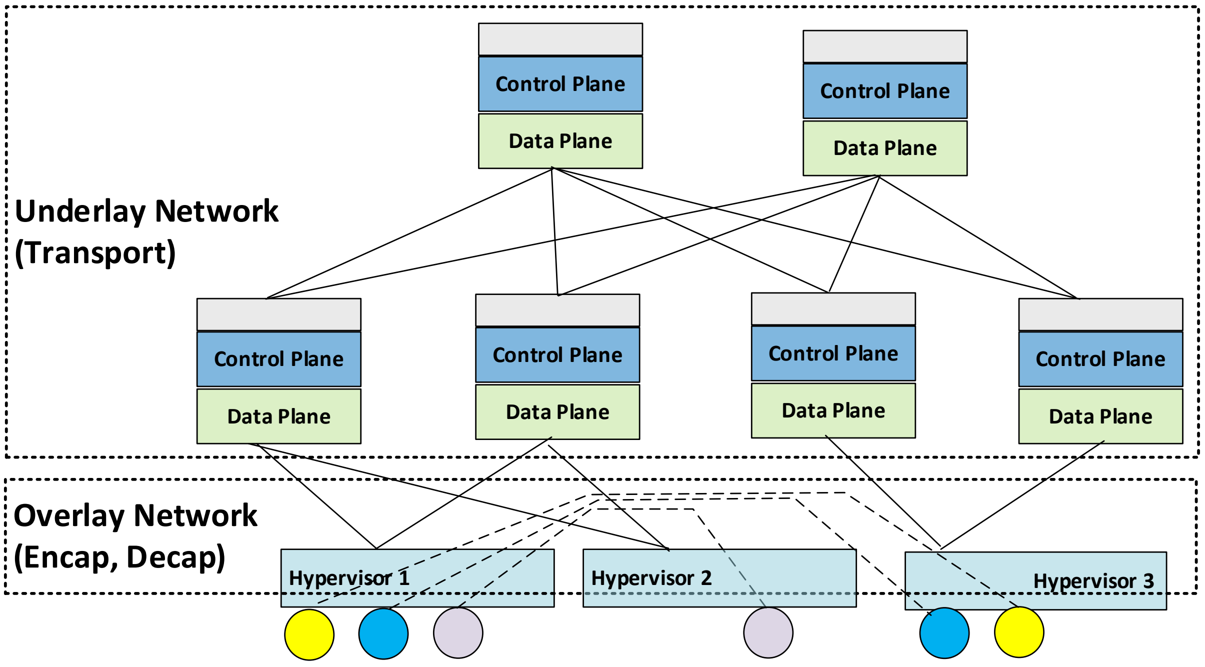


Figure 7-13 - Host Overlays

For deployments where there is a mix of bare-metal and virtualized workloads, the ToR switches take care of pushing/popping the overlay headers for all kinds of workloads beneath them. These are defined network overlays (see Figure 7-14).

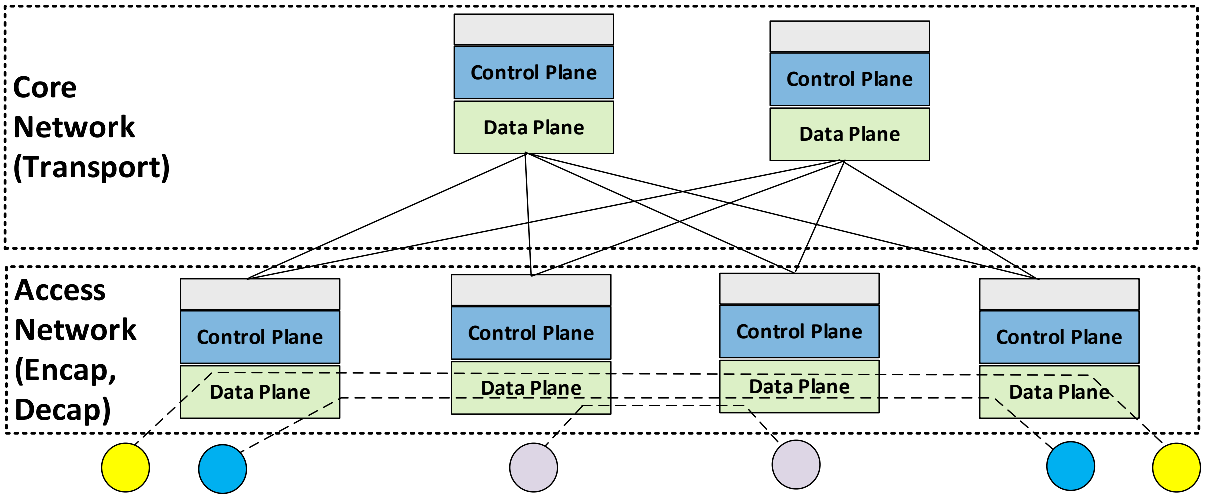


Figure 7-14 - Network Overlays

Both host overlays and network overlays are very popular and are a common deployed option to address the challenges mentioned in Chapter 3. Each has pros and cons, but hybrid overlay environments can realize the best of both worlds, supporting host and network overlays and enabling optimized physical-to-virtual (P2V) communication (see Figure 7-15)

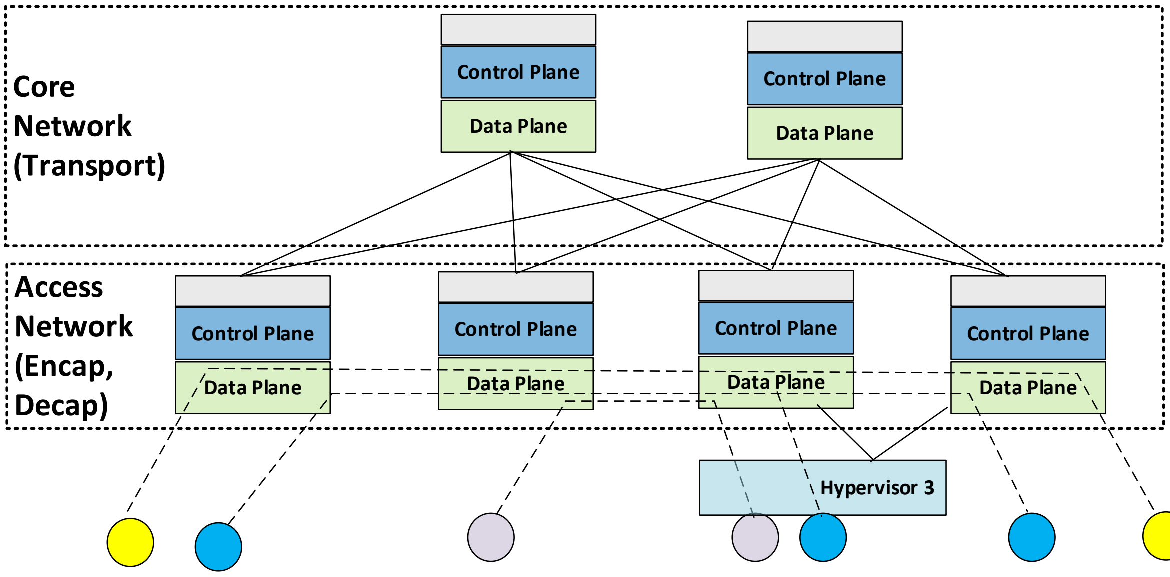


Figure 7-15 - Hybrid Overlays

7.6.2 VXLAN Key Concepts

Virtual Extensible LAN (VXLAN) is an overlay technology for network virtualization providing Layer 2 extension over a shared Layer 3 underlay infrastructure network (typically a leaf-spine pod-based Clos topology) by using MAC address in IP User Datagram Protocol (MAC in IP/UDP) tunneling encapsulation (for an introduction on VXLAN see [54], [36] and [79]). The two key VXLAN concepts are:

- VXLAN Network Identifier (VNI)
- VXLAN Tunnel Endpoint (VTEP)

7.6.2.1 VXLAN Network Identifier

Each Layer 2 subnet or segment is uniquely identified by a VXLAN network identifier (VNI) that segregates the traffic in the same way a IEEE 802.1Q VLAN ID segments traffic. As is the case with a VLAN, VMs on the same VNI can communicate at layer 2 with each other, whereas VMs on different VNIs need a L3 device to communicate with each other.

The VNI:

- Improves scalability – Although the VNI performs a similar function to the VLAN ID, it provides one big advantage over the VLAN ID: the VNI is 24 bits in length, potentially allowing more than 16 million VXLAN segments. The 12-bit VLAN ID provides only 4094 usable segments. Thus, the VXLAN protocol can support network segmentation to a size which is the square of 802.1Q's size, adequate to modern DCNs.
- Simplifies administration – In a VXLAN deployment, a VM is uniquely identified by the combination of its MAC address and its VNI. Two or more VMs can therefore have the same MAC address if they are in different VNIs, which helps simplifying the administration of multi-tenant networks.

7.6.2.2 Host-based VXLAN Tunnel Endpoint

The entity that performs the encapsulation and decapsulation of packets is called a VXLAN tunnel endpoint (VTEP).

VTEPs can be hardware-based, residing on the Leaf Switches, or software-based residing in hypervisor hosts, such as VMWare vSphere ESXi hypervisor hosts, or kernel-based virtual machine (KVM) hosts, as shown in Figure 7-16

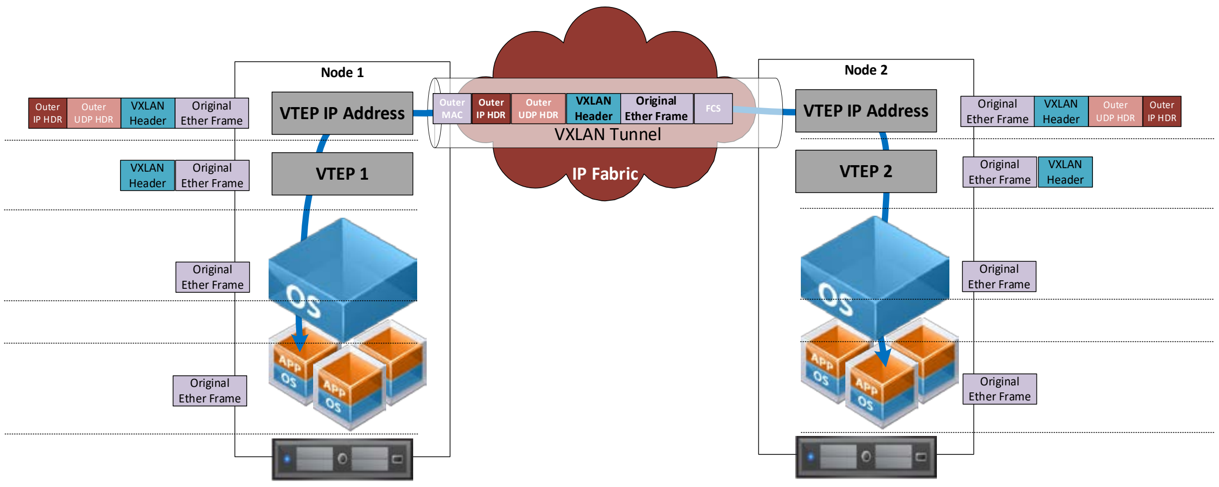


Figure 7-16 - VXLAN Encapsulation with Host-based VTEPs

Each VTEP has two interfaces: one is a switching interface facing the VMs in the host and providing the communication between VMs on the local LAN segment. The other is an interface facing the underlay network. Each VTEP has a unique IP address used to route the UDP packets between VTEPs inside the underlay IP Fabric.

As shown in Figure 7-16, when VTEP1 receives from VM1 an Ethernet frame addressed to VM3, it uses the VNI and the destination MAC to look up in its forwarding table which VTEP to send the packet to. VTEP1 then adds a VXLAN header that contains the VNI to the Ethernet frame, encapsulates the frame in a Layer 3 UDP packet, and routes the packet to VTEP2 over the Layer 3 network.

VTEP2 decapsulates the original Ethernet frame and forwards it to VM3. VM1 and VM3 are completely unaware of the VXLAN tunnel and the Layer 3 network between them.

7.6.2.3 Network-based VXLAN Tunnel Endpoint

Non-virtualized, or bare-metal, servers still exist – for example, non-x86 servers (UNIX and mainframe devices), storage (NAS, iSCSI SANs), and certain database and high-performance compute instances. These devices typically do not support VXLAN and need to continue to reside on VLAN segments.

One way to connect these devices is to place at the network edge gateways acting as VTEPs, as shown in Figure 7-17. VTEP gateways map VLANs to VXLANs and handle the VXLAN encapsulation and decapsulation so that the non-virtualized resources do not need to support the VXLAN protocol. This permits the VXLAN and VLAN segments to act as one forwarding domain across the Layer 3 boundary.

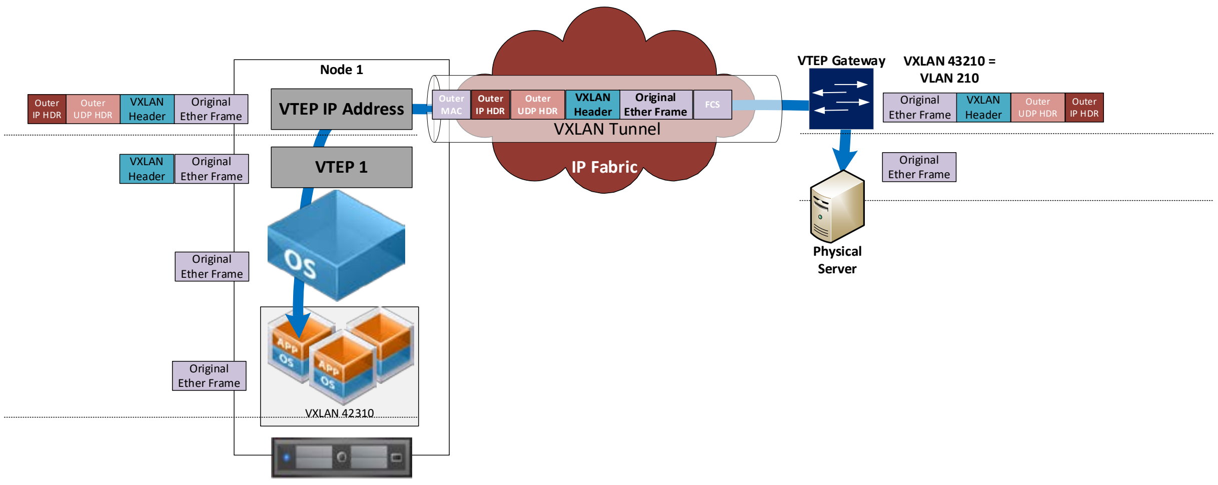


Figure 7-17 - VXLAN Encapsulation with Hybrid VTEPs

For example, as far as the Physical Server 1 in Figure 7-17 is concerned, VM1 and VM2 appear to be residing in the same VLAN segment in which it resides, VLAN 210, and the VTEP Gateways acts as a proxy translating VLAN 210 into VXLAN 43210 and vice versa.

7.6.2.4 *VXLAN pros and cons*

This paragraph briefly summarizes pros and cons of VXLAN encapsulation

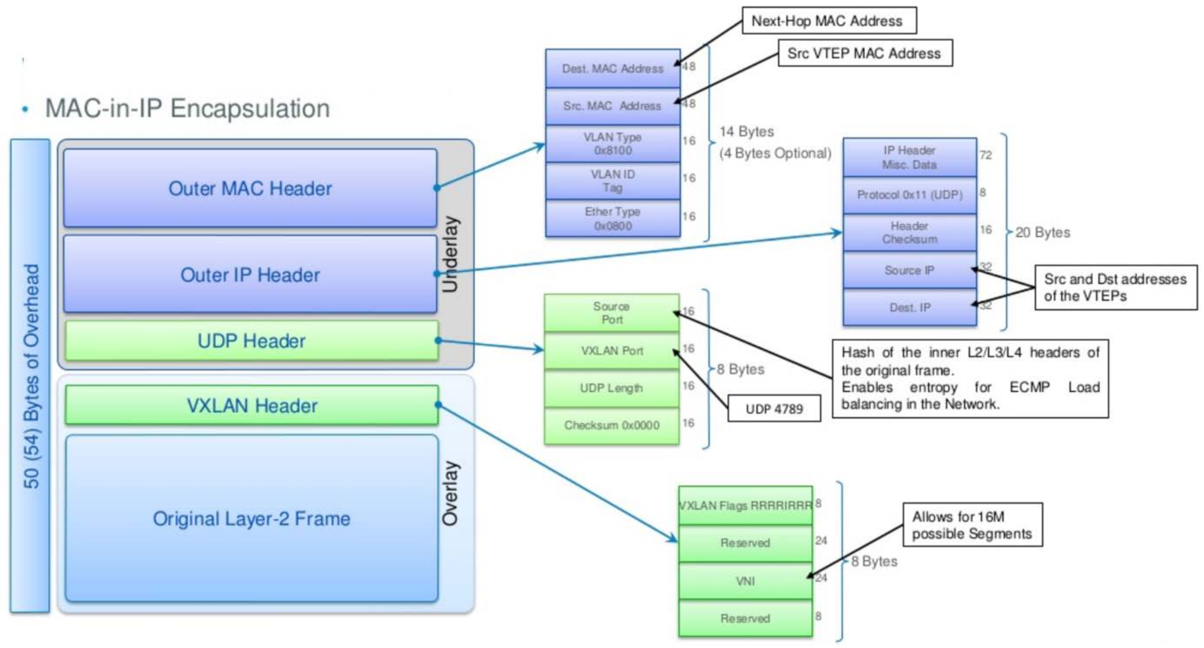


Figure 7-18 - VXLAN Encapsulation Format

Pros :

- VXLAN encapsulates the L2 frames into a UDP packet adding 54 bytes of overhead
- Leverages Layer3 ECMP taking advantage of the full network topology
- Minimizes flooding and provides Multitenancy by design
- Traditional VLANs are expressed on 12 bits (4096 possible values). VXLAN uses a 24-bit VNI identifier, which extends the number of possible Broadcast Domains to 16M
- Integrates physical and virtual workloads
- Achieve some of the goals of SDN (Control plane is still distributed, though)

Cons:

- Fabric must support 9216 bytes jumbo frames
- underlay must support multicast
- communication and compute overhead due to the need of handling the outer header

7.6.2.5 *Network Discovery and Overlays*

In the initial IETF VXLAN standard (RFC 7348) [54] the VTEP peer discovery and remote end-host learning were defined using a multicast-based flood-and-learn behavior: the overlay broadcast, unknown unicast, and multicast (BUM) traffic was encapsulated into multicast VXLAN datagrams and transported to remote VTEP switches through the underlay multicast forwarding. In this scenario, flooding could be a challenge for the scalability of the solution.

Enabling multicast in the underlay could also be problematic because some organizations do not allow multicast in the datacenter or WAN network. So, although BUM flooding through multicast in the underlay works, this technique isn't efficient and doesn't scale well, and presents some challenges as:

- there are more possible VNIs (16M) than multicast groups supported by the underlay network, so at some point it may become necessary to assign multiple VNIs to the same multicast group, which increases inefficiency.
- the use of multicast requires running PIM and IGMP in the underlay network and creates administrative overhead in managing IGMP groups, troubleshooting PIM, etc.

7.6.2.6 VXLAN control and management plane

To build a more robust and scalable overlay network, many solutions implement a control plane for VXLAN. In this section we briefly introduce the two protocols below, used in Juniper Networks Contrail:

- Open vSwitch Database Management Protocol (OVSDB)
- XMPP + Ethernet VPN (EVPN)

7.6.2.6.1 OVSDB

OVSDB [80] [81] is an OpenFlow configuration protocol designed to manage Open vSwitch implementations. OpenFlow provides the control plane, managing physical and virtual switches and determining how packets are forwarded between source and destination VMs. OVSDB provides a management plane that enable a standard way to configure and manage switches, even those from different vendors.

Usually OVSDB is combined with an overlay controller, such as Juniper Contrail or VMware NSX, that uses OVSDB to provision the VTEPs and to handle MAC address learning. The controller provides the VTEPs with MAC addresses and the VXLAN tunnels over which the addresses can be reached. In return, the VTEPs inform the controller of any MAC addresses they learn. This centralized replication of MAC addresses is much more efficient than MAC learning through multicast.

OVSDB support is not limited to virtual switches. A hardware VTEP gateway (see Figure 7-19) supporting OVSDB can participate in a controller-based overlay network.

7.6.2.6.2 EVPN

In EVPNs [82], MAC address learning is driven by the control plane, rather than by the data plane, which helps control learned MAC addresses across virtual forwarders, thus avoiding flooding. The forwarders advertise locally learned MAC addresses to the controllers. The controllers use MP-BGP to communicate with peers. The peering of controllers using BGP for EVPN results in better and faster convergence. In EVPN, traffic across different virtual networks is isolated because MAC learning is restricted to the virtual networks to which the virtual machine belongs: in this way virtual networks can share the same MAC addresses without any issue.

The result of a MAC address lookup is a next hop, which, like IP forwarding, points to a local virtual machine or a tunnel to reach the virtual machine on a remote server. The tunnel encapsulation methods supported for EVPN are MPLSoGRE, MPLSoUDP, and VXLAN. The encapsulation method selected is based on a user-configured priority.

7.6.2.6.3 XMPP

EXtensible Messaging and Presence Protocol (XMPP) is the management protocol (see [81]) used by Contrail to program the control plane of the virtual switches in its overlay networks.

The schema of the messages exchanged over XMPP is described in an IETF draft [draft-ietf-l3vpn-end-system] in which the control plan protocol specified in BGP/MPLS IP VPNs is used and extended via the XMPP protocol to provide a Virtual Network service to end-systems (hosts). These end-systems may be used to provide network services or may host end-user applications.

7.6.3 Juniper Networks Contrail: an example of Overlay-based SDN

Juniper Networks Contrail [83] is a SDN solution that orchestrates the creation, management and operations of scalable hybrid networks. It provides self-service provisioning and enables service function chaining for dynamic application environments. For an introduction on how Contrail uses VXLAN, see [84]. Contrail can be used with open cloud orchestration systems such as Red Hat OpenStack/OpenShift and private cloud solution like VMWare vSphere. It can also interact with other operations support systems (OSS) and business support systems (BSS) using northbound APIs

Contrail is a SDN overlay controller using XMPP/OVSDB as management and control plane. Contrail vRouters reside in hypervisors and use XMPP to exchange learned MAC addresses with each other and the controller. To support bare-metal servers, Contrail also uses OVSDB to exchange learned MAC address information with VTEP gateways, which might not support XMPP.

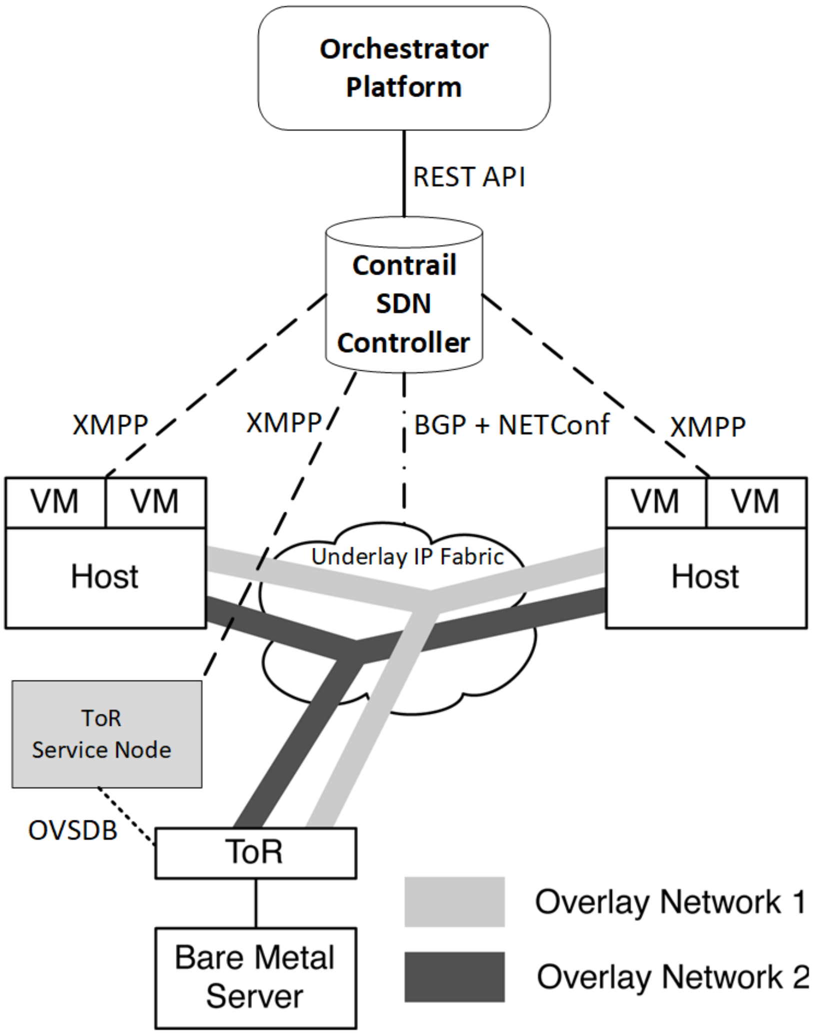


Figure 7-19 - Contrail Architecture

Contrail can use VXLAN as overlay and supports MPLS over GRE and UDP as well

7.7 Hybrid SDN / SDN on API

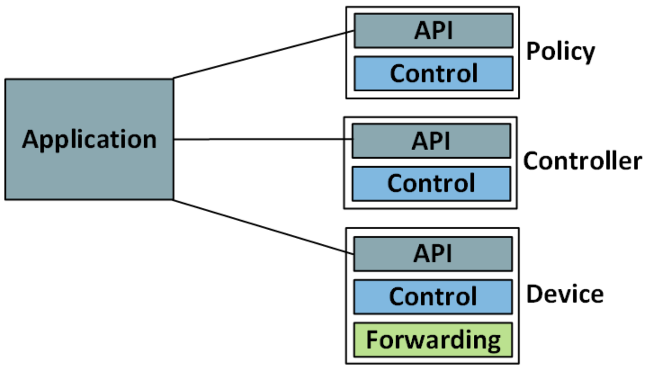


Figure 7-20 - Hybrid SDN approaches

One of the goals of SDN is to centralize the control of the network: the approach in Classical SDN is to simplify the switches, removing the Control Plane from the switches and using a centralized system to control them. Different approaches could be used to centralize the control (see Figure 7-20): for instance, network programmability and centralized control can be achieved by allowing network applications to utilize an API to connect to the policy engine, to the controller and/or directly to the SDN device and leave the distributed control plane on the device.

Hybrid SDN via APIs is popular for SDN application developers, since, at the same time it provides programmability and centralized control and it supports legacy devices and Classical SDN devices at the same time.

7.7.1 Programmable SDN via Device API

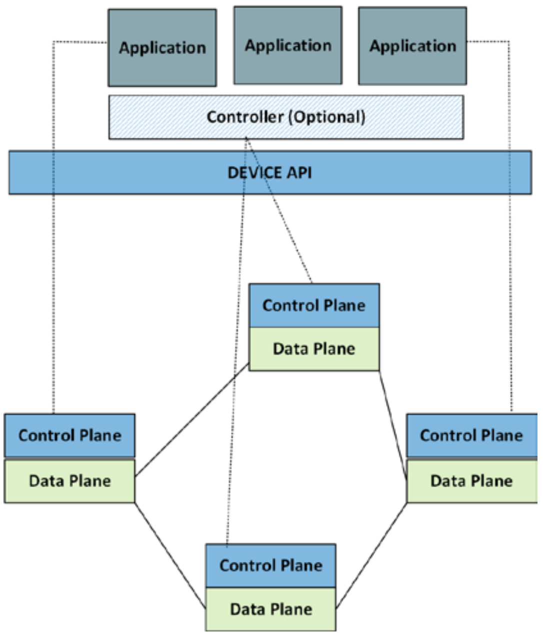


Figure 7-21 - Hybrid SDN via Device API

Figure 7-21 shows how in SDN via Device APIs, the devices expose an API by using which external applications or an optional central controller can control the network behavior. It is worth noting that in this solution the controller is optional. However, even when the controller is available it does not provide all the benefits of classical/SDN because:

- The control plane is still on the devices
- The applications can directly invoke the actual API

In this solution the controller, when available, proxies the requests coming from the application to the devices. However, as already mentioned, the application may communicate directly with the devices, for instance using protocols such as CLI and SNMP.

These systems, even though they have been available for a long time, are difficult to maintain and are mainly used by somewhat uncommon static management tasks, not the dynamic systems required by modern datacenters. Newer protocols and approaches exist: the most popular of these is the RESTful API. It is worth noting that in Figure 7-21 that communication to devices using these APIs can be done either through a controller or directly. In either case, the solution uses improved APIs providing the ability to program the forwarding plane through proprietary API methods exposed by the device and customized by the manufacturer.

7.7.2 Programmable SDN via Controller API

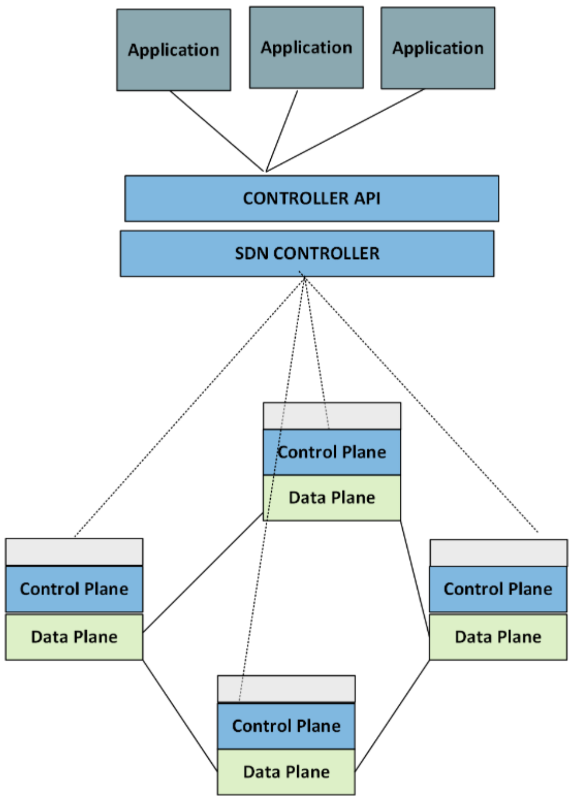


Figure 7-22 - Hybrid SDN via Controller API

SDN via Controller-level APIs shown in Figure 7-22 provides a platform on which SDN applications can be built. This type of SDN uses APIs provided by the controller, however, due to the lack of a standard NBI, applications written for one controller's set of APIs may not run on different controllers. The fundamental distinction between this solution and Classical SDN is the southbound APIs the controller uses:

- in Classical SDN, the southbound protocol is OpenFlow.
- In Hybrid SDN via Controller APIs, the southbound protocol is one (or more) legacy protocols re-purposed to provide SDN-like features through the controller.

One of the goals to implement APIs on the controller is to provide a level of abstraction between the devices and the application, hiding the details of each device and each protocol, and making it possible for the application to interact with the controller at a higher level: a use case is the presentation via controller API of the network topology information collected and stored in the controller.

7.7.2.1 Controller API in the WAN: SD-WAN use-case

Many SD-WAN implementations in the market use a controller API. In these solutions, existing routing protocols such as Border Gateway Protocol (BGP), Border Gateway Protocol Link State (BGP-LS) are used towards the network core to announce prefixes connecting remote sites.

Implementations of SD-WAN using these protocols leverage the technology existing in devices to create SDN applications that can control routes and paths throughout the network. In other solutions, the SD-WAN controller only processes the messages between the application and device: an example of this is the classic use of NETCONF as an SDN API. NETCONF is one of the most popular device management and configuration protocols used especially in higher-level devices such as core routers. These devices expose their configurable information via data models defined using the YANG data definition language.

7.7.2.2 Controller API in the Datacenter: OpenDayLight

A SDN controller that fits squarely in the SDN via Controller APIs category is ODL. One of the most important feature of ODL is its support multiple southbound protocols: amongst the others, it supports OpenFlow (giving support for Classical SDN), NETCONF (supporting SDN via APIs), and BGP-LS/ PCE-P (supporting SDN via APIs).

7.7.3 Hybrid SDN via Policy API

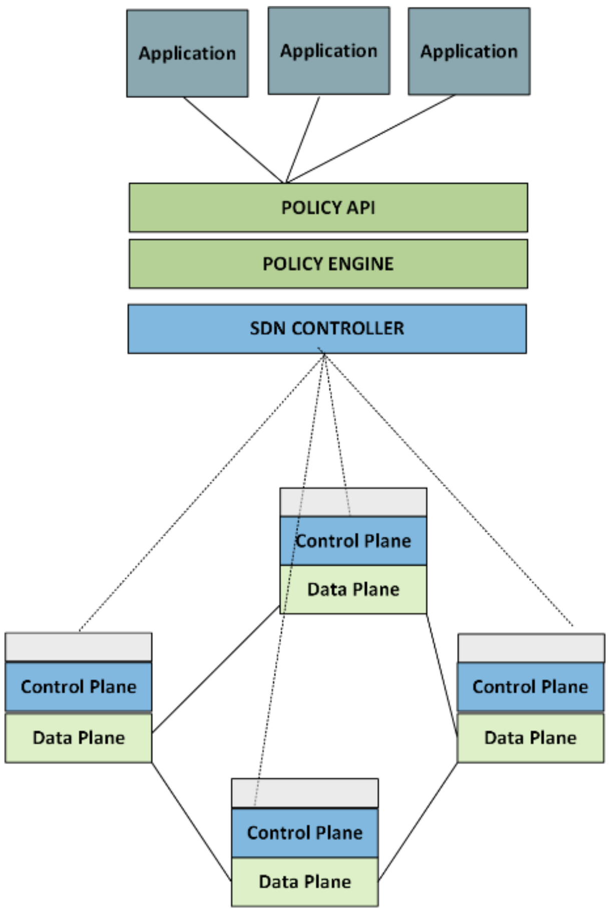


Figure 7-23 - Hybrid SDN via Policy API

Figure 7-23 shows another approach to SDN: a hybrid SDN using APIs located above the controller. These APIs expose policies, rather than single devices or network capabilities. The policies are provided in different flavors and address different domains, but they all manage the network configuration using a *declarative* approach, rather than an *imperative* one, where the main difference between the two is:

- Imperative approach: with this strategy for systems and APIs, the user must input exactly how the task should be completed.
- Declarative perspective: with this strategy for systems and APIs, the user must input exactly what is to be accomplished. The system will resolve how to do it.

In the declarative control model, instead of manually hard-coding a set of basic instructions, the application requirements instruct, at a high-level, the switches and they implement how, where, when and what they can. For example, the control tower commands an airplane where to take off. The pilot of the plane, not the control tower, manages the take-off process. This is the principle of declarative control model from the promise theory. The declarative approach introduces a new level of abstraction between the hardware and the software and a methodology to adapt networking across various hardware platforms, capabilities, and future evolutions.

Baggage handlers follow sequences of simple, basic instructions
Air traffic control tells where to take off from, but not how to fly the plane

Figure 7-24 - Declarative vs Imperative approach [85]

These policy-based API solutions are gaining space in both SDN via APIs and Classical SDN categories. An example of this type of API is the NBI idea of intents, which enhances the API level of abstraction enabling truly declarative requests. The notion of controlling network behavior via policy is an active area of research in SDN, and in the next paragraph a Cisco APIC-DC is presented as an example of this approach[12].

7.7.4 Cisco APIC-DC: an example of SDN via Policy API

Cisco ACI fabric [85] [86] is a set of distinct components like switches or routers but provisioned and monitored as a single entity. The ACI controller, called the Application Policy Infrastructure Controller (APIC), is the central point of management of the fabric, distributing policies to all the switches that are part of the fabric.

The Cisco ACI Fabric OS runs on the building blocks of the architecture, which are the Cisco Nexus 9000 Series nodes. The Cisco ACI Fabric OS is object-oriented and enables programming of objects for each configurable element of the system. The ACI Fabric OS renders abstract policies (intents) from the controller into a concrete model that runs in the physical infrastructure. The concrete model is analogous to compiled software; it is the form of the model that the switch operating system can execute.

Cisco ACI fabric is managed, monitored, and administered by the Cisco APIC Controller, which can be configured by a REST NBI managed directly by the API operators through the built-in GUI, or via REST calls or Python scripts (see Figure 7-25)

[12] It is worth mentioning though that in Cisco APIC the concrete policies are mostly implemented in proprietary hardware, typically on Nexus 9k switches

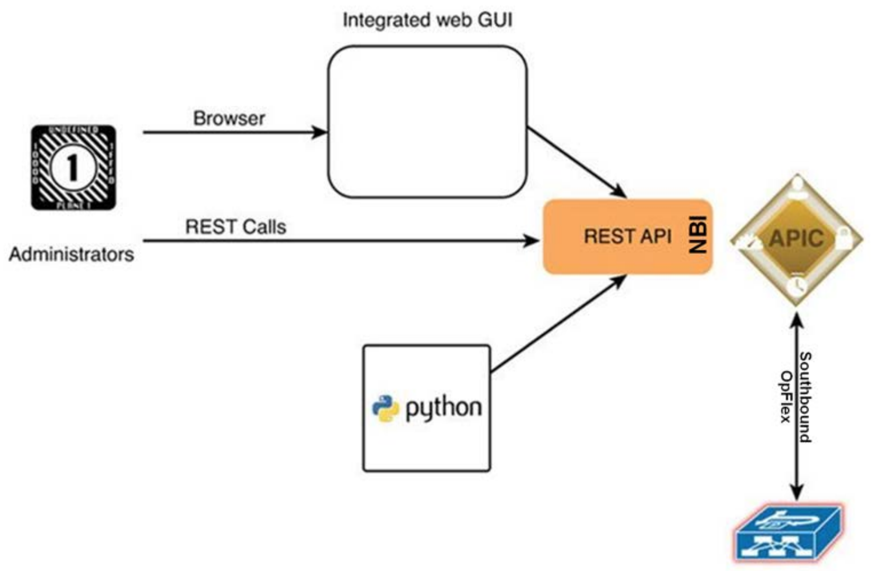


Figure 7-25 - APIC Northbound and Southbound interfaces

7.7.4.1 Cisco APIC model and promise theory

The Cisco APIC policy model is an object-oriented model built on the promise theory (see Figure 7-26). Promise theory is based on the scalable, declarative management of intelligent independent objects, handling configuration changes requested by the control system. These objects also trigger exceptions or faults back to the control system. This reduces the complexity of the control system and improves scalability. This approach can scale further, replicating the delegation in a hierarchical, top-down architecture, allowing the underlying objects to in turn request state changes from one another and/ or lower-level.

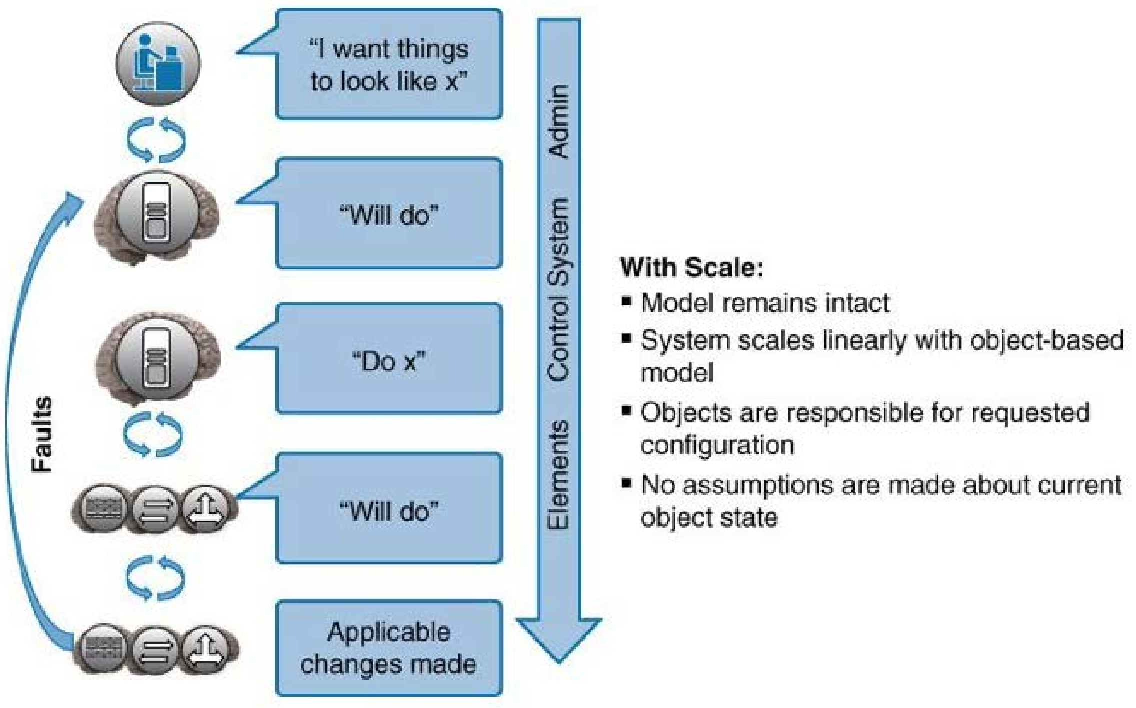

"I want things to look like x"
"Will do"
"Do x"
"Will do"
Applicable changes made
Admin
Control System
Elements
Faults
With Scale:
▪ Model remains intact
▪ System scales linearly with object-based model
▪ Objects are responsible for requested configuration
▪ No assumptions are made about current object state

Figure 7-26 – Promise Theory Approach [85]

7.7.4.2 Physical Topology Model

The physical Cisco ACI fabric is built on a spine-leaf design; its topology is illustrated in Figure 7-27. For further details on the spine-leaf architecture see Par. 5.3

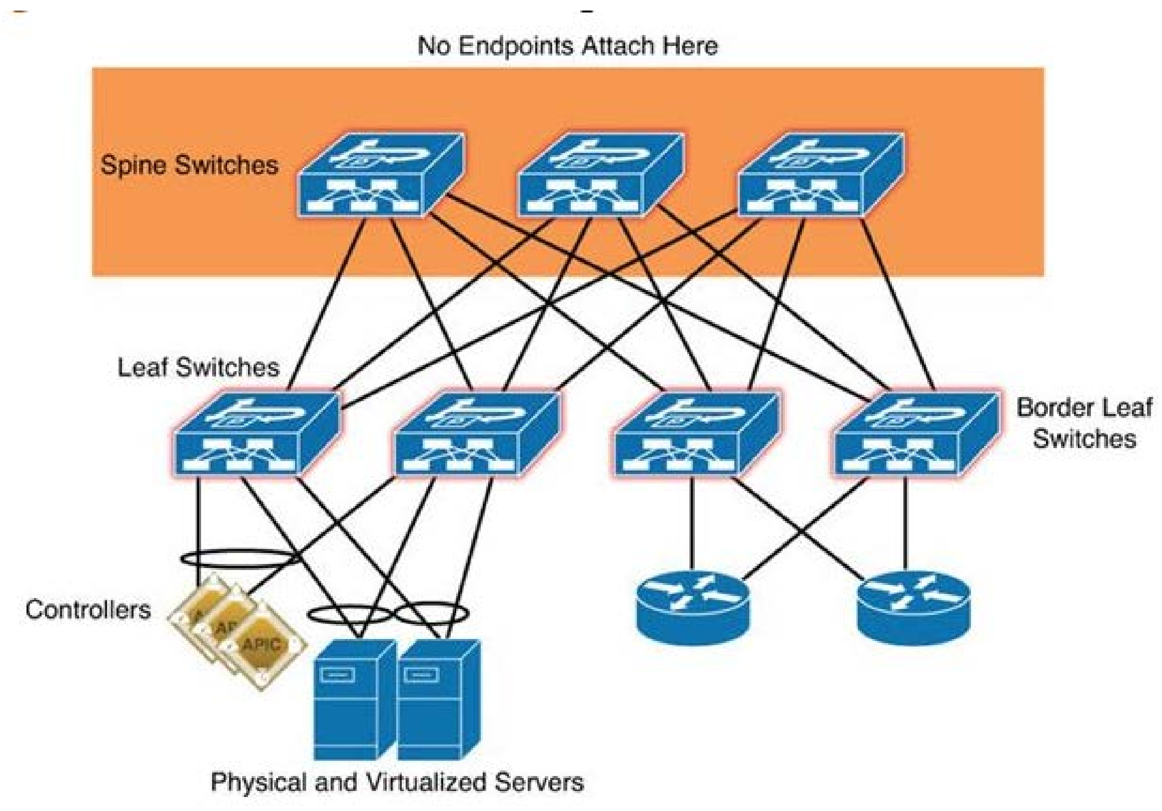


Figure 7-27 - Cisco ACI Fabric [86]

8 Network Functions Virtualization

8.1 Abstract

Network Functions Virtualization (NFV) is a technique that virtualizes entire classes of network functions to build blocks that may be chained to create network services. For a thorough view on NFV and SFC see [47] ; [87] provides a historical perspective and [88] describes some datacenter use cases of NFV and SFC.

This chapter presents the basic concepts behind NFV and shows how they could be used in an Software-Defined Enterprise Datacenter Network.

8.1.1 The transition to Network Function Virtualization

In modern datacenters, virtualization of servers and storage is an already proven and well-established technology: many independent server and storage systems have mostly been replaced by their counterpart on shared hardware.

In the long journey to infrastructure as code, the next step of the infrastructure virtualization is the virtualization of the network functions. This process widens the scope of virtualization to the network devices and their functions, triggers the Network Functions Virtualization (NFV) and enable the (Virtual) Network Functions Chaining (NFC)

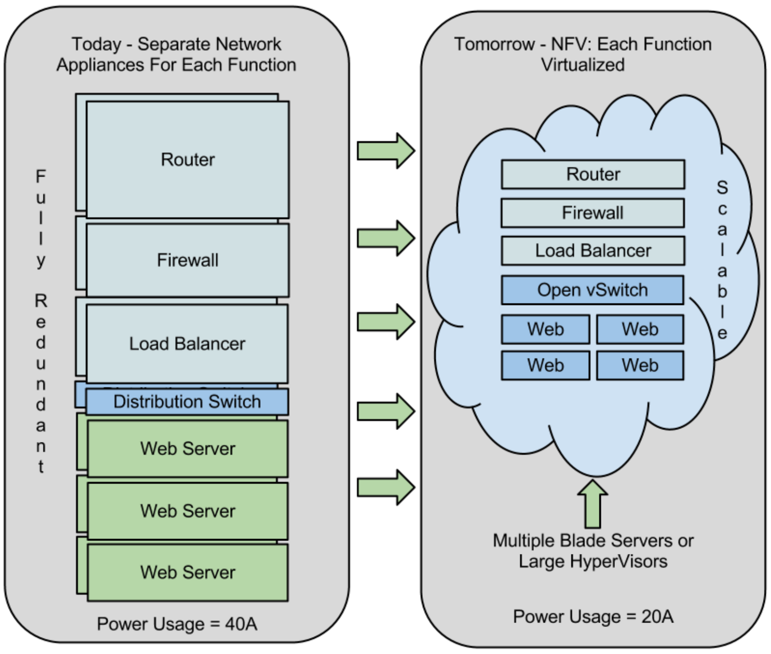


Figure 8-1 - Switch to Network Function Virtualization [47]

The acronym NFV references all the services: the virtual network functions, their management platform, and the infrastructure integrating these components on a hardware platform. NFV is more accurately defined as "*the method and technology that enable the* *replacement of physical network devices performing specific network functions with one or more software programs executing the same network functions while running on generic computer hardware"* [47]. In NFV terminology this (virtual) implementation of the network functions is referred to as virtualized network function (VNF). A VNF provides a specific network function (e.g. firewall, load-balancer, IPS) as a software component (container, VM, etc.). Usually, chaining of these VNFs may be required to implement the complete network pipeline being virtualized: this is chain is referred as Service Functions Chaining (SFC) or Network Functions Chaining (NFC).

One example of SFC is the replacement of a physical load balancer appliance with a virtual machine (aka virtual appliance) providing the load balancing function. The VM runs the same operating system and has the same look and feel of the physical appliance—but runs on top of a non-dedicated, shared, and generic virtualization platform. With NFV, the network functions can be implemented on any hardware able to offer the required hardware resources and virtualize them. Virtualization has reached the point that the physical device can be masked up to the point that COTS hardware can now provide the infrastructure for NFV, or in other terms, with virtualization, general purpose hardware can be used to provide the resources needed by a VNF.

8.1.2 The need of a modern Architectural framework

Traditional network hardware and software used to be customized and tightly integrated. NFV adopts a different approach, allowing software developed by the vendors to run on general purpose shared hardware, creating several points for management. The NFV architectural framework is developed to ensure that these touch points are standardized and compatible across multiple vendors.

NFV was first introduced at the SDN OpenFlow World Congress in 2012 [89] by a consortium of service providers, whose goal was to address the major challenges faced by network operators. One of the main issue was the need to upgrade the hardware when the operators wanted to provide innovative services to their customers. The group proposed NFV to tackle these challenges and improve efficiency by "*leveraging standard IT virtualization technology to consolidate many network equipment types onto industry standard high-volume servers, switches and storage, which could be in Datacenters, Network Nodes and in the end user premises*" [90].

To accomplish this objective and identify the specifications needed to move from the legacy network and vendor centric approach to NFV, seven of the principal TelCo operators formed an Internet specification group (ETSI ISG)—under the ETSI independent standardization body [91].

This group officially started in early 2013, defining requirements and an architectural framework that can support the virtualized implementation of network functions performed by COTS hardware devices from vendors.

8.2 The NFV Framework

8.2.1 The three criteria of ETSI NFV

The ETSI group used three key criteria for producing the recommendations:

1. Decoupling: complete separation of hardware and software.
2. Flexibility: automated and scalable deployment of the network functions.
3. Dynamic operations: control of the operational parameters of the network functions through granular control and monitoring of the state of network.

Based on these criteria, a high-level architectural framework was established.

8.2.2 The High Level ETSI NFV Framework

The ETSI working group established a high-level framework, defining the distinct zones of interest shown in Figure 8-2

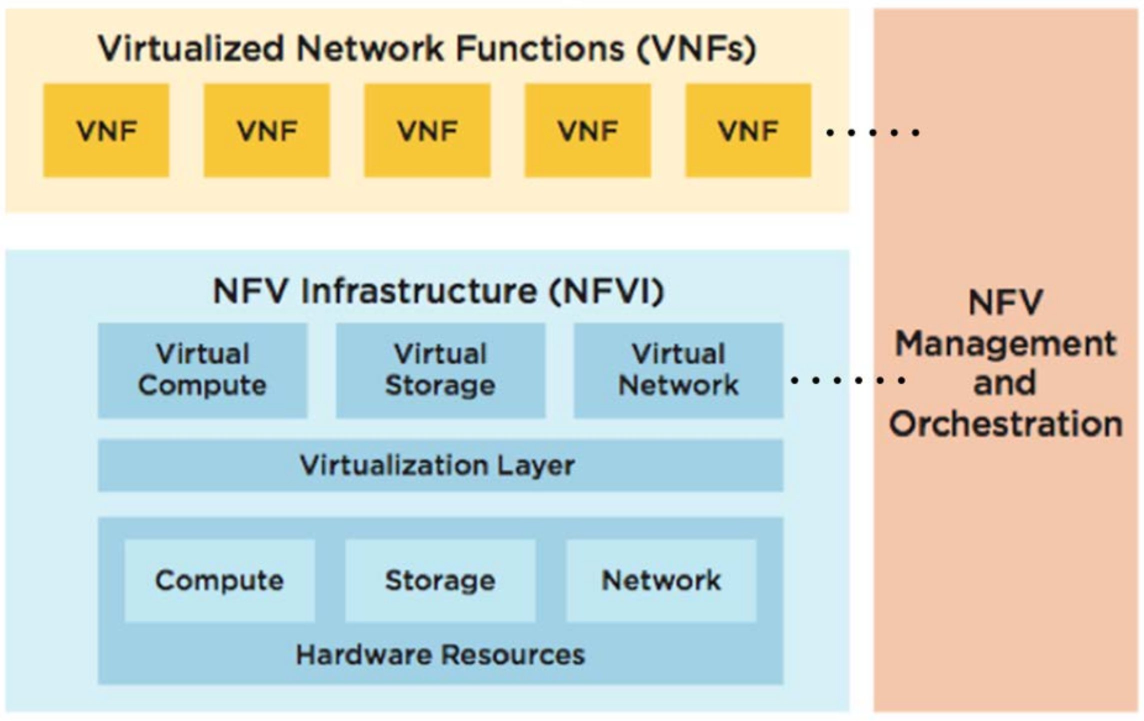


Figure 8-2 - High Level NFV Framework [87]

This framework, generally referred to as the *ETSI NFV framework*, forms the foundations of the standardization and development covering VNFs management, VNFs relationships and interdependencies, data flows and resource allocation. ETSI ISG categorized these roles into three high-level blocks, namely:

1. The infrastructure block
2. The Virtualized functions block
3. The management block

In ETSI's definition, the formal names of these blocks are named as:

- Network Functions Virtualization Infrastructure (NFVI) block: This block is the foundation of the overall architecture, teaming up the hardware to host the virtual machines, the software to make virtualization possible, and the virtualized resources.
- Virtualized Network Function (VNF) block: this block uses the virtual machines offered by NFVI and builds the functions on top of them by adding the software implementing the virtualized network functions.
- Management and Orchestration (MANO) block: this block co-operates with both the NFVI and VNF blocks, managing all the resources in the infrastructure layer, the resource creation and deletion, and their allocation to the VNFs.

8.2.3 NFVI and resource sharing

The two fundamental goals of NFV are the separation of software and hardware of a network device and the capability to share hardware resources pools among different VNFs. These two goals are achieved through VNFs running on a NFVI deployed either as standalone entities or chained together on a chain of network services, each provided by a VNF. The protocols associated with the function being virtualized within a VNF do not need to be aware of the virtualized implementation, exactly what happens when applications servers are virtualized, without any need to change the application layer.

Since these VNFs by design do not need to run on dedicated or customized hardware, COTS hardware can be used to run them through virtualization platforms allowing also to share the same hardware among multiple VNFs. These virtualization platforms, hypervisor-based or container-based, are already mature, having been in use in the datacenters for years. The NFVI leverages COTS hardware as a shared set of resources creating virtual resource pools that can be allocated on demand to the VNFs, as shown in Figure 8-5

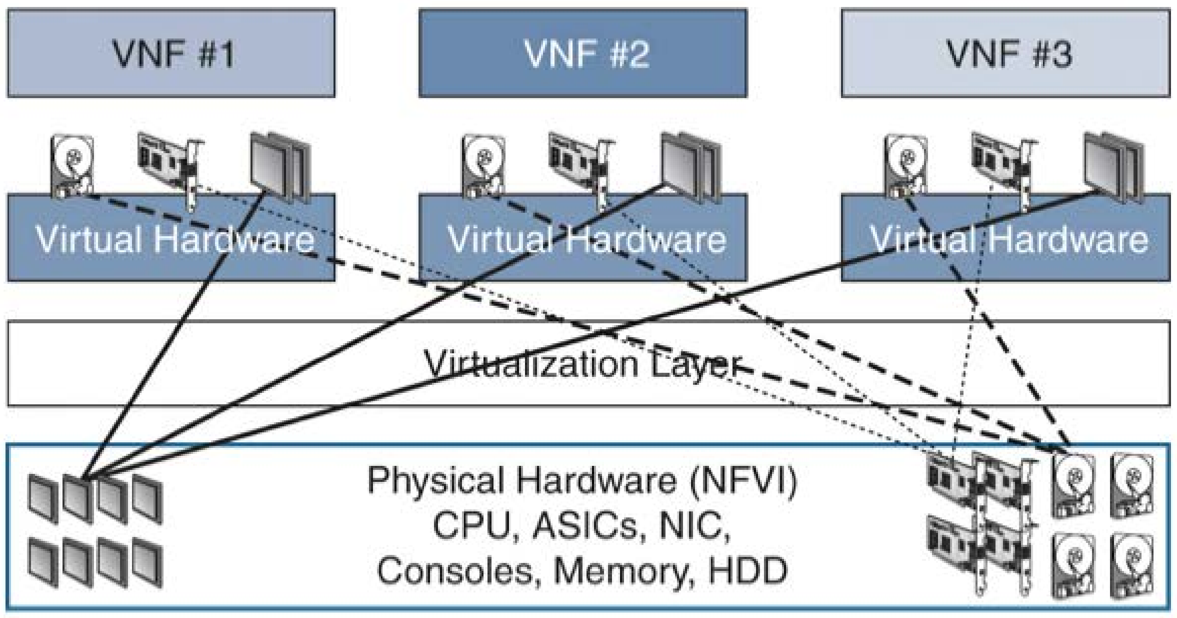


Figure 8-3 - Hardware resources shared between different VNFs [47]

8.2.4 Main advantages of NFV

"NFV proposes a framework to transform the approach to network design, deployment and operation, at the same time while offering many layers of improvement and efficiency across these" [87]

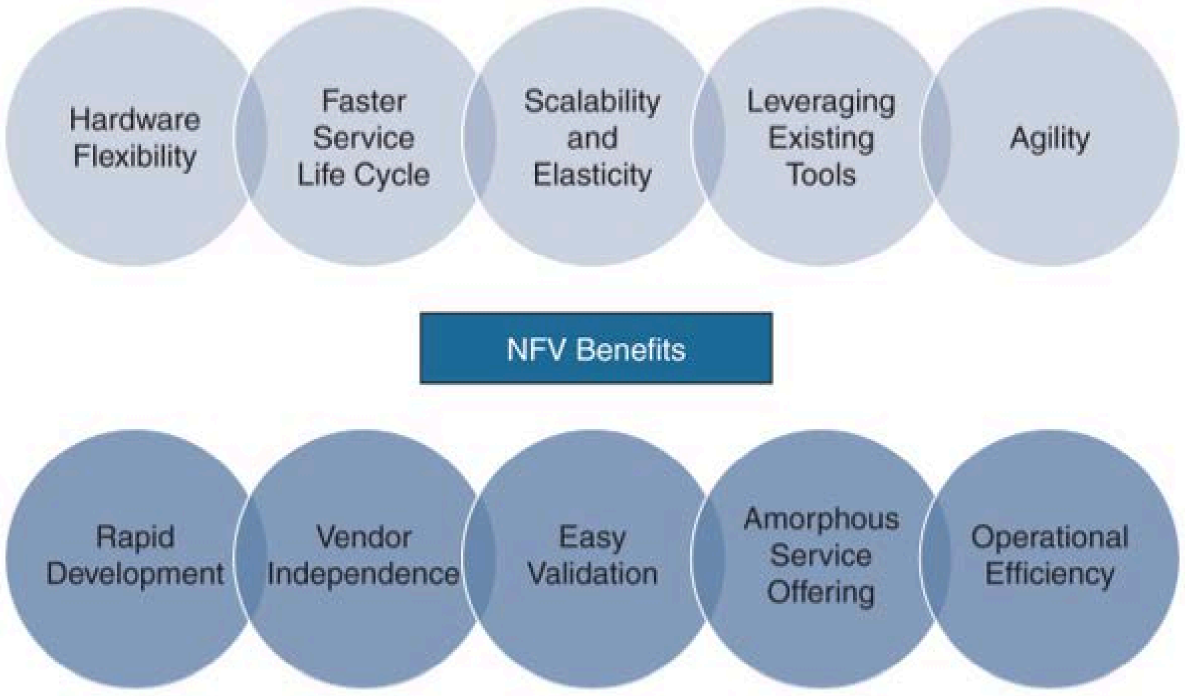


Figure 8-4 - Main advantages of NFV [47]

8.3 Network Function Chaining

To provide an end-to-end network service, the VNFs must be assembled and their order and interfaces must be properly defined: this is the process called “*Service Function Chaining*” (SFC) or *Network Function Chaining (NFC)*. For a description of the SFC architecture as defined by IETF, see [92] and [88]

NFC is the process of assembling VNFs into a desired network service. This process involves 3 steps:

1. Define which VNFs (e.g. firewalls, Load Balancers) are needed in the solution
2. Define the order on which the VNFs should be applied. This order is called the service chain and specifies the path through which the packets with a certain tag flow. The VNFs themselves are created through virtualization, with a virtualization software layer on top of underlying physical hardware, often through use of white boxes.
3. Apply the NFC to the packet process/transport infrastructure (i.e. through a network Service design defined in a SDN Controller SFC capable)

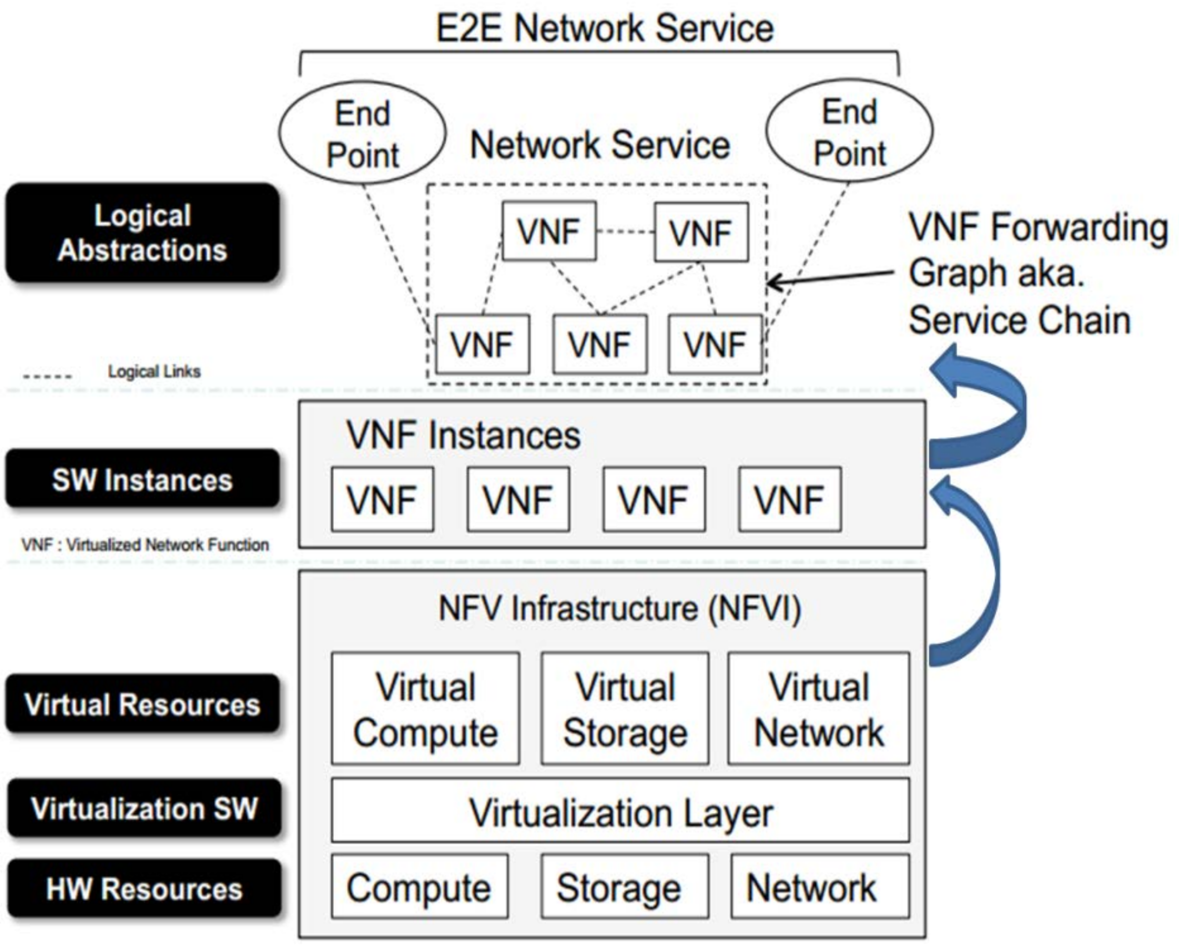


Figure 8-5 – NFV, VNFs and NFC

8.3.1 SFC Architecture

8.3.1.1 SFC Architecture principles

SFC is based on five key architectural principles:

1. Topological independence: the underlay does not need to change to deploy SFC
2. Plane separation: the dynamic realization of Function Paths is separated from packet handling operations (e.g., packet forwarding).
3. Classification: Traffic that satisfies classification rules is forwarded according to a specific Service Function Path (SFP). Classification can occur at varying degrees of granularity.
4. Shared Metadata: Metadata/context data can be shared amongst Service Functions and classifiers, between Service Functions, and between external systems and Service Functions (e.g., orchestration). One use of metadata is to provide and share the result of classification (that occurs within the SFC-enabled domain, or external to it) along a Service Function Path.
5. Service definition independence: The SFC architecture does not depend on the details of Service Functions themselves.

8.3.1.2 SFC Service Function Chain Classification and Encapsulation

IETF WG identifies a Network Service Header that could be "*inserted onto packets or frames to realize service function paths, enforced at the packet level*" [56]. In this way the service insertion does not need to be in-path (on layer 3) or provided as L2 transparent service, but could be completed de-coupled from the data path

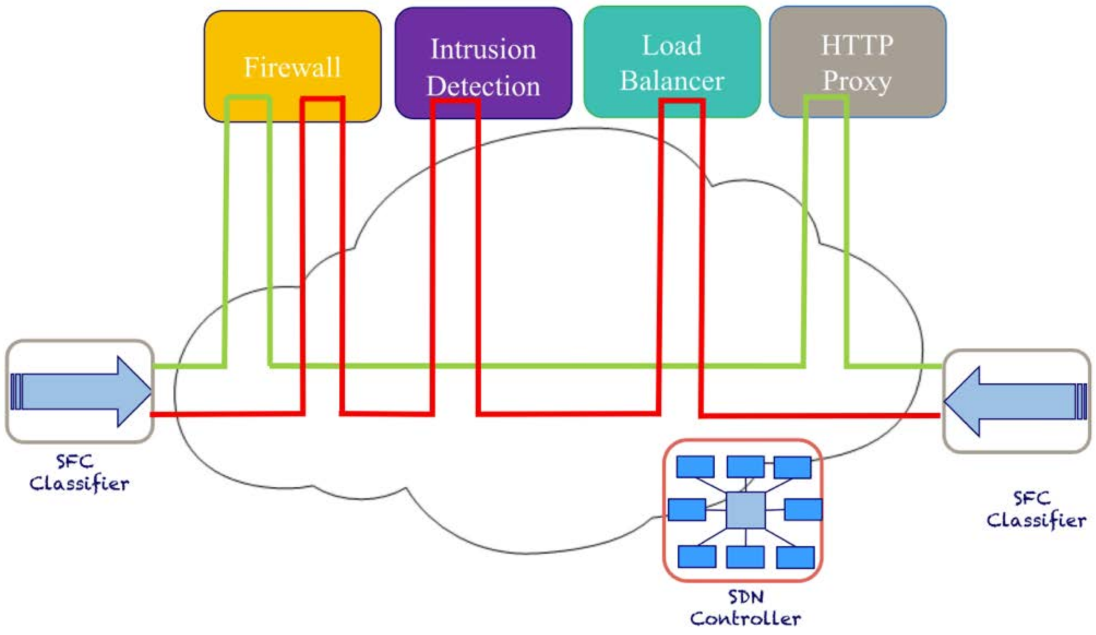


Figure 8-6 - SFC Classifier

NSH relies on the job completed by a network classifier (see Figure 8-6). When the classifier identifies the traffic to be forwarded to the service chain path, additional header information in the data frame is added to it. This additional header is called the service function chain encapsulation. There are multiple possible encapsulation headers, and existing overlay techniques, such as Layer 3 Virtual Private Network (VPN) or segment routing (SR), can be used for this purpose. These overlay methods depend on the presence of an IP network. IETF is driving standardization for a new SFC encapsulation format under the banner of network service header (NSH), which can work with various other underlying networks.

8.3.1.3 Network Service Header (NSH)

The Network Service Header (NSH) offers a standard for SFC encapsulation that is regulated by IETF and supported by multiple vendors in the networking industry. NSH is composed of two major components:

1. the first provides information about the service path the traffic flow takes in the network,
2. the second carries additional information about the payload in the form of metadata

Applications and higher-level protocols can use the metadata component of NSH to send their information along the service path. This information can be helpful in the design making process for the service path selection and any other special handing the packet may need. NSH protocol's header is defined as a set of three types of headers—base header, service path header, and context header—as shown in Figure 8-7

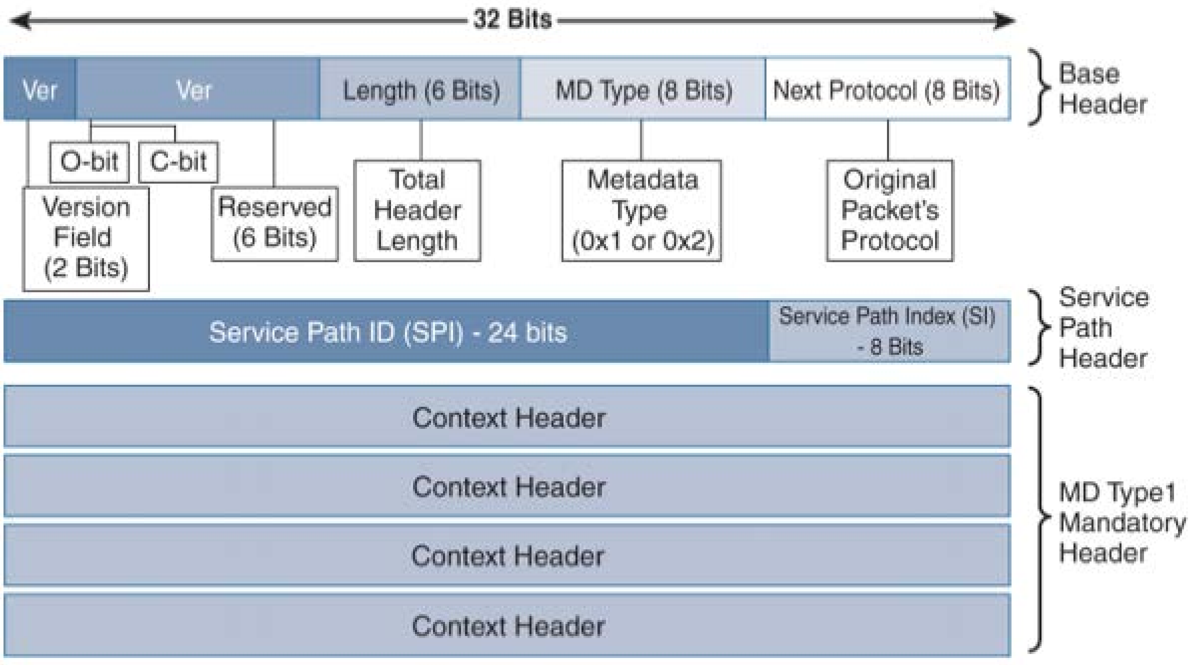


Figure 8-7 - NSH Protocol Header

8.3.1.4 Metadata

A major advantage of SFC is the capability to carry and consume application-level information provided as metadata containing the contextual information about the data that is transported through the SFC domain.

The SFC classifier insert metadata in the service header, such as the context header of NSH. The SFC may extract this information from higher layer protocols, such as the HTTP method contained in an HTTP transaction or a user-agent contained in the HTTP header. Once the metadata is added to the SFC protocol's header, the nodes (containing Service Functions) in the path can read, process, and react to the data and take the corresponding predefined action. The exchange of metadata across the SFC elements can be accomplished by different methods.

8.3.2 SFC Use Cases in the DCN

This paragraph presents 3 use cases for SFC as defined in [93] and in [88]

8.3.2.1 North-South Traffic

North-South traffic originates from outside the datacenter and is typically associated with flows originated by the Enterprise Users. The traffic may also be associated with Guest users accessing news, email, social media and other websites. Usually, this traffic is destined to services hosted in the datacenters. BYOD and social networking applications, amongst the others, require this traffic to be analyzed, application and users to be identified, transactions to be authorized, and at the same time security threats must be mitigated or eliminated. To this end, various service functions, as illustrated in Figure 8-8, are deployed in different VNFs at various topological locations in the network. The VNFs are selected based on the policy required for the specific use case

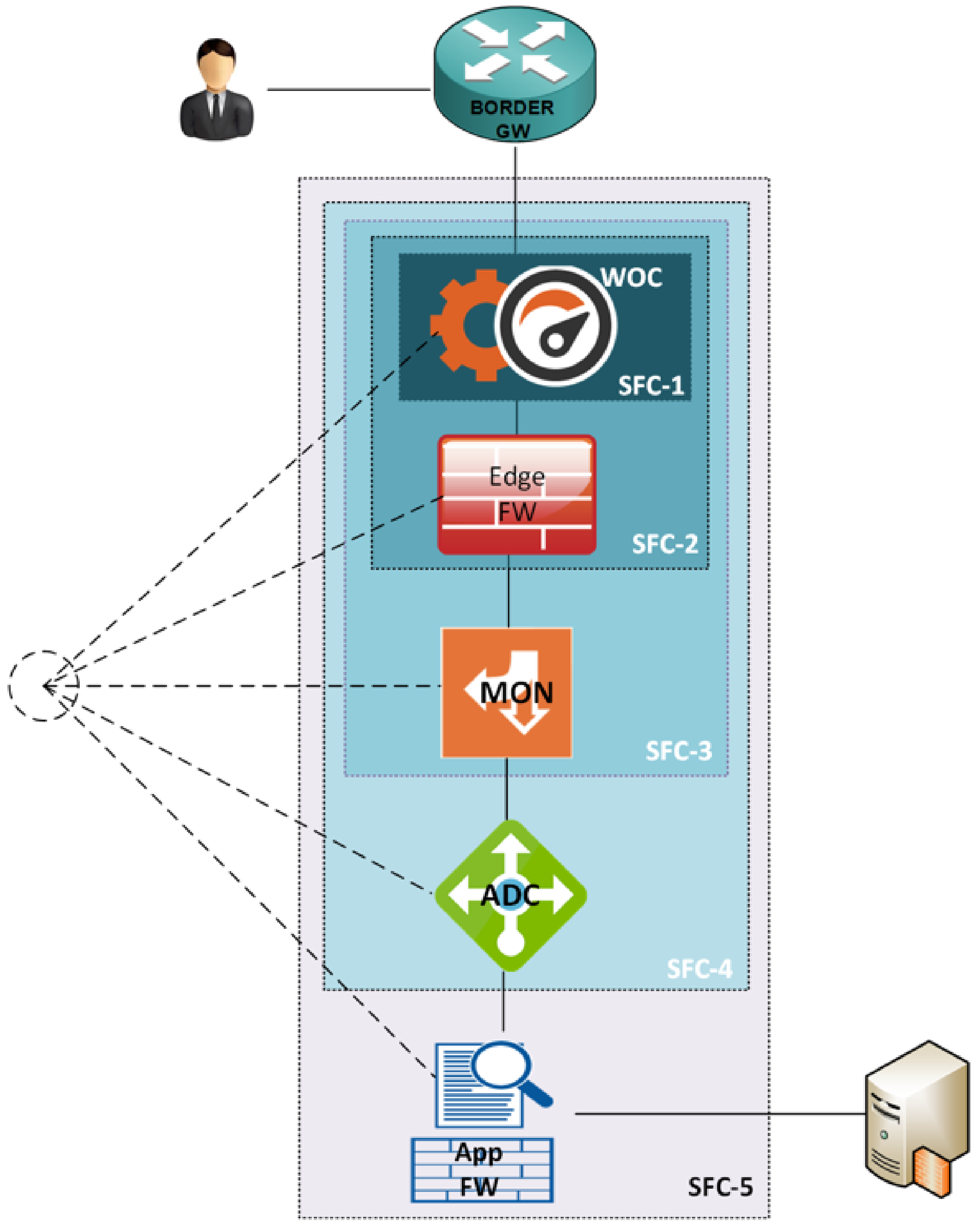


Figure 8-8 - SFCs for North-South Traffic

Figure 8-8 shows the ordered list of VNFs, from top to bottom, representing the data flow from End Point to Workload and vice versa.

Traffic does not always strictly flow through all the VNFs in that order. Traffic flows through various combinations the VNFs. The connections between VNFs (represented by the dashed lines on the left) map the network topology required to achieve the traffic flows. Each permutation represents a SFC. Certain ordering (chains)of the VNFs are intrinsic to the nature of the VNF applied. For instance, to be effective, the Web Optimization Controller (WOC) requires the flow to be decrypted first, so the VPN VNF must be applied prior to WOC. Vendor implementations of VNFs enable choices for various deployments and ordering.

8.3.2.2 East-west traffic

This is the predominant traffic in datacenters today. As mentioned in the previous chapters, this explosion in east-west traffic is leading to newer datacenter network fabric architectures. Unlike north-south traffic, where security threats may come from outside the datacenter, any threat to this traffic comes from within the datacenter.

SFC applied on the east-west traffic id captured in a generalized fashion in Figure 8-9. ADCs[13], although shown as isolated VNF in each of the tiers, are often consolidated into a smaller number of ADC VNFs and shared between multiple tiers. Virtual IP addresses in these VNFs represent the single ADC instances. Flows are terminated at the VIPs and reinitiated towards the load balanced workloads.

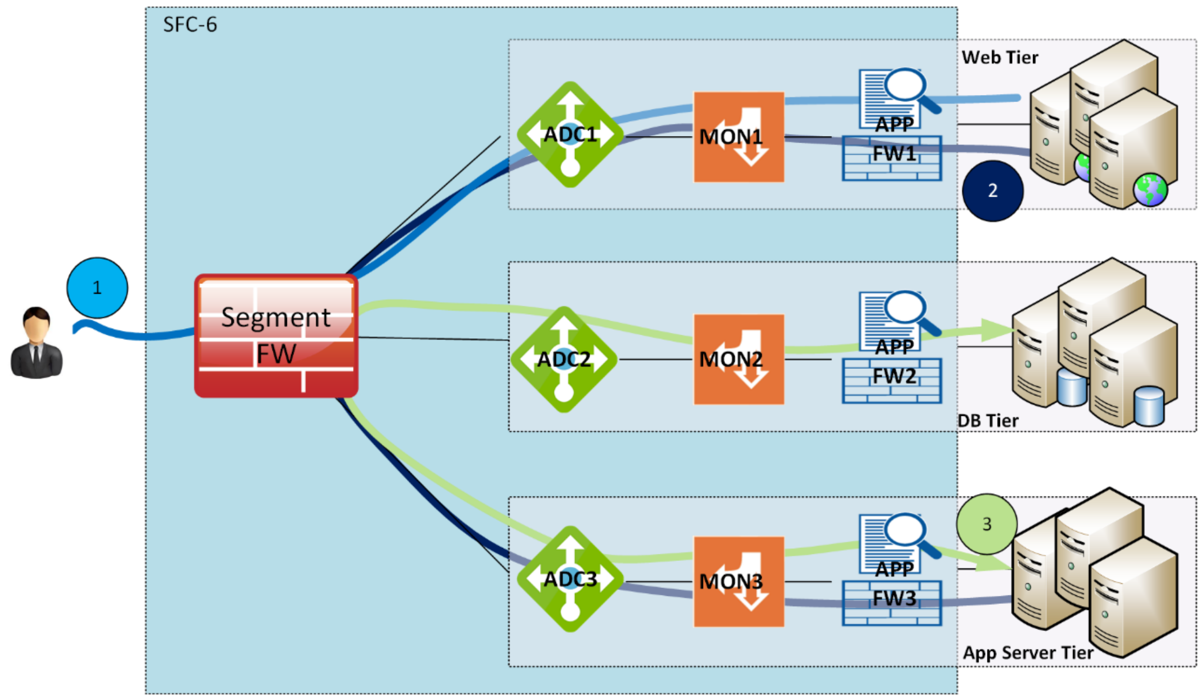


Figure 8-9 - SFC for East-West

As an example

1. HTTP GET request arriving at Application Deliver Controller (ADC1) is load balanced to a webserver pool represented as Workload1 (Light Blue Line).
2. To respond to the GET request, Workload1 generates traffic to an application server in a pool represented as Workload2 through ADC2, which load balances the webserver-initiated traffic (Dark Blue Line)

[13] An **application delivery controller** (ADC) is a network device that helps perform common tasks, such as those done by web sites to remove load from the web servers themselves. Many also provide load balancing.

3. Likewise, the application server, as part of processing the webserver’s request generates traffic to a DB server pool represented as Workload3 through ADC3, which load balances the application server-initiated traffic (Green Line).

The traffic arriving at different ADC might arrive to different VIPs, each corresponding to its tier but belonging to the same ADC. In every tier, the traffic flowing between the ADC and the designated server is monitored by one or more application firewalls specializing in different types of threats (i.e. Web Application Firewalls, Network Intrusion Prevention systems, etc.). Again, steering can enable the sharing of these VNFs (as done for the ADC).

8.3.2.3 Multi-tenancy

Multi-tenancy is relevant in both enterprise and service provider environments. Enterprises might see internal organizations, partners or business units as tenants, and therefore the service models they apply must be tenant aware. Multi-tenant service delivery can be accomplished in two primary ways:

- VNFs themselves are tenant aware: every VNF is built to support multiple tenants.
- VNF instances are dedicated for each tenant

In both cases, the Service Provider manages the VNFs. To support multi-tenant VNFs, traffic being serviced by a SFC must include a tenant identifier [94] carried along with the traffic to be serviced. It is typical of tenant assets to be deployed in an isolated layer2 or layer3 domain such as VLAN, VXLAN or VRF.

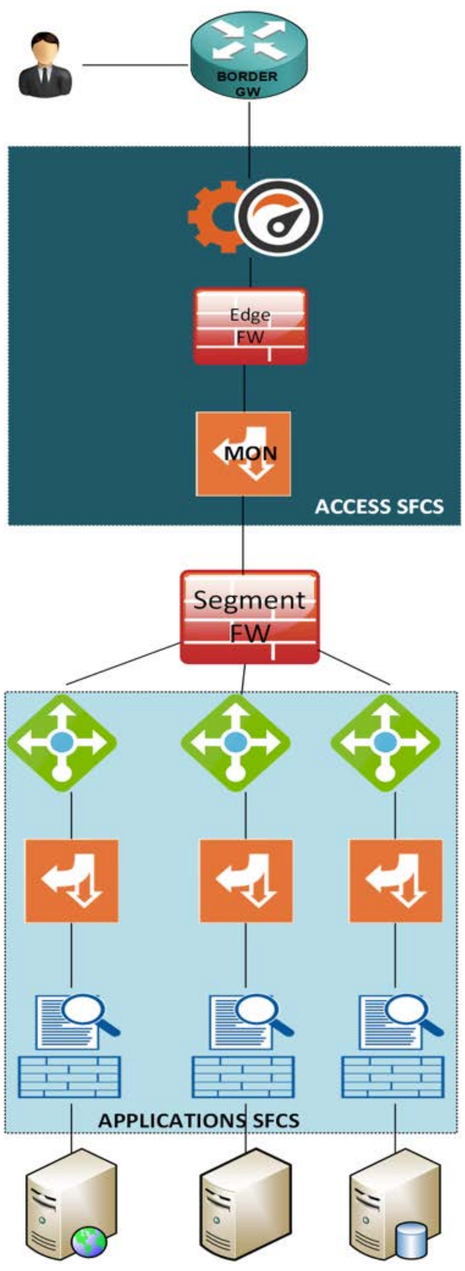


Figure 8-10 – SFC for Multitenancy

Although this model is feasible, it lacks flexibility needed by the service providers. Access SFCs focus on servicing inbound and outbound datacenter traffic, while Application SFCs focus on handling application traffic.

On each tenant, service providers deploy one "Access SFC" and several "Application SFCs". On the other hand, depending on the enterprise, datacenter operators may not need Access SFCs. If these Access SFCs are needed, the operator may also deploy a minimal Access SFC including WOC and VPN functions to support the branch and mobile user traffic, while at the same time utilizing the security policies in the application SFCs. The latter is the case in zero-trust network [1] architectures where security policies are applied near the resources and applications as opposed to the perimeter.

9 Datacenter Network as a Service with SDN, NFV and NFC

9.1 Abstract

This chapter shows how SDN, NFV and NFC work together to further expand the concept of Software-Defined Networking and achieve a dynamic, adaptive DCN able to provide Network as a Service through an orchestration platform. For a deeper discussion on the dependencies and interaction between SDN, NFV and NFC see [47] , [95] and [96]

9.2 SDN and NFV

Even though SDN and NFV are two independent approaches, born for different reasons, they share many objectives and can mutually advantage and support each other 'adoption. As already discussed in the previous chapters, when traditional network devices are considered, the control, data, and hardware planes are tightly integrated together, as shown in Figure 9-1. This makes hard to scale them independently. This architecture doesn't offer flexibility to implement new services or the agility to absorb changes. Both SDN and NFV play a role in breaking this bonding in two different dimensions (see Figure 9-1 below):

1. SDN separates the control plane from the data plane, using a centralized control plane to manage, manipulate, and monitor the data plane.
2. NFV decouples the network function from the vendor-built hardware, facilitating the use of generic hardware to run the software implementing the network functions.

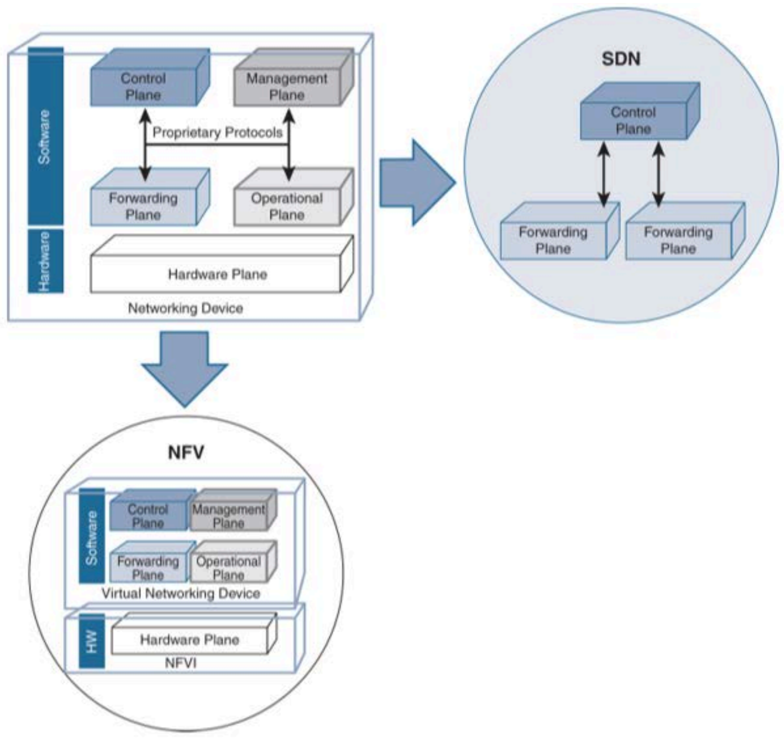


Figure 9-1 - SDN and NFV [47]

Even though NFV and SDN address differently the problem of a flexible, scalable, elastic, and agile network deployment, the principles of SDN can be applied to NFV by separating the control plane from the forwarding plane and virtualizing the network functions.

Figure 9-2 reflects this blended relationship, and in this scenario NFV uses COTS hardware and implements the forwarding plane as a VNF, while the control plane function is delegated to the SDN controller

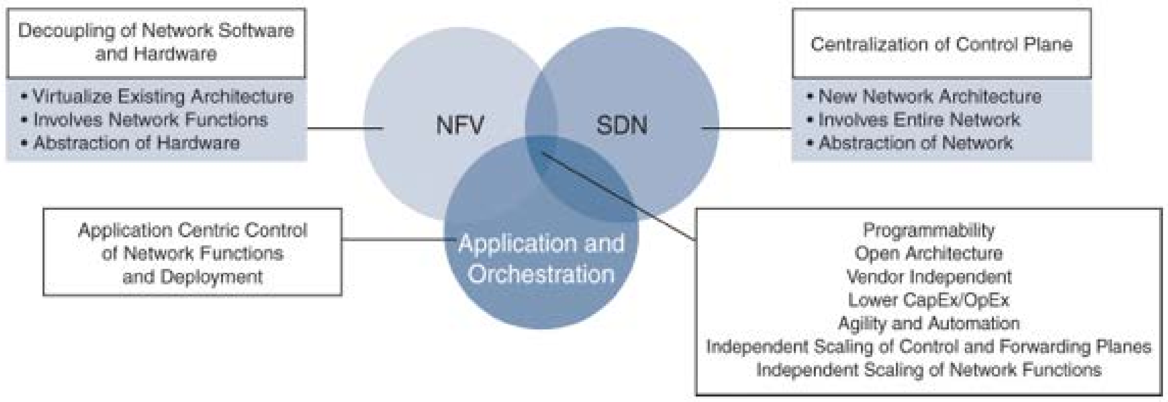

Decoupling of Network Software and Hardware
• Virtualize Existing Architecture
• Involves Network Functions
• Abstraction of Hardware
NFV
SDN
Centralization of Control Plane
• New Network Architecture
• Involves Entire Network
• Abstraction of Network
Application Centric Control of Network Functions and Deployment
Application and Orchestration
Programmability
Open Architecture
Vendor Independent
Lower CapEx/OpEx
Agility and Automation
Independent Scaling of Control and Forwarding Planes
Independent Scaling of Network Functions

Figure 9-2 - Combination of SDN and NFV [47]

Applications may stitch this relationship together maximizing the advantages of both technologies and offering a new approach to networking. As summarized in Figure 9-2, the combination of these three areas meet cloud scale requirements for on-demand expansion, optimization, deployment, and speed.

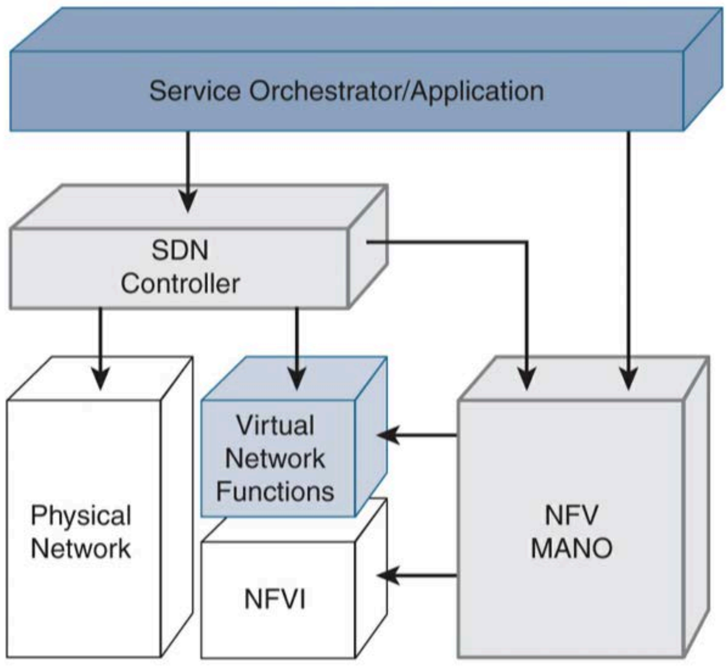


Figure 9-3- Network orchestration for SDN and NFV [47]

Figure 9-3 shows SDN working with both physical and virtual devices providing network functions, while NFV provides the VNFs and manages the physical infrastructure hosting them. The Orchestration layer, situated at the top performs end-to-end service orchestration and interacts with both the SDN controller and NFV.

9.3 Network as a Service

9.3.1 Virtual Infrastructure build

To maximize the benefits of NFV and SDN, the network should be provisioned, managed and maintained enabling, whenever possible, the use of network programmability. These technologies enable a programmable network which, ultimately, creates a dynamic infrastructure that can quickly adapt and implement business needs.

This paragraph presents how a network managed by applications and programs boosts its efficiency while contributing to achieve the goals of SDN and NFV introduced in the previous chapters. For the sake of simplicity, we assume that NFV infrastructure elements (computing, storage, and networking), along with underlay network that provides connectivity, are already deployed and available. Figure 9-4 shows the flow of events in an environment where NFV, SDN, and application coexist. It is worth noting that in a multi-tenant environment the impact of the transformation (both on the physical and virtual networks) triggered by a user will be limited to the tenant to which the user has access.

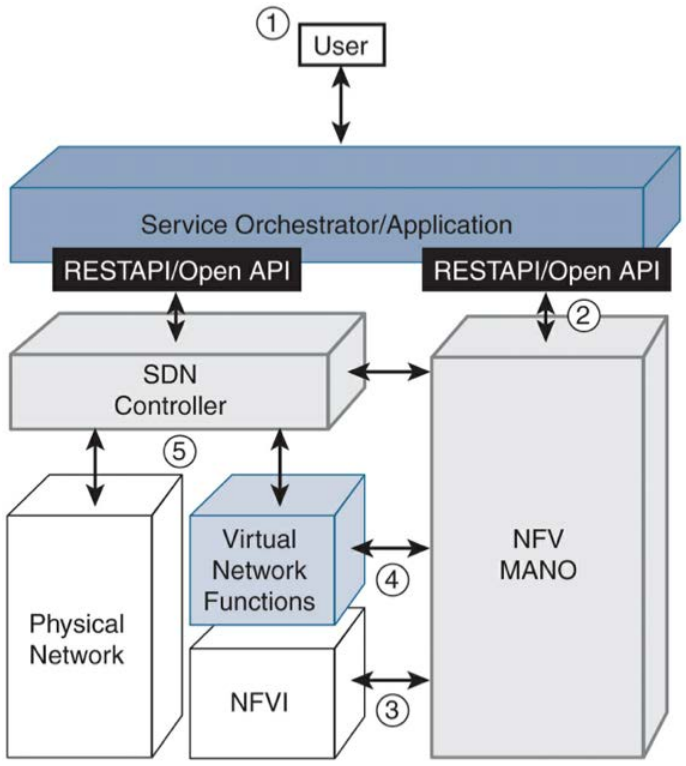


Figure 9-4 - Programmability Flow in a NFV/SDN enabled network [47]

The steps are as follows:

1. The network design and implementation flows are initiated from the application layer (i.e. from a user/developer opening a ticket on the Service Portal). The Orchestration layer is on the top the hierarchy and communicates directly with the SDN Controller through NBI and with the NFV-MANO through its NBI. The orchestration/application layer may consist of a single application or a group of different cooperating applications. These applications assume the roles of service orchestrator, network monitor and manager and may be written in any language that can communicate using the NBI of the MANO and SDN blocks. The NBI usually provides a REST API other Open APIs published by the developer of the SDN and MANO tools.
2. Based on the service description, the application asks MANO to instantiate the virtual machines and the VNFs that are needed for the network service. The VNF and NFVI communicates using the ETSI-defined reference points (see Chapter 8).
3. Once the VNFs are created, MANO programs the virtual switch to interconnect the VNFs using the Virtual Link Descriptor information.
4. When the VNFs are provisioned and enabled, this generates a topology for the Virtual Network Service. At this point, the network forwarding plane is created, and it can be a pure Layer 2 network, a VXLAN-based network, a Layer-3/IP-based network, or an MPLS-based network. The network is ready to perform the functions, such as firewall, load-balancing, NAT, etc. that are all in place. Though the network is using the actual physical network (part of the NFVI) as its underlay, this network itself may be used as an underlay for the service layer that uses SFC to provision a service overlay: in other terms, SFC provides an application-based routing overlay running on top of a network function virtualization underlay, and the latter "sees" the physical network as its underlay.
5. The application involves the SDN/Service Function controllers at this point and uses it to provision the service path for the traffic based on the defined policies. This communication from the controller to the VNFs uses the SDN southbound protocols that were introduced in Chapter 7. Network Configuration Protocol (NETCONF), RESTCONF, and gRPC are the most popular choices. Other protocols, such as XMPP used by Juniper's Contrail, PCEP, OpenFlow, or Open APIs may also be used.

The five steps outlined above could also be represented as different stages of a network topology transformation, as shown below:

- Step 1: The starting point is the physical infrastructure, which gives the topology view, shown in Figure 9-5 and serves as the original underlay for NFV

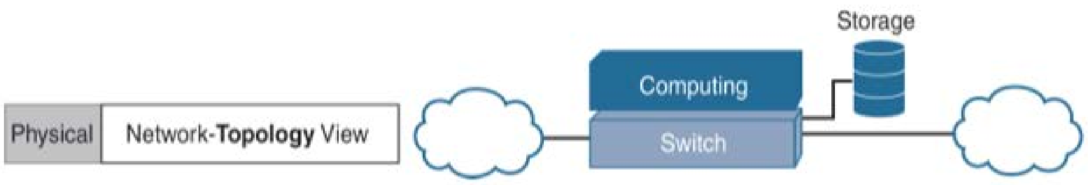


Figure 9-5 – Step 1: View of the Physical Network-topology [47]

- In Step 2 and 3, the NFV overlay is created on top of the Physical network and presents the virtual network topology view shown in Figure 9-6: a fully functional network with all the VNFs interconnected in the desired topology to offer a service.

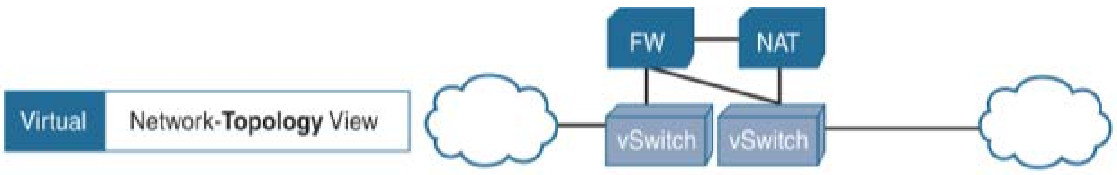


Figure 9-6 – Step 2 and 3: View of the Virtual Network-topology [47]

- Step 4. To the end user, the interconnection of the VNFs is not significant, but it is more useful to know what service this offers— shown as virtualized network service view in Figure 9-7.

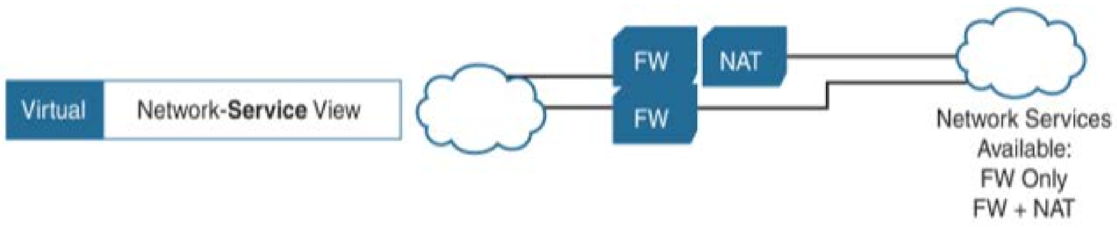


Figure 9-7 – Step 4: View of the Network-Service [47]

- Step 5: Finally, when SFC is implemented, the service topologies are logical networks offering different services to the traffic depending on traffic type, metadata, and other higher-level information. This is implemented as a policy for traffic forwarding and processing and can be referred as a virtualized service policy view.

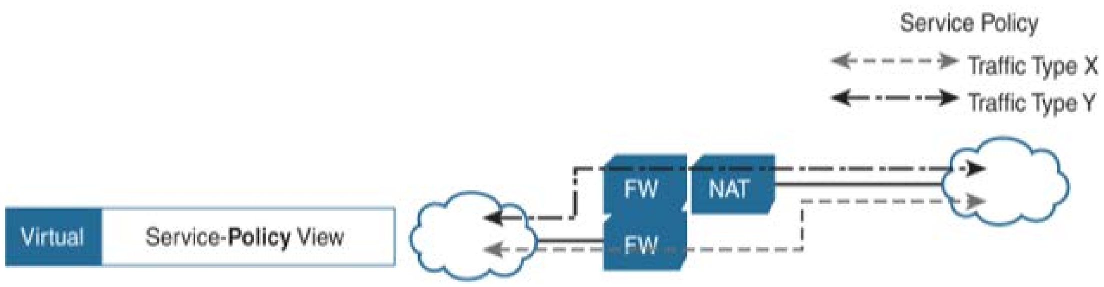


Figure 9-8 – Step5: View of the Service-Policy [47]

9.3.2 Enable Network Self-management and monitor

Once the infrastructure is Network-as-a-Service capable, the orchestration layer (more precisely a set of applications accessing the Northbound API of the MANO and SDN controller) can take up the role of monitoring the network. The monitoring can be at different levels: for example, monitoring the VNFs for the functions states and, monitoring the virtual machine states, and monitoring the infrastructure. These applications can be programmed to take autonomous decisions based on information in the monitoring data. The following use cases show how this arrangement could benefit:

- A traffic path change may be required to handle a certain traffic stream, a bandwidth demand surge, or a network fault. This decision to change the traffic path can be made by the logic in the application, and it can then be propagated to the device through the SDN controller. A typical use case could be identified in SD-WAN, where, due to WAN constraints, some data flows (i.e. videocalls) could be routed on a different path, shaped or de-prioritized based on application metadata, or Enterprise requirements like enterprise-wide adaptive call admission control.
- An increase in demand (expected or unexpected) overloading the VNFs resources can be detected by MANO, and this information can be used to trigger VNF elasticity. This can be done by the MANO’s blocks or by control applications based on specific global policies.
- A fault in the VNF’s (or host’s) code can result in a potential impact to the network. If the application is programmed with the intelligence to identify and fix the fault, it can automatically remediate the error condition and restore or protect the network.
- The application could also allow the user, operating support system/business support system (OSS/BSS), or other applications to interact with it and request changes to the network service, scale, or topology. These inputs could result in the application translating the request to the exact change needed and then send the instructions to MANO or SDN for implementing the changes.

9.4 Security Considerations

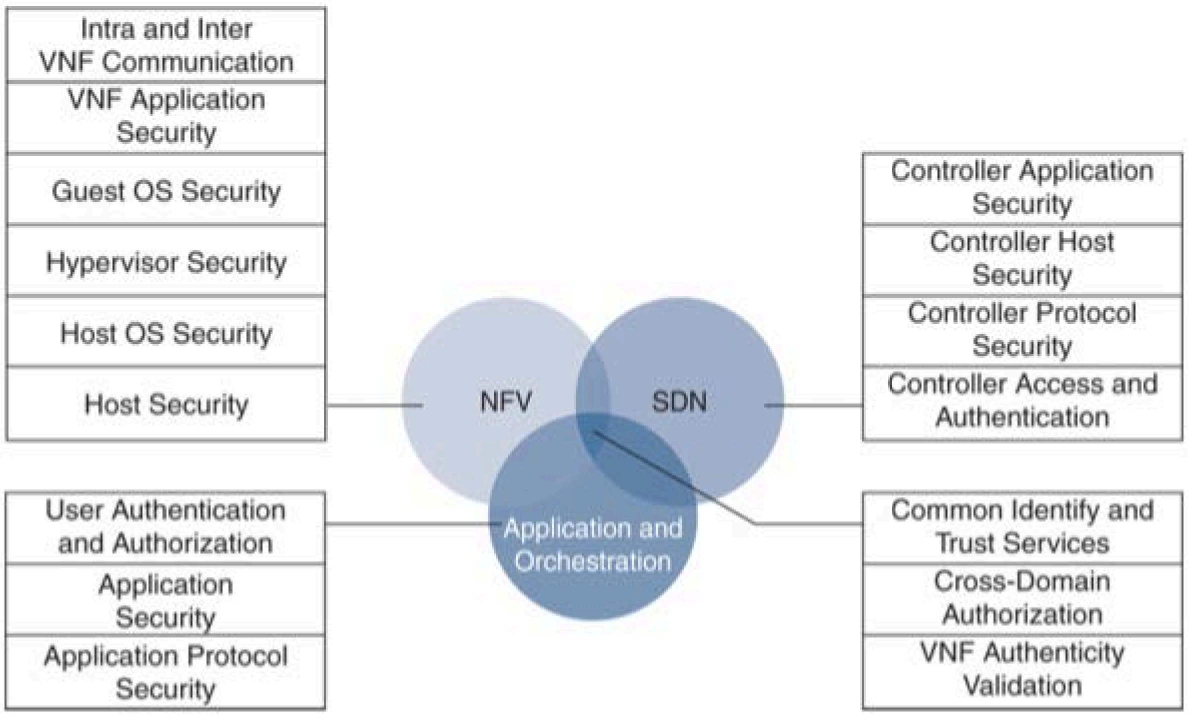

Intra and Inter VNF Communication
VNF Application Security
Guest OS Security
Hypervisor Security
Host OS Security
Host Security
User Authentication and Authorization
Application Security
Application Protocol Security
NFV
SDN
Application and Orchestration
Controller Application Security
Controller Host Security
Controller Protocol Security
Controller Access and Authentication
Common Identify and Trust Services
Cross-Domain Authorization
VNF Authenticity Validation

Figure 9-9 - SDN and NFV Security Considerations [47]

Some of the basic security requirements are:

- Intra- and inter-VNF communication: The communications traffic between VNFs can take two paths, one within the same server, using virtual links, and the other between servers, using the physical links. Security measures must be defined in both scenarios for intra- and inter-VNF communication.
- NFV infrastructure: The host operating system (OS), hypervisor, firmware, and BIOS must follow the standard best practices for the System Development Lifecycle. The external access to the infrastructure must be secured.
- SDN protocol security: The traffic from the SDN controller to the NFV infrastructure must be secured. Proper measure must be taken and policies for encryption and authorization must be implemented. As an example, even if OpenFlow encryption is not mandatory, the use of TLS between the switch or end device and the controller implements a reliable device authentication, secures the control protocol and prevent eavesdropping and man-in-the-middle attacks.
- SDN controller security: The SDN controller is an application running on a Host or Virtual Machine (VM) environment, and this environment can be secured adopting the same strategies described in the NFV infrastructure.

- SDN application security: the application needs to be periodically assessed against any known vulnerability and proper measures need to be taken. As an example, provided that Open Daylight Controller (ODL) is based on Java, any security threat and vulnerability in Java needs to be evaluated and patched to ensure ODL is not vulnerable.
- User and administrator AAA policies and platforms: the AAA platform must support multiple domains: compute infrastructure, VNF, Orchestrator, SDN components, hypervisor, and applications. Each of these domains may be managed from different administrative or operational groups from both regulatory, compliance and operational perspectives. If the NFV infrastructure is hosting multitenant, then AAA of each tenant must be incorporated and comply to the required level of security of the corresponding administrative unit.
- Common security policy: As the multiple domains are tightly connected, a specific user may need to access multiple domains with different privileges. Security controls, policies and platforms must provide this flexibility using tools as Single-sign-on authentication, Role-based access controls, non-repudiation, secure logging and accountability.

SECTION III- DESIGN OF AN ENTERPRISE DATACENTER NETWORK

10 The Distributed Datacenter Model (DDC)

10.1 Abstract

IT Infrastructure services are going through a sea of changes in terms of how services are delivered, managed and consumed. Within Enterprise-level organizations, IT teams are moving towards IT as a service model. This model offers significant advantages utilizing Cloud or Cloud-like services which lead to a distributed consumption architecture where the concept of datacenter itself changes. This chapter presents the "*Distributed Data Centre (DDC)"* Model, by which an Enterprise can consume and provide IT infrastructure and IT services across internal and external providers. This chapter describes the DDC as a founding architecture for the software-defined enterprise datacenter. An introduction to CNF from the design standpoint can be found in [97] and in [98].

10.2 Applications and services landscape

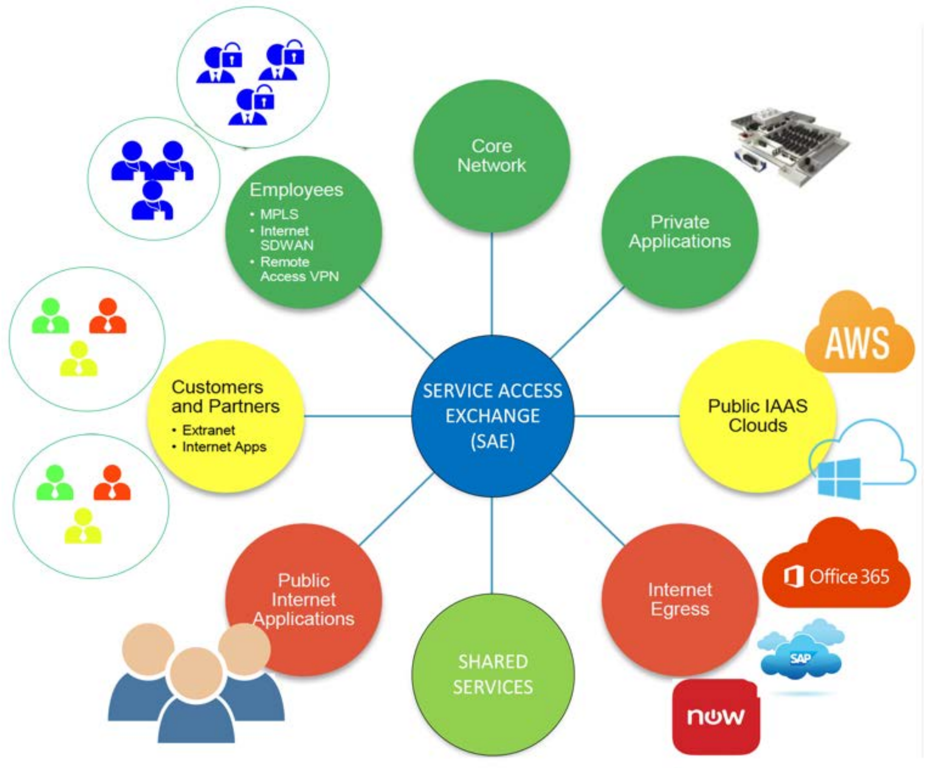


Figure 10-1 – Distributed Datacenter Topology – high level

As shown in Figure 10-1, Enterprise applications are split across private cloud, IaaS, and public SaaS/PaaS. The relative distribution of these three consumption models depends on how fast the Enterprise is adopting IaaS and SaaS/PaaS services. To improve elasticity, efficiency and reliability:

- Application consumers and providers are terminated in a Carrier Neutral Facility (CNF) providing connectivity to all the parties (Internet access, Cloud Providers, MPLS WAN, partners, etc.).
- The CNF interconnects all the providers through a transport network, called Service Provider Network (SPN).
- The SPN is connected to the Consumers through an Inter-Zone Gateway (IG).
- IG enforces centralized security policies between groups of varying security levels.
- The combination of IG, SPN and links is called Service Access Exchange (SAE) and located in a (network of) sites located all over the globe and interconnected to each other, called Carrier Neutral Facility (CNF)

10.2.1 Colocation and Cloud computing in the enterprise

"Cloud computing and colocation to Carrier Neutral Facilities are natural-born allies in the Enterprise Market. It's not a choice between alternatives, but a case of one supporting and dragging along the sales of the other. The four points below, among others, drive to an infrastructure world composed of varying percentages of colocation and cloud " [99]:

- Cloud computing must be hosted somewhere: while some cloud providers may build and maintain their own datacenter, they also (unexpectedly) rent a large chunk of space from colocation companies.
- Only few enterprises expect to go full in the cloud soon. Commit to a strong long-term investment on a datacenter may have a significant impact on capital: many companies see colocation as a more-flexible option compared to the long-term investment required for a Datacenter. As enterprises shrink their datacenters by moving some workloads to IaaS, PaaS and SaaS clouds, they will reach a point where they must determine whether it's cost-effective to operate a datacenter that is only partially at capacity.
- As enterprises understand that they will be deploying cloud computing in some way, if not now, soon, they review their long-term plans for on-premises datacenters.
- For different reasons, mainly ability to change, independence from a single provider, features differentiation, cost and appeal, Enterprises usually see themselves in a multi-cloud future. Although there are alternatives to colocation centers for connecting securely at high speed to the cloud, if an enterprise expects to build applications using the resources of multiple clouds, for latency and flexibility sake, then colocation is one of the only viable option to support this transformation.

10.2.2 Service Access Exchange

The Service Access Exchange (SAE) shown in Figure 10-2 combines links, local infrastructure, SPN and IG located in a colocation facility (Carrier Neutral Facility, CNF), providing reliable interconnections to different providers with the following characteristics:

1. It is a virtualized, orchestrated and automated communications hub where application consumers and providers meet.
2. It groups connection types by internal, external known with business relationships (third parties), and external unknown
3. Creates standardized connectivity models for employees, partners, customers, private datacenters, SaaS clouds, applications, and Internet
4. Orchestrates repeatable connectivity models
5. Publishes connectivity models in a Service Catalogue
6. Provides the agility, scalability and reliability requirements enabling adoption of the cloud, and at the same time moves the cost model from the Capex to Opex (See 2.2.5)

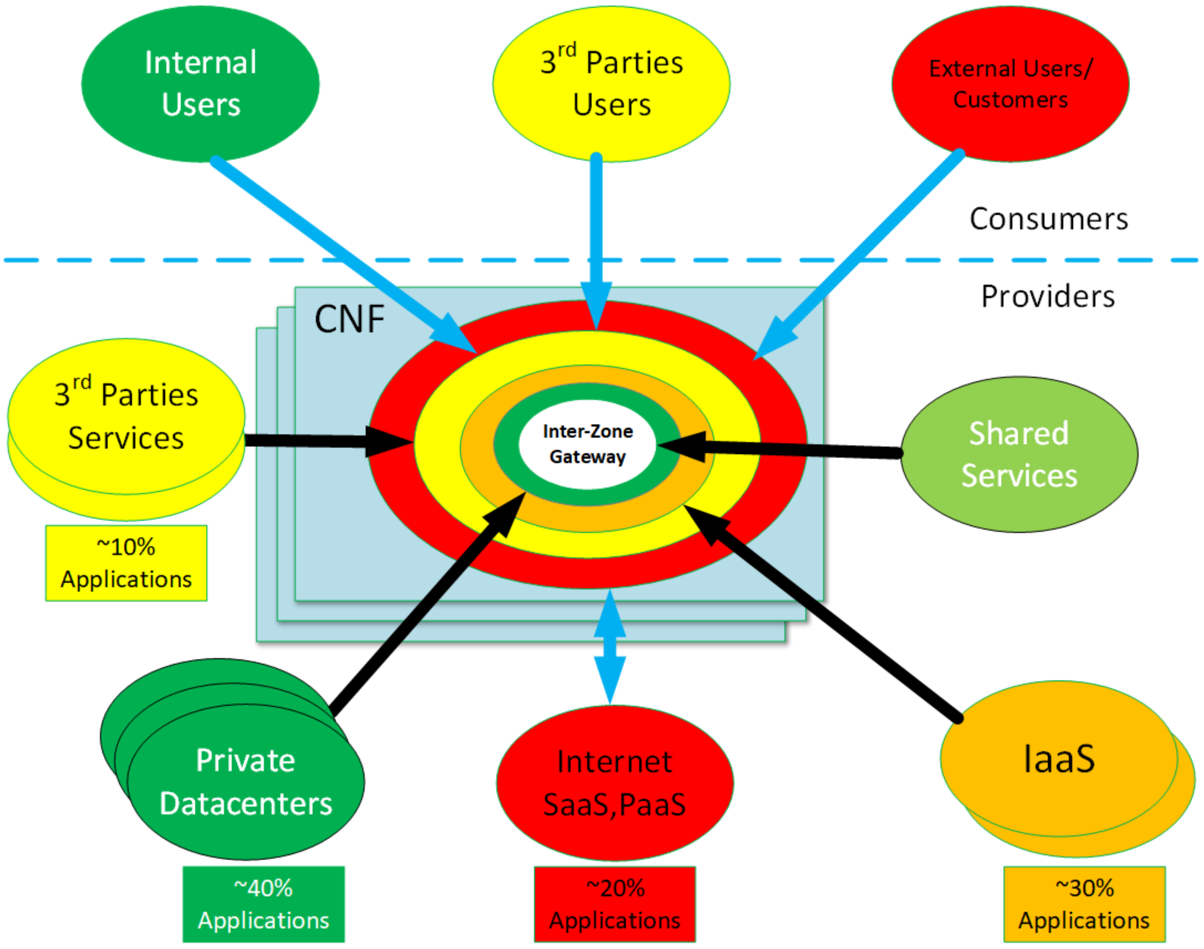


Figure 10-2 – Service Access Exchange

10.3 Distributed Datacenter Topology Diagram

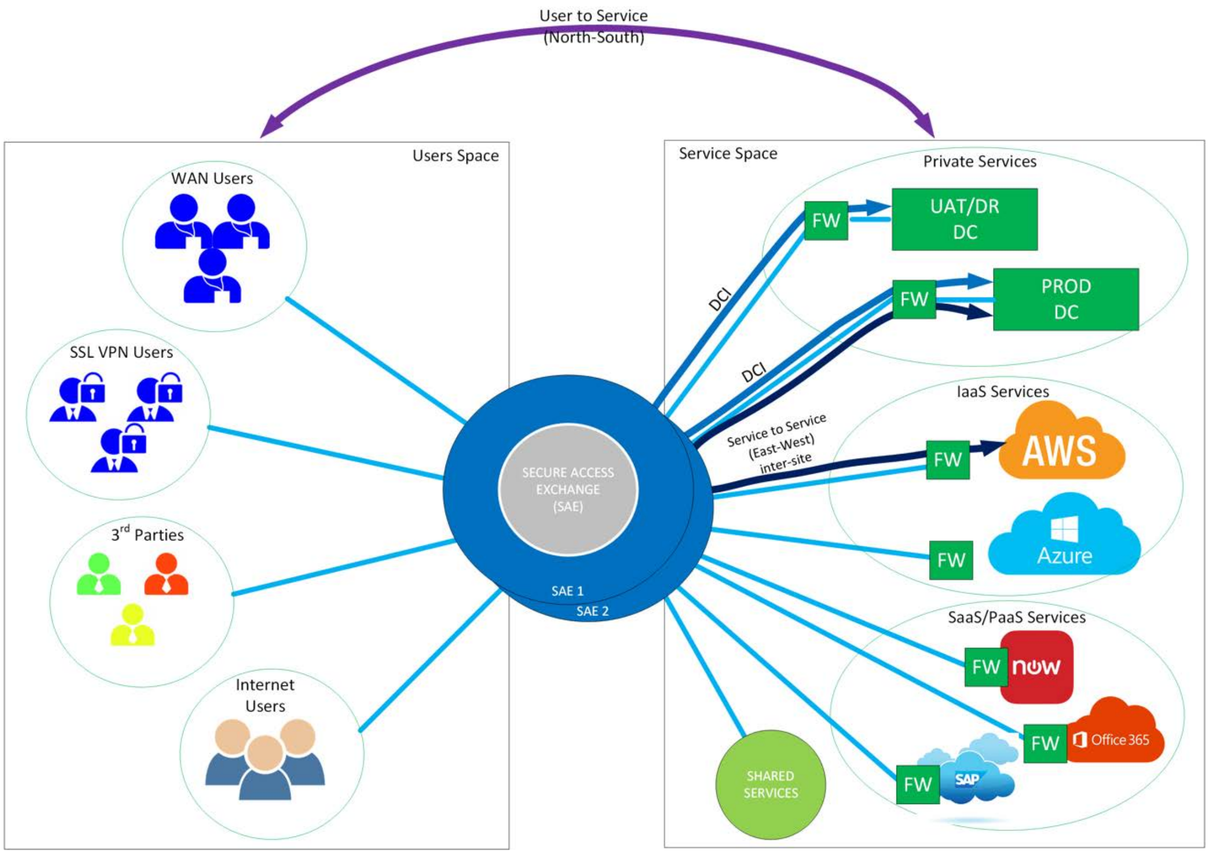


Figure 10-3 - Distributed Datacenter Topology

In the DDC model, SAEs are connected to each other through a redundant inter-regional network. Each one of the four regions defined in par. 4.6 (EMEA, NA, LATAM, APAC) has (at least) two private Datacenters, one for PROD/DEV and the other for UAT/DR and each DC is connected to at least two CNFs in the same region. The SAE could be configured as Active/Active (i.e. using VXLAN or other overlays) or Active/Standby with status/session replication over the SAE Global network (see Par. 4.6)

From a high-level perspective, the following flow types could be identified in every region

1. North-south: user to service
2. East west Traffic: service-to service, that could be divided in four different categories
   i) Private intra-provider, single location single trust: this traffic does not reach the SAE, is segregated inside the provider (i.e. east-west inside the private datacenter)
   ii) Private inter-provider multiple locations, single trust: this is typically backup or synchronization traffic through the Datacenter interconnect (i.e. backup traffic, DB synchronization, etc.): this traffic usually does not traverse the SAE, even though in some models the DCI is provided through the CNF as well.

iii) Private inter-provider multiple locations, multiple trust: this private traffic traverses the SAE, i.e. PROD traffic going from Private Cloud to IaaS, or 3rd party provider to Shared Services (i.e. Authentication requests). Depending on the source trust level and destination trust level this traffic might or might not traverse the IG.

iv) Hybrid inter-provider: these traffic flows have one leg originated on a Private network (i.e. IaaS) and the other leg on a Public network (i.e. PaaS). This traffic traverses the IG.

10.4 Service Space

10.4.1 Private Datacenters

10.4.1.1 Technical Integration and Management (TIAM)

From a technical integration point of view, a Private Data Centre model is required to operate different service categories:

- Infrastructure services: Network, Storage, Compute
- Security (CIA) Services: Confidentiality, Integrity, Availability (Backup, DR, Monitoring)
- Service Aggregation and Orchestration

The systems inside the datacenter (storage, network, applications) are categorized as Production (PROD), Development (DEV) and User Acceptance Test (UAT). This segregation is maintained across all the service categories and all the platforms (typically Wintel, Linux, AIX and other *nix, and mainframes)

10.4.1.2 Network Services

The Distributed Datacenter model will include the following network services:

- Connectivity: To operate in a Distributed Datacenter model, each provider will be connected using MPLS or P2P connectivity. This enables seamless connectivity between each provider.
- Global Server Load Balancing: solutions such as Local Server Load Balancing do not have much bearing on federation, however, if the applications are distributed across different providers, then it is recommended to have a GSLB solution from the same vendor (I.e. standardize) across the providers environments. Solutions such as DNS load balancing can be considered and standardized, Global Server Load Balancing or GTM. Having the same vendor across different providers enables the Enterprise to leverage enhanced intelligent traffic management.

- Data Centre Interconnect (DCI): With the SAE model the DCI architecture could be simplified by leveraging the CNF infrastructure, however a careful planning of this service would require considering the overall Enterprise's DC strategy (How the applications are distributed, Storage replication strategy and DR).
- Software Defined Networking (SDN), NFV and SFC: While Software defined networking is used to decouple the physical network from the logical network and more in general to separate control and data plane and provide location independence, different SDN solutions between the providers will have an adverse impact on workload mobility. Moreover, introducing a higher-level hierarchy as SAE might require, in the medium/long term the definition of a Hierarchical SDN model, specifying how/at what level the software defined SAE network would interact with the private SD-DCN. On the Datacenter domain, at least three different networks will converge in the CNF: SD-DCN in the private datacenter, global SD-WAN and SD-SAE component in the CNF.

10.4.1.3 Storage

Most of the storage services are local to a provider. The Enterprise consumes the storage services from each provider, however, storage replication impacts DR. Technically, storage replication is achieved between heterogeneous arrays. The Enterprise will devise a DC strategy. Replication is between the same providers based on The Enterprise's RPO rather than devising a strategy to replicate between providers.

10.4.1.4 Security Services

Confidentiality and Integrity: Security is a core function. From a technical standpoint, The Enterprise consumes some of the security services directly from the vendor, such as Network and Application security. The Enterprise will ensure that guidelines and polices are defined and audited centrally. Each provider will adhere to these guidelines and policies.

In a Distributed Datacenter model, The Enterprise owns the Security Operations Centre. The Enterprise consumes the network and application security services from each provider. The logs from each of the provider's environment must be pushed to a central Security Operations Centre. This ensures The Enterprise has complete visibility of security events and can perform required analysis.

Availability. In a Distributed Datacenter environment each provider may offer backup and recovery services. The Enterprise defines the backup policies which the providers will adhere. However, if applications are distributed across different providers and there is future workload mobility; then backup of the same application can be in two different backup solutions. If the Enterprise ensures that workload mobility is between the same providers and not across the backup solution, standardization is not required across the Distributed Datacenter environment. With this model, The Enterprise will leverage the backup solution from each provider. In so doing, The Enterprise may also face challenges when exiting the contract. The Enterprise may have to maintain the backup solution for recovering old data. One way to address this issue is to understand the risk and ensuring that the provider has a contractual obligation to recover all the data before exiting the contract.

10.4.1.5 Disaster Recovery model

For the sake of this paper, the private DCs Availability strategy is 1+1 with offline copy to a third site. The deployment of this Strategy will be accomplished through the operation of the primary DC designed with 2 Intra-DC HA zones, paired with one DC providing Intra-Region DR and offline copy to third site.

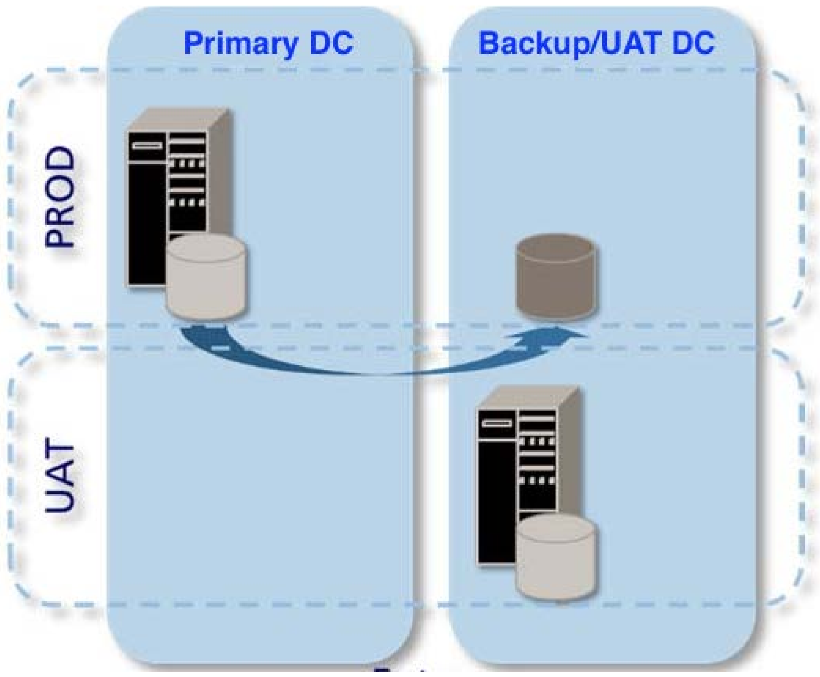


Figure 10-4- DR Model: Primary (PROD, SIT) and Secondary (DEV, UAT, DR) DC.

Extremely stringent service recovery time objectives (RTOs) and data recovery point objectives (RPOs) drive Intra-DC HA design, while intra-region DR design is driven more by survivability of mission-critical applications in case of large regional disaster situations rather than single DC failure. The residual risk is managed through Offline copy to a third site. Disaster Recovery Capability.

In each region, only one datacenter will be classified as Primary - *production active*. This enables a better cost optimization in allocating the bulk of the efforts and resources to run smooth and provide the needed high availability or 24 by 7 operations.

User Acceptance Test, (UAT), mimics very closely the production environment. It will not have the full capacity of the production environment but has all the characteristics and identifying properties of the production environment. The decision on reuse of UAT resources sacrificing them to recover operations capability is aligned with, and is an example of the principles presented in par. 4.2

10.4.1.5.1 Sacrificial UAT

UAT systems in the secondary datacenter will be sacrificed to host PROD workloads both in case of real DR and DR Test invocation

- During DR Test, only the UAT systems under Test scope will be sacrificed
- During Real DR invocation, all the UAT systems hosted in the secondary datacenter will be sacrificed to host PROD workloads

10.4.1.5.2 ISO DR Section

The secondary DC will host a section, (Isolated DR test Environment) able to host (on demand) PROD workloads. The UAT systems will be re-purposed to host PROD workloads and logically connected to this section.

10.4.1.5.3 DR Test

The DR Test context in the secondary DC will share the UAT infrastructure. This implies that during DR testing, the UAT resources under test are unavailable. It is critical that testing mimics production. A key point of the testing is that it should include the required complex changes for successful DR testing. Real disaster procedures should be simplified to reduce risks. Enough complications will arise from unexpected shortage of human or technical resources. Another key point is that, to avoid any impact on Prod systems, in Normal Mode of Operations (NMO):

1. The ISO DR section will be kept fully isolated from the datacenter network
2. Only the staff performing the DR test will be able to access this zone.
3. Only the UAT systems hosting applications whose PROD counterparts are under the scope of DR test will be shut down, logically connected here, and then re-purposed to host PROD workloads.

10.4.1.5.4 Real DR invocation

During real DR invocation, (aka DR Mode of Operations, DRMO):

- All the UAT systems will be sacrificed to host PROD Workloads and logically connected to the ISO DR Test environment

- PROD workloads will be brought up in the “ISO PROD” context
- ISO PROD context Isolation will be broken, and this context will become the “new “PROD brought up in the DR Datacenter.

10.4.2 IaaS Services

As already mentioned in this paper, IaaS (see Par. 3.6.2.1) is a cloud solution providing the basic infrastructure able to host virtual servers and/or virtual machines. Usually IaaS services are connected to the CNF through a direct connection to the closes IaaS Datacenter. As an example, in case of Microsoft Azure, there will be a redundant link (“*ExpressRoute private peering*”) connecting the CNF to the closest Microsoft Datacenter.

10.4.3 PaaS/SaaS Services

As mentioned in Par. 3.6.2.2 and 3.6.2.3 with PaaS infrastructure, users can access the applications (i.e. a database server) without needing to have direct access and/or manage the underlying platform and operating system. With SaaS Customers do not manage the infrastructure, not even the application, they just consume the services provided by the platform (for example, the MS-Office suite in the case of Office365.

In the CNF scenario, PaaS and SaaS will be services accessible on the public network, usually provided through a dedicated high-speed link with the PaaS/SaaS provider. As an example, in case of Microsoft Azure/PaaS and Microsoft O365/SaaS, the two services are usually provided through a redundant link (“*ExpressRoute Public/Microsoft peering* “) connecting the CNF to the closest Microsoft Datacenter

10.4.4 Shared Services

10.4.4.1 DNS and IPAM

DNS and IPAM are treated as a horizontal layer cutting across different providers. In other words, these services are owned directly by the Enterprise or by a single provider, and each provider consumes them both externally (i.e. public IP allocations) and internally (address allocation and hostnames)

10.4.4.2 NTP

Time is uniformly synchronized across each of the providers. NTP is seamlessly extended across the Enterprise environment

10.4.4.3 Authentication, Authorization and Identity Management

The Infrastructure is spread across multiple providers. Depending on the type of service/SLA or compliance, The Enterprise sets guidance on where the infrastructure is hosted (Ex: Dev on Public Cloud). However, Active Directory and any other user authentication (TACACS, LDAP, etc.) are treated as a horizontal service handled by The Enterprise

10.4.4.4 Service Aggregation and Orchestration

When services are consumed from multiple providers, which could be a combination of Traditional (hosted in the Enterprise Datacenter), Cloud or Cloud like services, there is a need to integrate these services through Cloud aggregation solutions providing a unified view s from different providers. Polices will be defined on how the services are consumed.

10.5 User Space

10.5.1 Internal Users

Internal users are corporate users (see par.2.4) who access the services from both corporate network, or any other location, using both corporate and personal devices.

10.5.1.1 MPLS/WAN

The Enterprise WAN is usually managed by a global contractor who provides connectivity from all remote sites to the CNFs. The WAN was usually provided through a MPLS network and deployed with WAN Optimization solution on site.

Nowadays, with the advent of SD-WAN, this approach is changing to a hybrid model, where the remote sites have dual (or more) connections and the dual link, path optimization, security services, are provided through SD-WAN + NFV solutions deployed on site.

10.5.1.2 VPN/Remote access

Remote users usually connect to the premises through SSL VPN, and /or remote access solutions using, for instance, Citrix access /VDI. The main difference between the two being that in the first case the user is terminated on a VPN Concentrator, in the second case instead the user generates two sessions: one to the Citrix infrastructure (i.e. a Citrix NetScaler with or without WAN optimizer), the second session from the NetScaler to the target servers.

From the SAE/DDC perspective these flows are provided through standard Internet access and terminated on specific devices, VNF providing the required services.

10.5.2 3rd Parties/Partners connectivity

3rd parties/partners connect to the Enterprise either to provide services and/or to access dedicated services provided by the enterprise. Depending on the requirements, this type of access is usually provided in two ways:

- through dedicated links (usually MPLS, or IPSEC tunnels). This type of access is preferred when the partner needs to connect an entire site (Site to site VPN) to the enterprise. The

Telco provider of these connections usually has a PoP on the CNF and the links are logically terminated on a dedicated device (physical or VNF) enforcing the access policies.

- through SSL VPN with specific access for external users. These users access the Enterprise through SSL VPN terminated on a device enforcing specific and granular access policies.

10.5.3 External Users, Customers

External users access the Enterprise through public services accessible on the Internet (i.e. Web applications, web services, API Gateways, etc.). To reduce the threat surface, usually, these services are accessible through one or more third parties (i.e. INCAPSULA, Akamai, etc.) providing DDOS prevention, and only these third parties can establish a connection to the perimeter.

From the SAE/DDC perspective, these users (or the third parties managing inbound DDOS prevention) connect to the Enterprise through an Internet link terminated on the SAE.

10.6 Distributed Datacenter Global Network

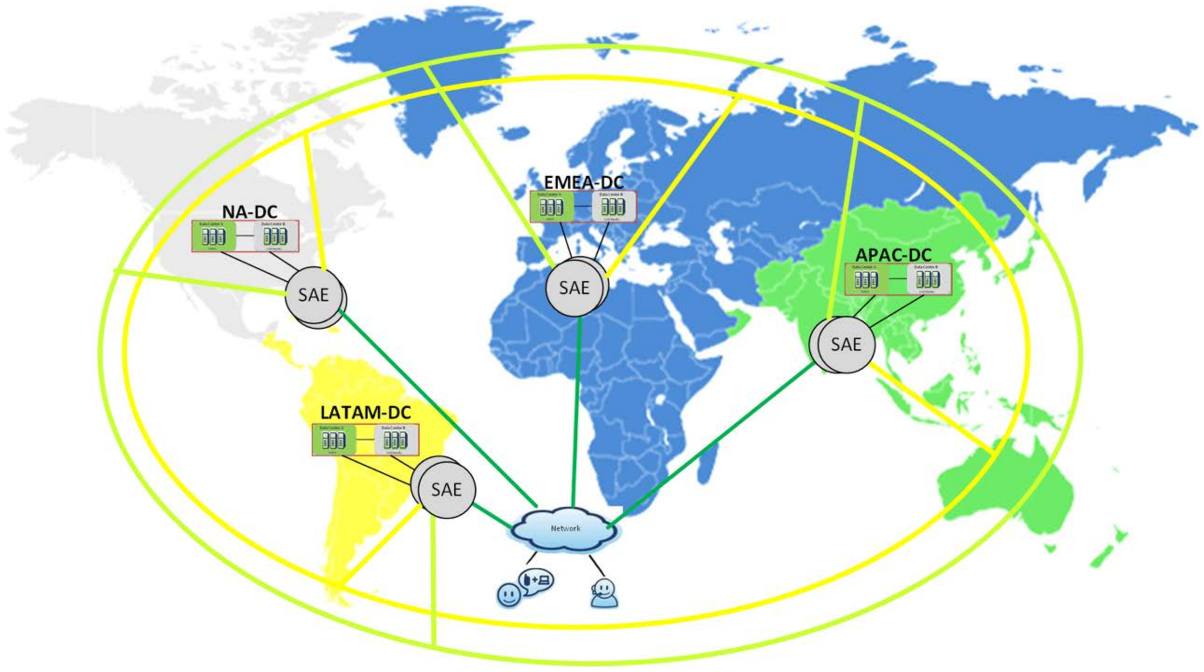


Figure 10-5 - Distributed Datacenter Global network

Figure 10-5 shows the interconnections of the SAEs in the four regions (APAC, EMEA, LATAM, NA). Each SAE is also connected to the Regional Private Datacenters, the regional providers (i.e. AWS, Azure, etc.) and to other external networks (i.e. WAN and Internet) not shown in the picture. All the SAEs are interconnected through the redundant inter-CNF network, which is completely managed by the CNF provider and out of the scope of this paper.

11 High-Level Design of an Enterprise Grade Software Defined Datacenter network

11.1 Abstract

This chapter, assuming that the Enterprise datacenter follows the model introduced in Chapter 10, details the Service Access Exchange (SAE) as a core component of the Distributed Datacenter Network (DDCN) , with the Inter-Zone Gateway (IG) and the Service Provider network (SPN), and finally presents a high-level topology for the private datacenters, and the related Network functions and NFCs that could be used in the private datacenter.

Both the IG and SPN are key components of the designs introduced in the next chapters. The high-level design for the DCN discussed in this chapter are applicable to the Private datacenters described in 10.4.1.

11.2 DDC

11.2.1 High Level Design

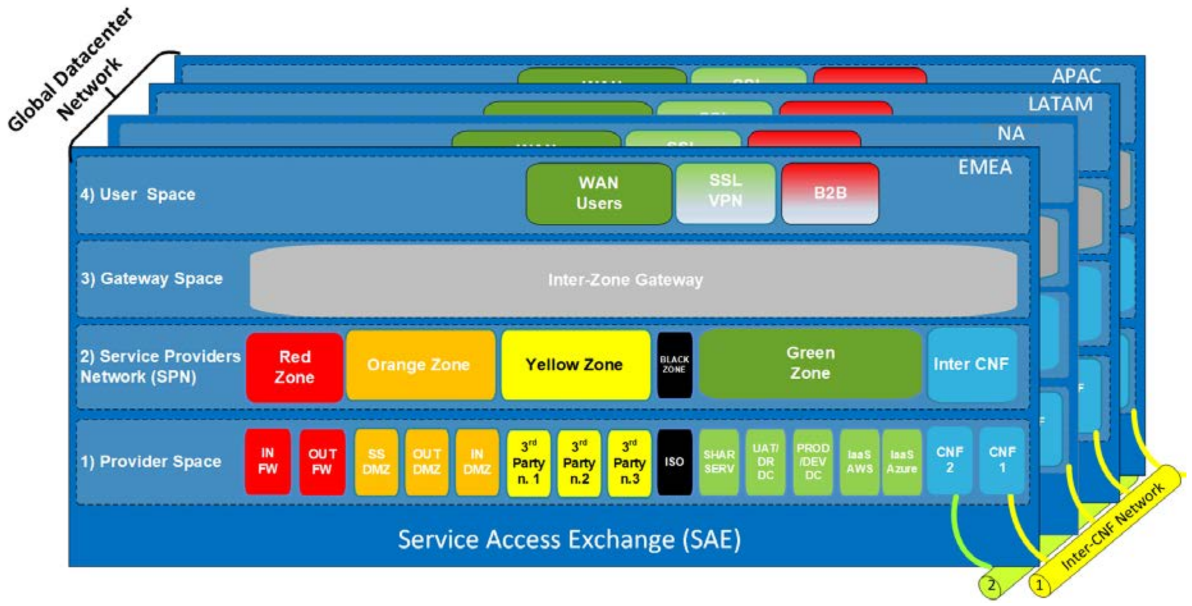


Figure 11-1 – DDC Network Block model

The block diagram in Figure 11-1 shows the main components of the DDCN described in Chapter 10. This chapter and the next three will focus on the datacenter network design for the Regional Private Datacenters (Primary PROD/DEV and Secondary UAT/DR), directly or indirectly owned by the Enterprise. However, to provide end-to-end Software-Defined Services, and to consume Datacenter-as-a-Service, the tools described in Section II (SDN, NFV, SFC, orchestration) should be applied to both SAE and private DC networks.

11.2.2 Service Provider Network (SPN)

The SPN connects various (physical and virtual) Service providers within a single SAE owned by the Enterprise. Within the SAE, different providers would connect their infrastructure. The SPN is segmented in different zones, as an example:

1. Blue zone: this is the SAE backend network, inter-CNF connecting the SAE with the other regions.
2. Green Zone: connects all the Providers offering private infrastructure services and/or facilities directly owned by the Enterprise.
3. Black Zone: connects all the Isolated networks located in different providers and Needed to provide distributed DR Test capabilities
4. Yellow Zone: connects all the resources provided by third parties managing their own infrastructure providing services to the Enterprise.
5. Orange Zone: connects all the Providers offering both inbound (i.e. PaaS, Forward or Reverse proxies) or outbound (i.e. Guest Wireless access) DMZ-located services.
6. Red Zone: connect all the perimeter devices, typically at least the inbound and the outbound Firewall infrastructure

Depending on the security and visibility requirements, different configurations on the SPN could be implemented using different Service Insertion Policies:

- If the provider complies with the Enterprise policies, or the infrastructure is directly owned by the Enterprise, then the policy enforcement point could be the context firewall. As an example, PROD traffic going from one private datacenter to the Enterprise/IaaS will need to traverse two context firewalls and these two will enforce the related security policies. In these cases, though, the SPN could provide a "network visibility" policy.
- If the provider is a 3rd party (i.e. not compliant with the enterprise security model, or with a Firewall not managed/trusted by the Enterprise IT SOC), then the SPN could be used as a policy enforcement point for the intra-zone traffic or, through service insertion, a policy might force the traffic to the Inter-Zone Gateway. It is worth noting that inter-zone traffic will always flow through the Inter-Zone Gateway, by design and by definition of Inter-zone.

11.2.3 Inter-Zone Gateway

The Inter-Zone Gateway (IG) provides an inter-segment policy enforcement point to serve as a “network supervisor” for:

1. Intra-Zone East-West traffic: the IG provides a policy enforcement point for those flows coming from 3rd parties, in the same zone that for any reason might require further inspection (i.e. 3rd party traffic to another 3rd party using non-standard ports)
2. Inter-zone East-west: the IG, by design, enforces security policies for data flows between different providing, isolating fault domains
3. North-south (User-to-Service): the IG enforces security policies for any user-to-service flow. All Enterprise users in each region must pass through a IG when accessing different services, regardless their location (IaaS, PaaS, Private DC, etc.).

11.3 Private Datacenter

11.3.1 Summary of the Specifications

As already specified in the previous paragraphs, each region contains two datacenters: the primary datacenter hosts PROD/DEV traffic, and the secondary hosting UAT and serving DR purposes. The general specifications are

- For the sake of this paper, every DCN must be able to accommodate initially about **1,500 physical hosts (32 racks with 48 servers each) dual-homed with 10G access links (about 3,000 10G access ports)**
- Both datacenters are connected to at least two regional SAEs
- The Private Datacenter does not have any other external connection except those linking it to regional or national CNFs

11.3.2 DCN Contexts

Every regional DCN contains 4 or 5 contexts (see Figure 11-2): DCI, NONPROD, PROD, ISO (optional, usually available only in Secondary DC) and DMZ. Each one of these contexts usually contains more than one segment, one (or more) Context Firewall, one (or more) Load Balancer acting as pure load balancer, reverse proxies and other network functions (NFs) that, whenever possible, should be virtualized.

These contexts are connected to the corresponding areas of the SPN defined in Figure 11-1 through a pair (or more) links connecting the regional datacenter with the SAEs.

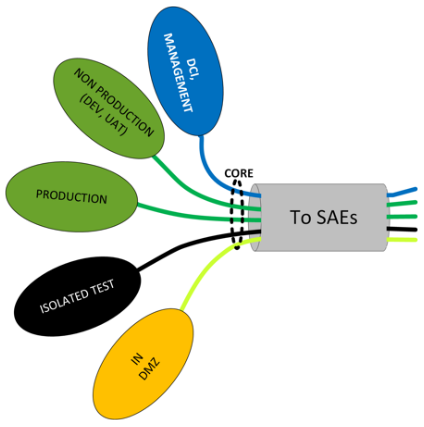


Figure 11-2 - Contexts in the Regional DC

11.3.2.1 CORE

The CORE context does not contain any service: its role is to route all the traffic to/from the internal contexts (DCI, PROD, NONPROD, and for the Secondary Datacenter also the ISO) to/from the SAE. All the contexts are connected to the CORE (and to the SAE) through a context Firewall.

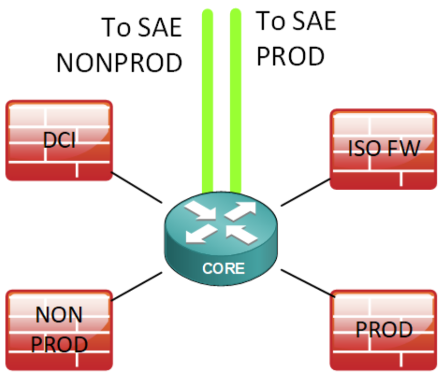


Figure 11-3 - CORE context

11.3.2.2 DCI

The DCI context contains connection to/from DCI (Datacenter interconnect) needed for synchronous/asynchronous backups, storage replication/deduplication traffic, management, ILO, etc.

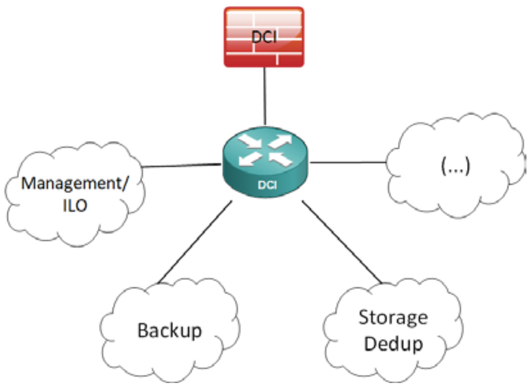


Figure 11-4 - Backend context

11.3.2.3 Production

The production context contains all the services needed for PRODuction. The PROD Context Firewall enforces the access policies on this context allowing communication to/from other authorized sources.

Due to the need of managing the Lifecycle of Legacy End of Support Systems this context might or may not also contain one sub-context where all the End of Support systems are connected.

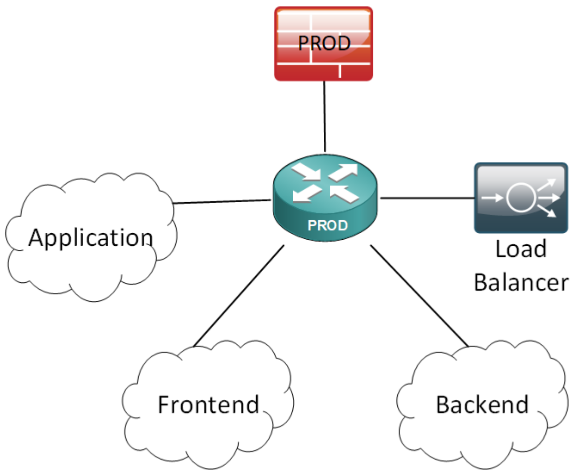


Figure 11-5 - -Prod Context

11.3.2.4 Non-Production

The non-production context contains the services needed for NON-PROD (UAT in the secondary datacenter, DEV in the primary Datacenter). The NONPROD Context Firewall enforces access policies on this context allowing communication to/from other authorized sources and, as an example, preventing UAT or DEV traffic to access PROD systems.

Due to the need of managing the Lifecycle of Legacy End of Support Systems, this context might or may not also contain one sub-context where all the End of Support systems are connected.

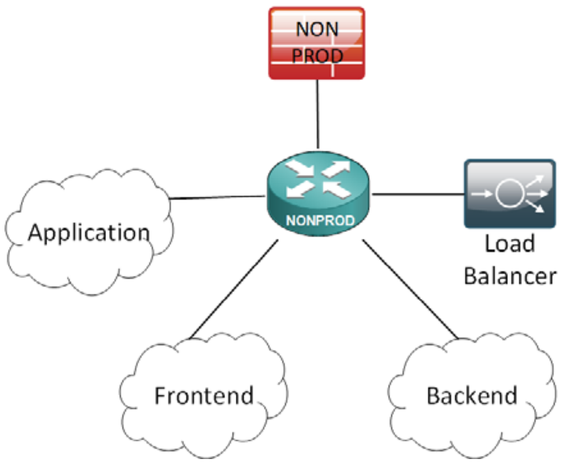


Figure 11-6 - Non-Prod Context

11.3.2.5 DMZ

The DMZ context contains inbound services accessible from outside. The DMZ context is connected to the IN DMZ in the SAE. Services located in this context are

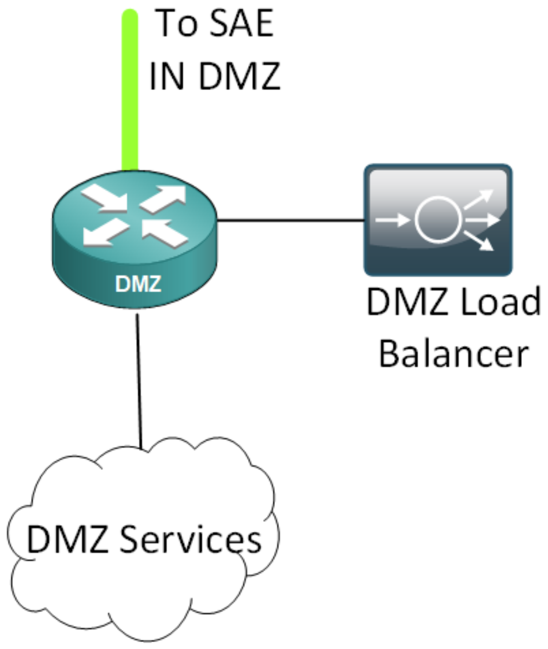


Figure 11-7 - -DMZ Context

The DMZ context terminates flows related to PROD/DEV/UAT externally facing applications. Usually the security requirements mandate that all the externally facing applications are terminated on a Reverse proxy/Load balancer which, then, can connect to the actual PROD or DEV server.

11.3.2.6 Isolated Test

The ISO Test context is available only in the UAT/DR Datacenter. Its main purpose is to provide DR Test capabilities. Essentially, this isolated context will provide a DR TEST environment where services and servers can be brought up to test the readiness of the DR procedures.

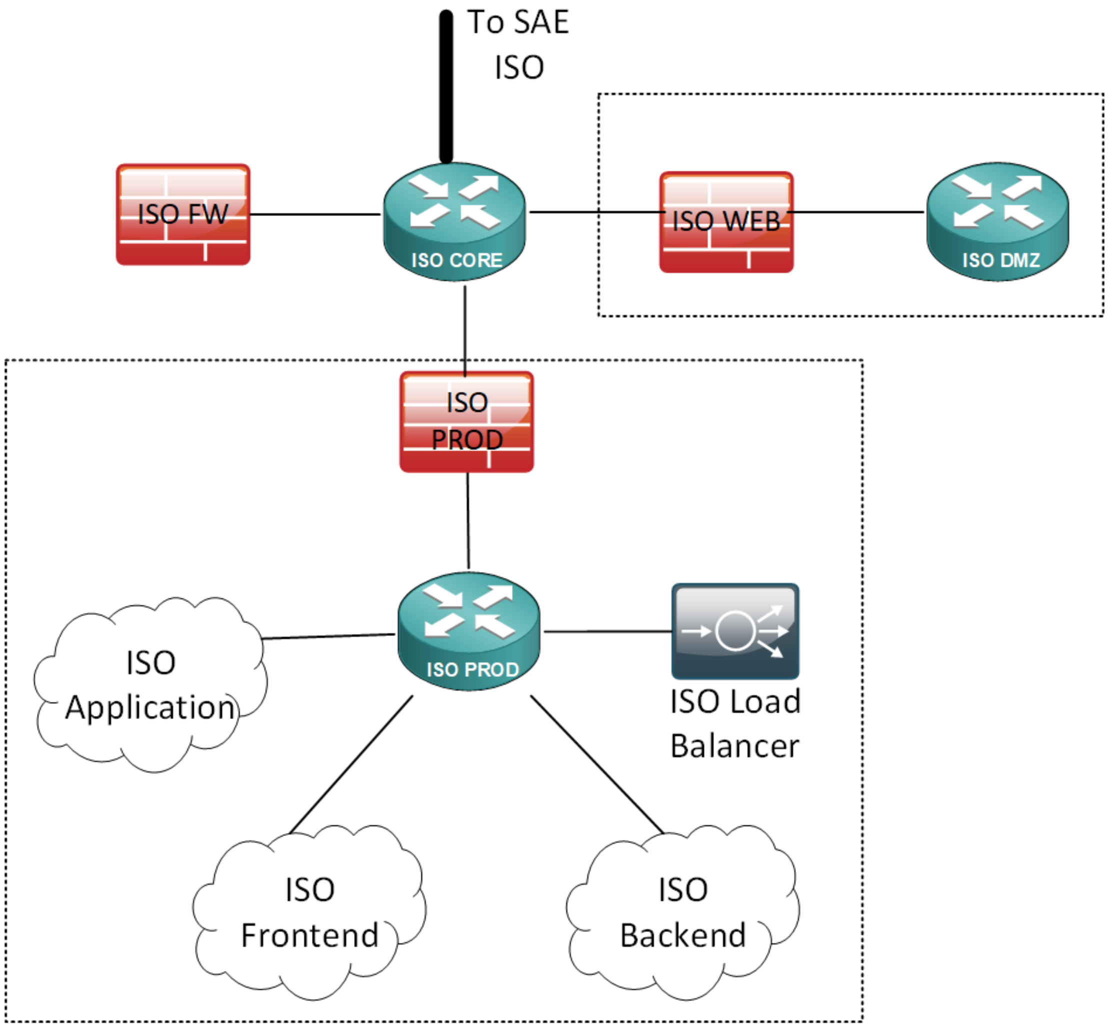


Figure 11-8 - ISO Context

The ISO Context purposes are essentially two:

1. During normal mode of operations: to proof the ability of providing DR for specific applications. By recovering the PROD applications under DR Test in the ISO Context, the segregation of these applications (and so the impact of the test) is guaranteed, and at the same time the Enterprise can perform the (often mandatory) DR Tests without affecting business flows.
2. In case of failure of the primary Datacenter, the entire infrastructure of the primary datacenter will be replicated in the DR ISO Context, and its isolation will be "broken" providing the Primary Datacenter services inside the Secondary Datacenter.

11.3.3 PROD/DEV Datacenter network topology

The primary datacenter hosts DCI, PROD, DEV and PROD/DMZ workloads

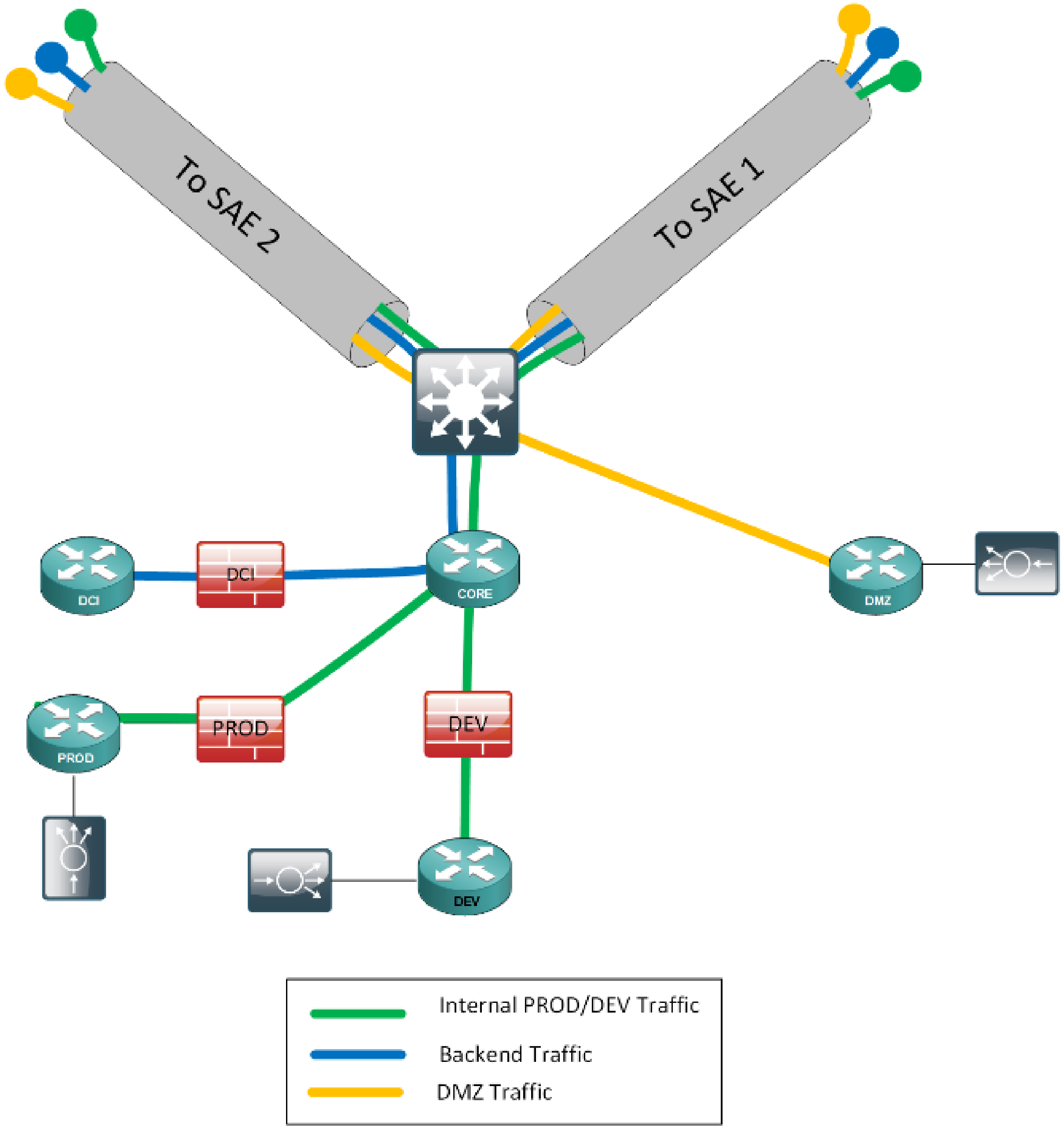


Figure 11-9 - PROD/DEV Primary DCN

The DCN networks interconnects the four contexts with both the SAEs. Flows depicted in Figure 11-9 might represent L2, L3 or specific L4 traffic, depending on the actual design. The DMZ terminates flows related to PROD/DEV externally facing applications. Usually the security requirements require that all the externally facing applications are terminated on a Reverse proxy/Load balancer which, then, can connect to the actual PROD or DEV server.

11.3.4 UAT/DR Datacenter Network Topology

The secondary datacenter hosts DCI, PROD, UAT, UAT/DMZ and DR TEST workloads

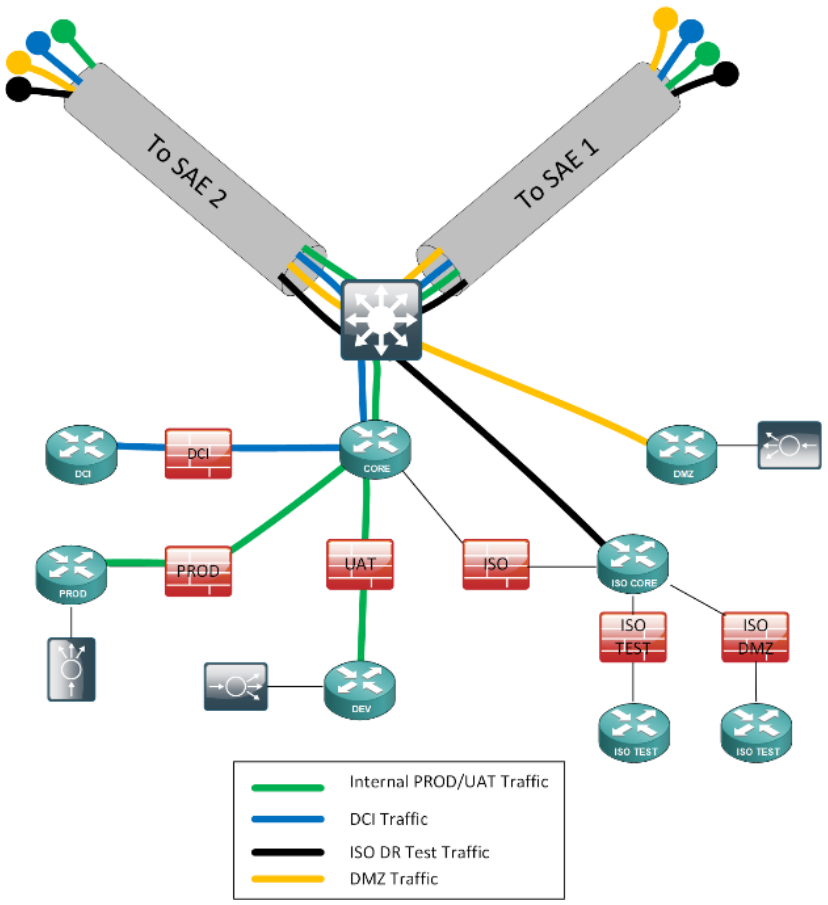


Figure 11-10 - UAT/DR Secondary DCN

The DCN networks interconnects the five contexts with both the SAEs. Flows depicted in Figure 11-9 might represent L2, L3 or specific TCP traffic, depending on the actual design.

11.3.5 Private DCN Network Functions

Different network functions are needed in the private DC. From a very high-level perspective, though, they could be summarized in the following list

- Stateful L3/L4 Firewall
- Application-Aware (L7) Firewall
- Load-balancer / Reverse proxy
- Network sensor/TAP

11.3.5.1 Stateful Firewall

The Stateful firewall is a L3/L4 function enabling specific flows based on policies defining source, destination address, protocol. It is used in the datacenter because it might be directly implemented in the TCAM of the networking devices and easily controlled by the SDN Controller.

11.3.5.2 Application-Aware Firewall

The Application-aware firewall is a L7 function enabling specific flows based on application level policies, performing deep packet inspection and other application level logic (i.e. preventing SQL injections attacks). It is worth noting that the same device might have one virtual context used as a Stateful firewall and another context used as a L7 firewall.

11.3.5.3 Load Balancer / Reverse Proxy

The load balancer is a L4/L7 function balancing the traffic on a set of target servers. The Load Balancer usually provides a virtual IP on its Frontend interface and connects to the backend targets. The load balancer can work

- at layer 4 (TCP): terminates the TCP session coming from the users on the frontend interface and establishes related sessions on the backend interface.
- At layer 5-7 (Application): the LB frontend terminates the inbound session and establishes other sessions with the backend. An example of this use is a HTTP/HTTPS forward proxy, where the users connect to the frontend, and their HTTP/GET request is processed by the LB that proxies and validates the requests and send them to the actual Web servers.

The load balancer usually has also a monitor component which polls the backend servers and it is fully aware of their status/load.

11.3.5.4 Tap

This function is needed to replicate the traffic that it sees and send it (usually but not necessarily) to a security-aware device. Typical examples of this functions are network monitoring tools used for compliance purposes and/or network traffic recording devices.

11.3.6 DCN Network Function Chains

The NFCs used in the DCN can be summarized as

- MP-FW: Multi-purpose Firewall: L3/L4/L7 Firewall and Tap
- ALG: Application Level Gateway: Tap and Reverse Proxy/Load Balancer

11.3.6.1 Multi-purpose Firewall

This chain can validate a flow correlating different information coming from the network, transport and application layer. If needed, this function can also send the related traffic to other target for further inspection/logging.

11.3.6.2 Application-Level Gateway

This Function Chain is a Multi-Purpose Load Balancer, able to send the traffic to a third-party function for further processing/monitoring/analytics.

12 Classical SDN Design using Big Switch Big Cloud Fabric

12.1 Abstract

This chapter presents the "classical" SDN (see Par. 7.4.2) design for (Primary and Secondary) private datacenters defined in Par. 11.3 using as a building block the Big Switch Big Cloud Fabric (BCF) ® architecture introduced in paragraph 7.5.4 (the technical solution presented in this chapter is based on [100]). Big Switch also offers a free demo lab for BCF [101] that has been used to validate part of the configuration presented in this chapter.

Despite its name, *Big Switch* is a pure software company, therefore it does not directly supply any hardware, and relies on commercial partners (i.e. DELL, Edge-core, etc.) to provide network gear compatible with BCF specifications and requirements. Even though the design presented in this chapter adopts DELL equipment, a similar approach could be taken using any other vendor supported by BCF.

12.2 DCN Physical Topology

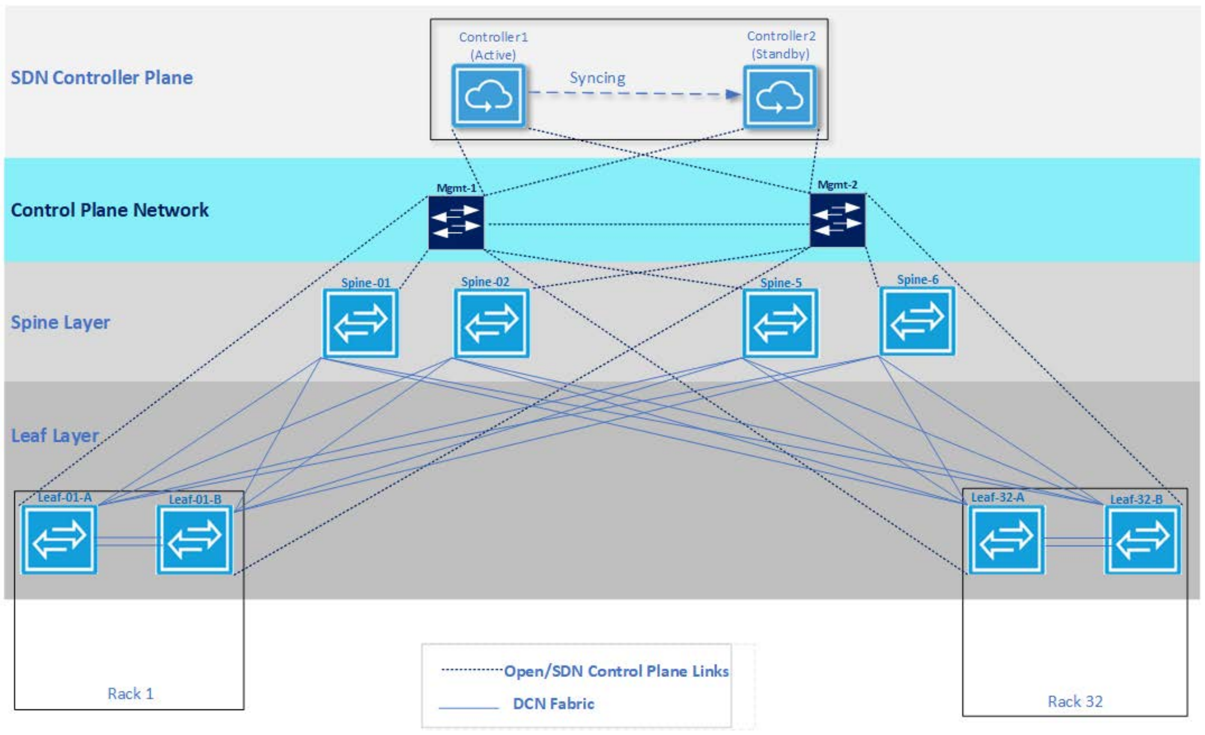


Figure 12-1 - BCF DCN Topology (Primary and Secondary DCs)

The proposed architecture is a classical SDN network based on a spine-leaf (3-stages folded Clos) topology with maximum 32 pods (racks that have servers), at the top of which there are two leaves connected to up 6 Spines, each with 64 40GbE ports. Each server is dual-connected to a leaf via 10G links.

The leaf switches are the DELL S4048-N with 48x10G access ports and 6x40G uplinks to the spine, so the maximum number of spines is 6. The spine switches are the DELL S6010-ON with up to 64x40G ports[14], so the maximum number of leaf switches is 64. Considering that every pod must be connected to two leaf switches, the maximum number of pods is 32.

12.2.1 Oversubscription

In a leaf-spine topology, oversubscription, if any, occurs at the leaf layer. Oversubscription is equal to the total amount of bandwidth available to all servers connected to a leaf switch divided by the amount of uplink bandwidth.

$$Oversubscription = \frac{total\ bandwidth}{uplink\ bandwidth} = \frac{48\ x\ 10Gbps}{6x40Gbps} = \frac{480}{240} = 2:1$$

This implies that the proposed Clos topology is blocking if all the ports are used. However, if the port-channel on the server is configured only, say, as LACP active/standby then the network becomes non-blocking (with half the access bandwidth)

12.2.2 Scalability and bi-sectional bandwidth

The presented solution supports up to 64 Leaf Switches, each with 48 10GbE Ports, which implies that a maximum of 64x48 = 3,072 10GbE active ports can be connected. The total access bandwidth is 3,072 x 10Gbps=30,720Gbps=30.72Tbps. Each one of the 64 leaf switches has 6 uplinks @40GbE, which implies that the total bi-sectional bandwidth is 6x64x40Gbps = 15,360Gbps=15.36Tbps. The solution can scale up to 3,072x10GbE ports (1,536 dual homed servers).

To scale this solution there are two options:

1. Scale up: the available documentation states that a single BCF supports up to 128 racks on a single BCF, however this size would need spine switches with 256P@40G, currently [15] not available in the commercial offering.
2. Scale out: another BCF network must be created with a new pair of Controllers and connected to the existing one through an external tenant. This solution would require, though, to define how the two (or more) SD-DCNs interact together.

[14] At the time of the writing the DELL-6010-ON is a fixed configuration switch with 32 ports, however the new release will support up to 64 ports.

[15] As per February 2018

12.2.3 Racks per pod

Every pod provides up to 480Gbps (2x6@40G ports) uplink bandwidth and up to 960Gbps (2x48@10G ports) access bandwidth. The access bandwidth might be provided as

1. 2x10Gbps access to a single server/blade (maximum 48 servers)
2. Multiple 2x1Gbps access links, in which case the actual number of physical servers per pod could go up to 480[16]
3. A valid combination of the previous

From a rack perspective, in the first case every pod would be contained in just one single rack packed with 48 servers (typically blade servers) as shown in Figure 12-2

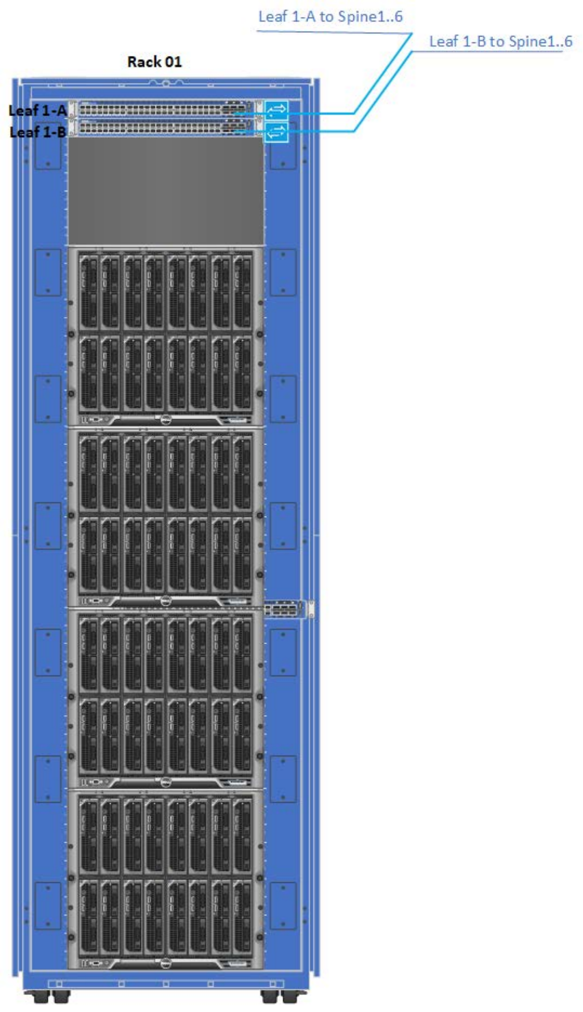


Figure 12-2 - Single Rack Pod (Front View)

in the second and third case, the pod might contain up to 10 racks with 48 servers each, all of them connected with 2x1G ports to the leaf switches, and the 2 leaf switches most likely placed in a Middle-of-Row topology (see Figure 12-3)

[16] This would require, of course, to correspondingly increase the number of racks, and provide an adequate inter-rack connectivity.

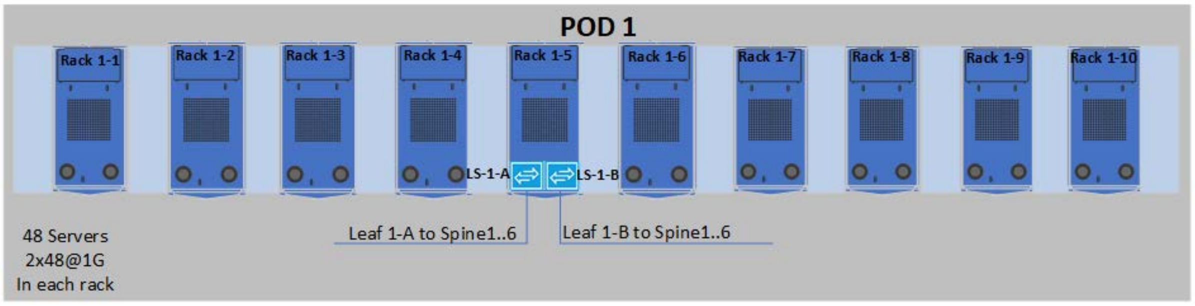


Figure 12-3 - 10 racks pod with MoR placement (Top View)

12.2.4 Network components

Big Switch business strategy is centered on working with white-box (aka brite box) switch manufacturers. These white-box switches are delivered with the Open Network Install Environment (ONIE) boot loader that can discover the controller and download Big Switch's Switch Light OpenFlow switch code. Switch Light is based on the Indigo open source OpenFlow switch software. The idea is to provide a specific architecture to the customer, placing in the market a bundle of switch, controller, and code is largely auto-configured.

12.3 Logical View of the Enterprise private DCN using BCF

12.3.1 Key logical components

To understand the design based on BCF, some key roles of the architecture must be defined:

- Tenant: an entity providing logical grouping of L2 and/or L3 networks and services, similar in function to a VRF entity (see Par 6.4.2).
- Logical Segment: a L2 network consisting of logical ports and endpoints and defining a broadcast domain boundary.
- Logical Router: an entity, defined inside the tenant, providing inter-segment, intra-tenant, and external network routing and policy enforcement services
- System tenant: the BCF "*default*" tenant. The logical router in the system tenant is called System Tenant Router.
- The System Tenant Router has only one type of interface, called a tenant interface, which enables routing between user-defined tenants

12.3.2 Primary Datacenter

From the logical standpoint, the Primary datacenter contains the system tenant interconnecting external, CORE (with PROD, DCI and DEV) and DMZ Tenants.

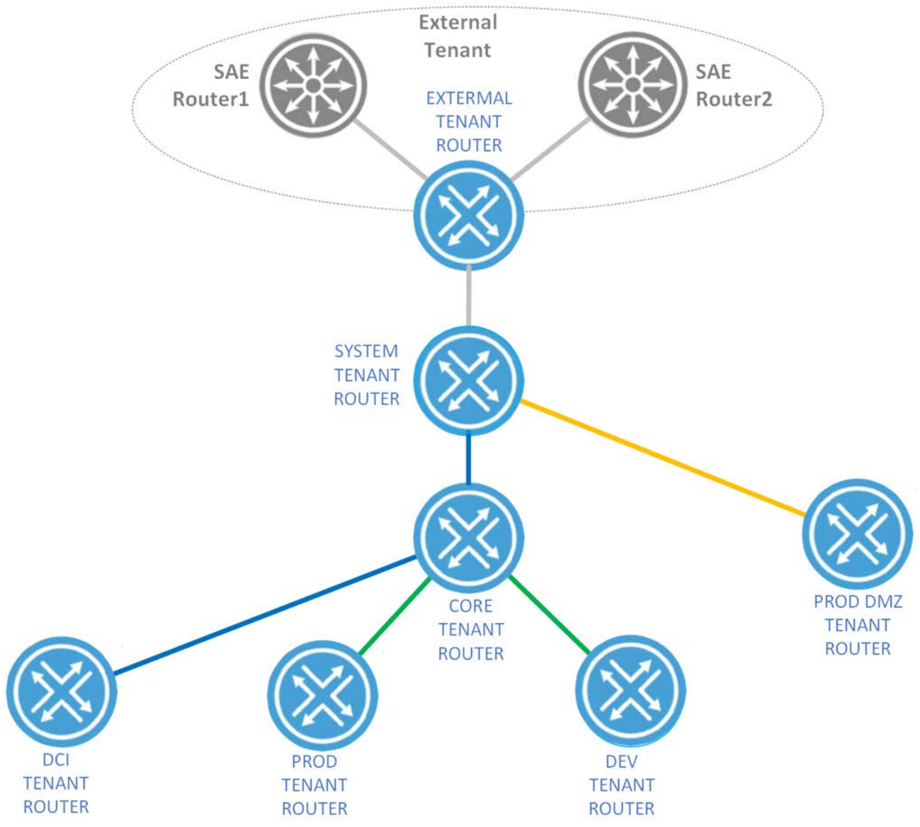


Figure 12-4 - Primary DCN with BCF – Logical View

12.3.2.1 Intra-tenant routing

In every tenant, the tenant router performs Intra-tenant routing (aka inter-VLAN routing). It is worth noting that, if needed, SFC can be deployed in the tenant router to intercept inter-segment traffic and send it to a VNF performing, for instance, Firewall or load balancing functions (see par. 12.4.2.1).

12.3.2.2 Inter-tenant routing

The inter-tenant routing is performed in two different ways:

- Inter-tenant routing between CORE, DCI and DEV, which are not directly connected to the system tenant (Tenants without a system interface): in this case a dedicated Context Firewall VNF performs inter tenant routing. The insertion of this VNF in the path is performed through BCF Service insertion (see Par. 12.3.3.1)
- Inter-tenant routing between External, CORE, DMZ which are directly connected to the system tenant (Tenants with a system interface): in this case the system tenant router directly performs the inter-tenant routing

12.3.3 Secondary Datacenter

From the logical standpoint, the Secondary datacenter contains the system tenant interconnecting the external tenant (where the SAEs are terminated), the CORE (where PROD, UAT and DCI workloads are located), DMZ (where UAT externally facing services are located) and ISO (providing DR capabilities) Tenants.

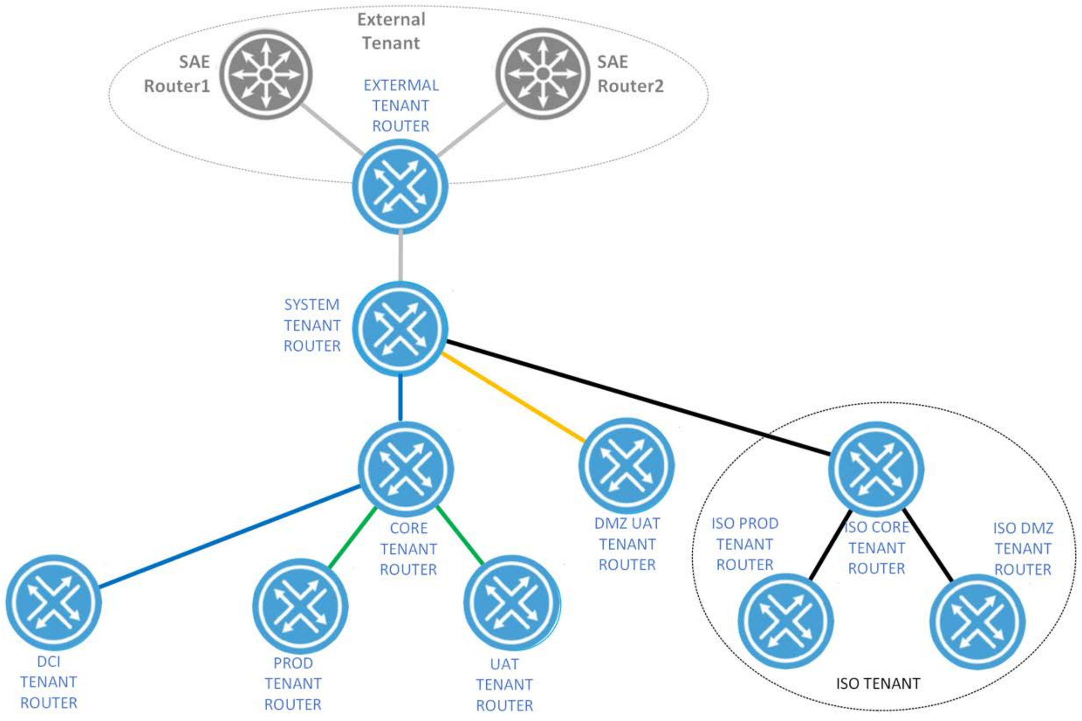


Figure 12-5 - Secondary DCN with BCF – Logical View

12.3.3.1 Intra-tenant routing

In every tenant, the Logical Tenant Router performs Intra-tenant routing (inter-vlan routing). It is worth noting that, if needed, a service insertion policy could be deployed to intercept intra-tenant traffic and send it to a VNF performing, for instance, Firewall or load balancing functions (see par. 12.4.2.1).

12.3.3.2 Inter-tenant routing

The inter-tenant routing is performed in two different ways:

- Inter-tenant routing between CORE, DCI, UAT, ISO-PROD and ISO-DMZ which are not directly connected to the system tenant (Tenants without a system interface): in this case a dedicated Context Firewall VNF performs inter tenant routing. The insertion of this VNF in the path is performed through BCF Service insertion (see Par. 12.3.3.1)
- Inter-tenant routing between External, CORE, DMZ and ISO which are directly connected to the system tenant (Tenants with a system interface): in this case the system tenant router directly performs the inter-tenant routing.

12.4 Network Function Chaining

In BCF, Service insertion is provided through IP next-hop forwarding implemented by policies defined in the logical router. The actual target of the forwarding could be a single VNF or a group of VNFs (i.e. load balanced at the IP level), the source could be all the traffic coming from a specific segment in a Tenant and the destination could be a segment on the same tenant (intra-Tenant NFC) or another tenant (inter-tenant NFC).

BCF does not support micro-segmentation and /or layer 2 service insertion.

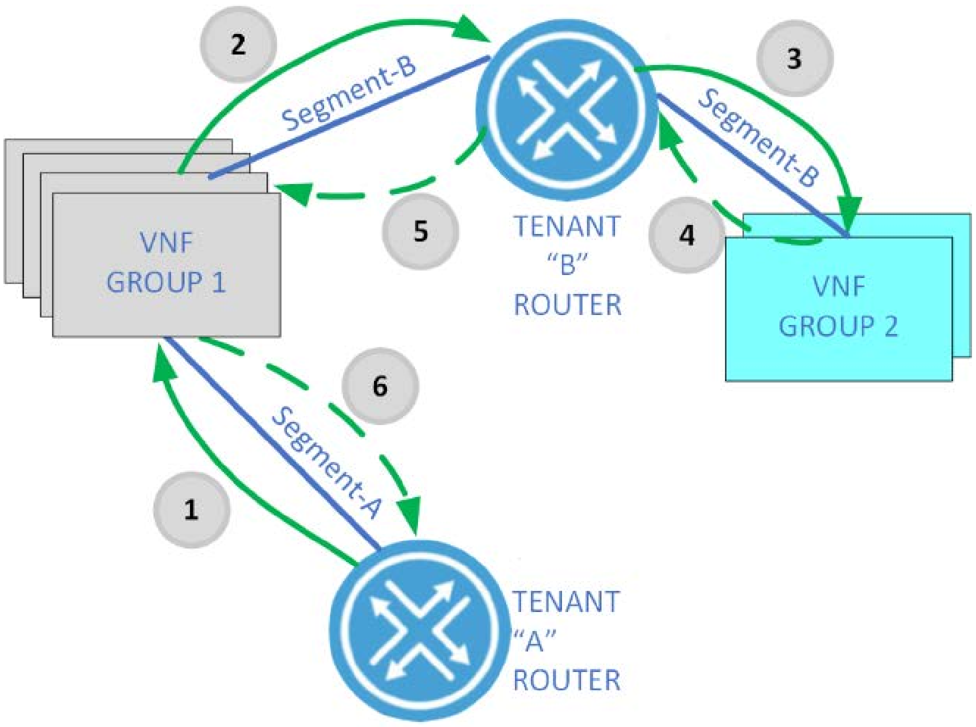


Figure 12-6 - Generic Network Function Chaining in BCF

12.4.1 Virtual Network Functions

In this paragraph, the VNFs used in the private datacenter are presented. These VNFs will be deployed in the BCF through the Service insertion policy mechanism specified above:

- Basic Firewall: provides Stateful L3/L4 functionalities, it is not application aware
- Application Aware Firewall: enables deep packet inspection and application awareness
- Load Balancer: balances workloads at the transport (TCP/UDP) and application level (for HTTP/HTTPS protocol).
- Load Balancer and Reverse proxy: provides Load balancing and advanced HTTP handling (terminates HTTP/s sessions, authenticates, validates queries, performs status monitoring, etc.)
- Virtual Tap: intercepts the network traffic and sends a copy of it to a specific target.

The VNFs specified above can be chained in different ways. For

12.4.2 Context Firewall

As specified in Par. 11.3, every tenant is connected to the rest of the network through a context firewall, that might or might not be application aware. In its simplest form the context firewall is just a L3/L4 firewall, but the actual number of VNFs involved could be 2 or more, depending on the actual service chaining internal to the firewall VNF itself: as an example, once the traffic hits the Firewall there could be a policy intercepting specific flows (per source, destination, protocol, application, etc.) and sending them to another, internal, VNF for further processing. And this process could be repeated.

In this case, essentially, BCF will trigger the first function of the NFC, and all the other functions would be chained by the firewall policies itself. It is worth noting that this principle might apply to any NFC, not just the firewall/security related.

12.4.2.1 Intra-Tenant Firewall

The intra-tenant firewall could be used to perform inter-segment routing for all or a subset of the specific segments terminated in the context. A typical use case could be, as an example, intercepting all the traffic from a VLAN containing hosts with a higher risk profile (i.e. PROD servers hosted on nearly end of support systems) and validate this traffic against stricter security policies enforced on the application-aware firewall.

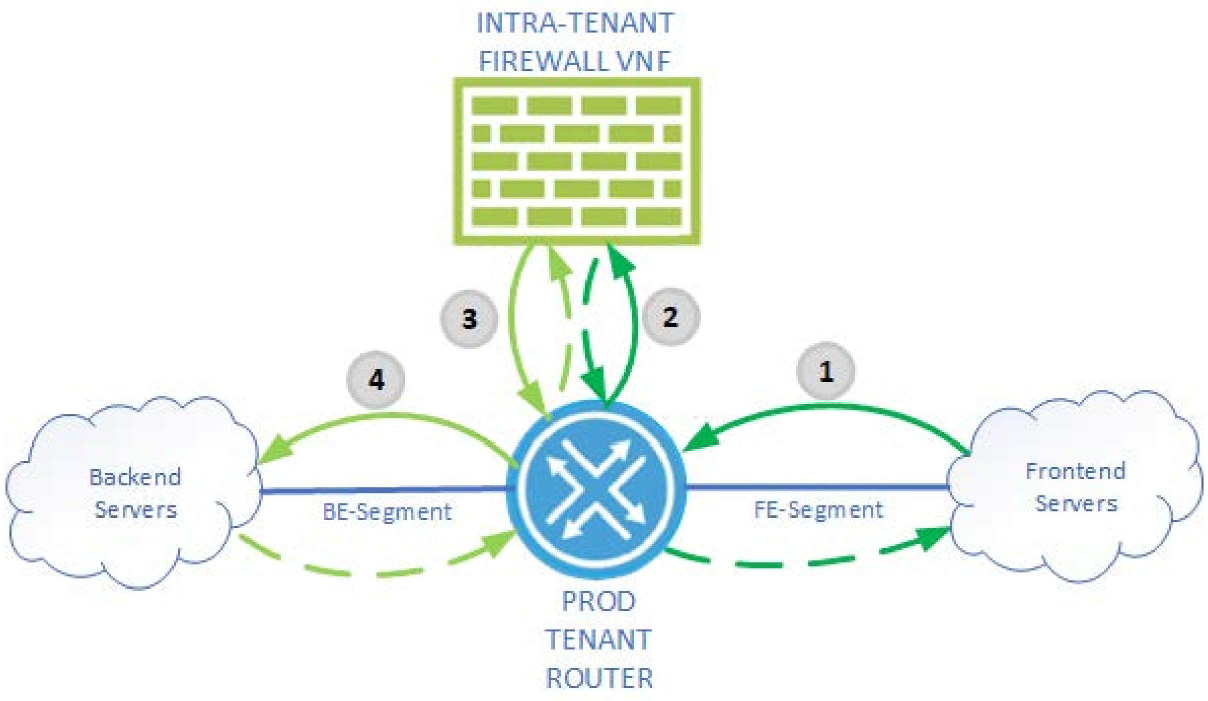


Figure 12-7 - Intra-Tenant Firewall Service Function Chain

12.4.2.2 Inter Tenant Firewall

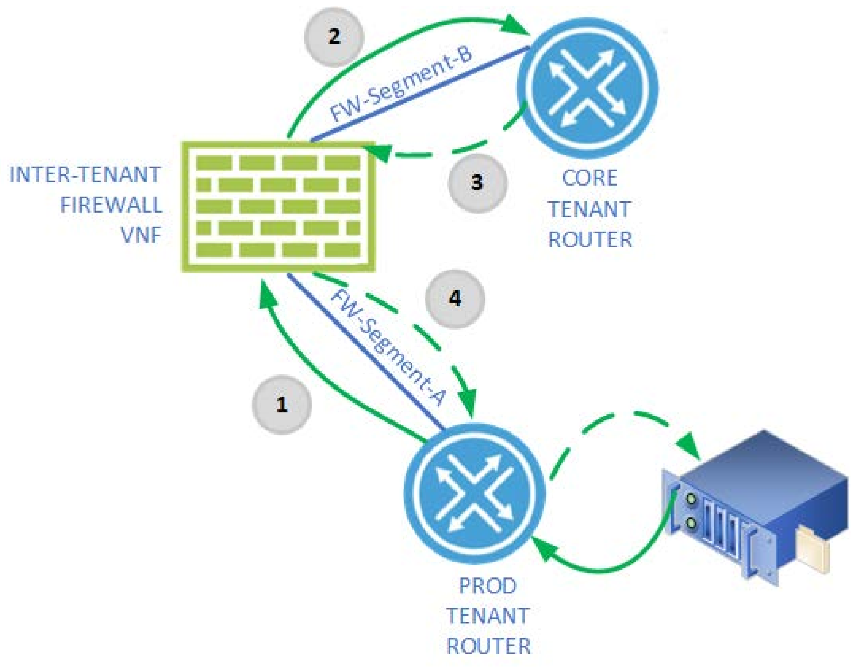


Figure 12-8 – Inter-Tenant Firewall Service Function Chain

12.4.3 Context Firewall + Load Balancing SFC

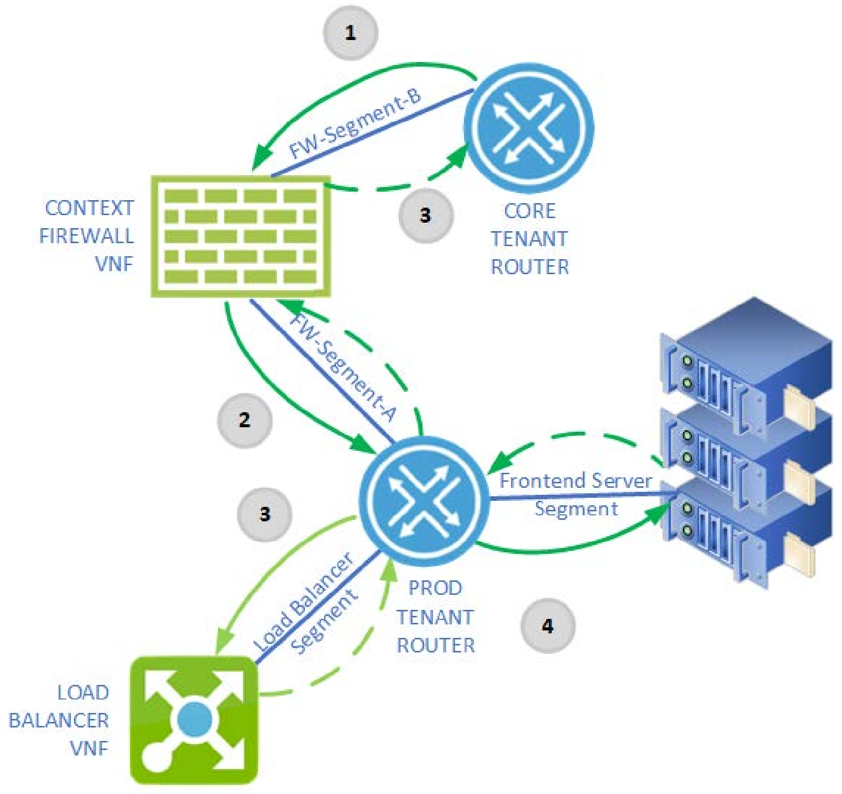


Figure 12-9 - Load Balancing + Firewall SFC

The load balancer could be deployed standalone, or as part of an advanced Firewall + Load Balancing SFC. The principle behind this approach would be that the deployment of a service cluster does not require any change on the firewall policy, only a change on the SFC in Tenant Router.

As an example, let's assume that a front-end server that was initially standalone needs to be deployed as, the flow would be

- Inbound traffic destined for the Frontend Server arrives to the CORE Tenant Router
- CORE Tenant router has an insertion policy directing all the traffic destined to the PROD networks to the PROD Context FW VNF
- PROD Firewall VNF has a rule specifying that if the target is the fronted server it should be allowed: PROD FW sends the traffic to the PROD Tenant Router.
- PROD Tenant Router has a Redirection policy which forwards all the traffic destined for the frontend server to the Load Balancer
- Load Balancer receives the inbound traffic and perform its duties accordingly.

12.5 Integration with VMWare vSphere

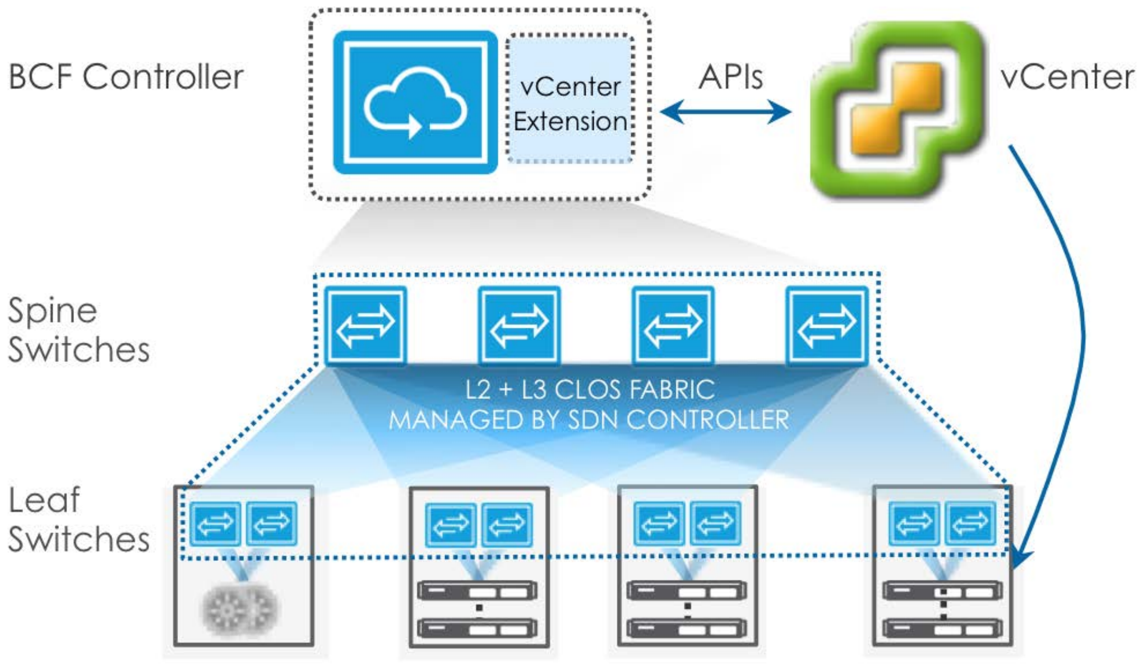


Figure 12-10 - Big Switch BCF integration with VMWare vSphere [102]

BCF can be integrated with VMware vCenter™ and supports application deployments on its physical SDN fabric — across both virtual and physical networks. The following aspects of the network are automated:

- Connectivity of the ESXi host
- Configuration of Layer 2 networks
- Provisioning on BCF of VM/VMkernel endpoint
- Migration of the network policies for vMotion

13 Overlay-based Design using Juniper Contrail

13.1 Abstract

This chapter presents the “*overlay based*” (see par. 7.6) high level design for the Enterprise (Primary and Secondary) private datacenters defined in Par. 11.3. This design is built with the Juniper Contrail architecture briefly introduced in paragraph 7.6.3 (for a more detailed introduction on this architecture see [103], [104], [105] and [106])

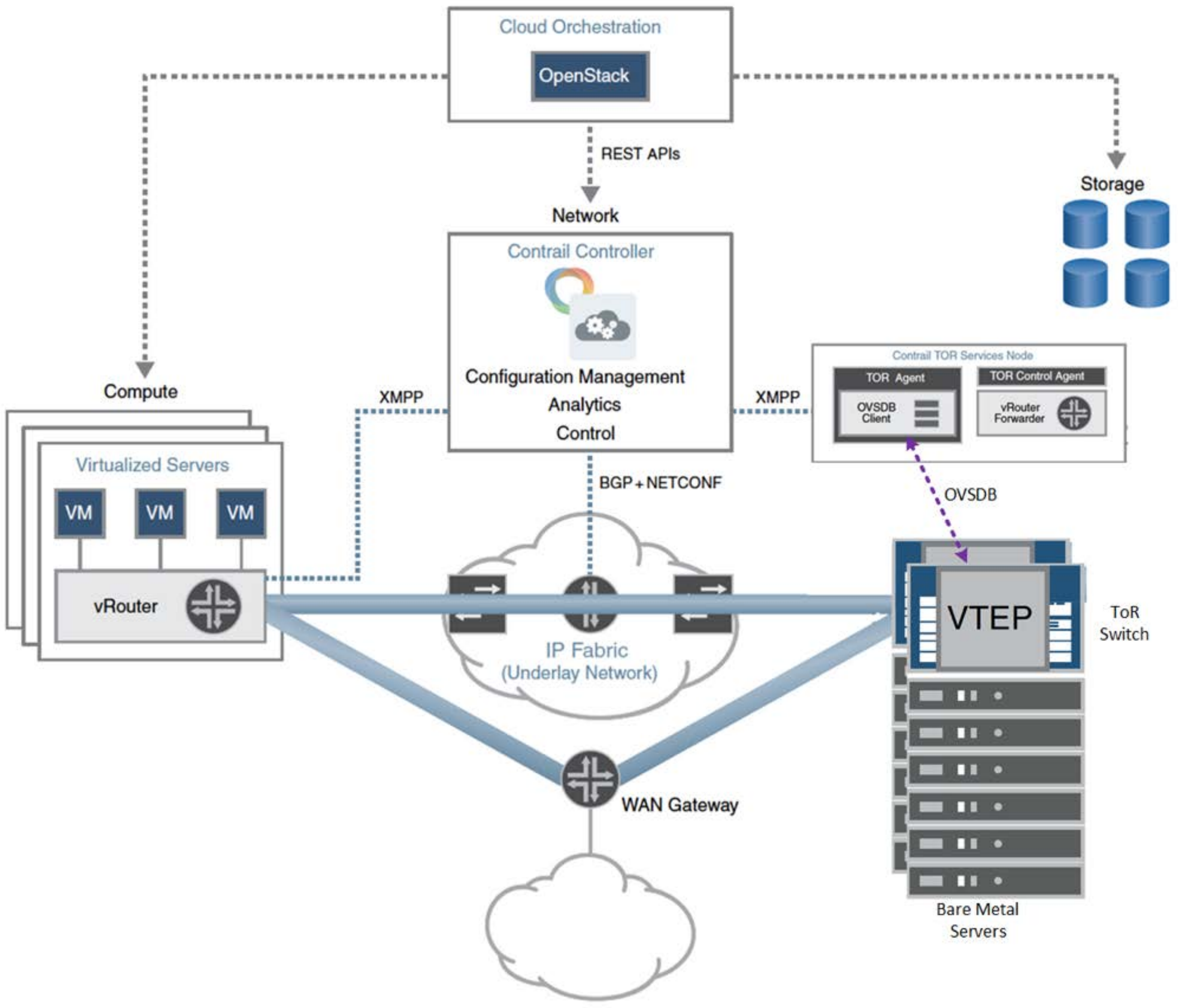


Figure 13-1 - Contrail Architecture

This chapter is divided in three sections

1. Design of the underlay DCN
2. Design of the overlay DCN
3. Network Function Chaining

13.2 DCN Underlay

In a VXLAN overlay implementation, the core requirement for the underlying physical network is that it must be a IP routed network. The underlay network must provide predictable performances and must scale linearly. In this HLD we will use a Clos (spine-leaf) architecture based on an IP Fabric built using Juniper Networks QFX5100 series switches, (currently) able to scale to a maximum of eight spine switches.

13.2.1 Physical topology

The proposed topology is a folded three-stage IP fabric using Juniper QFX5100-24S and QFX5100-96S. The QFX5100-24S is a 32 x 40GbE switch, and the QFX5100-96S is a 96 x 10GbE and 8 x 40GbE switch. An IP fabric of usable 3072 x 10GbE ports, as shown in Figure 13-2 can be created combining the QFX5100-24Q and the QFX5100-96S

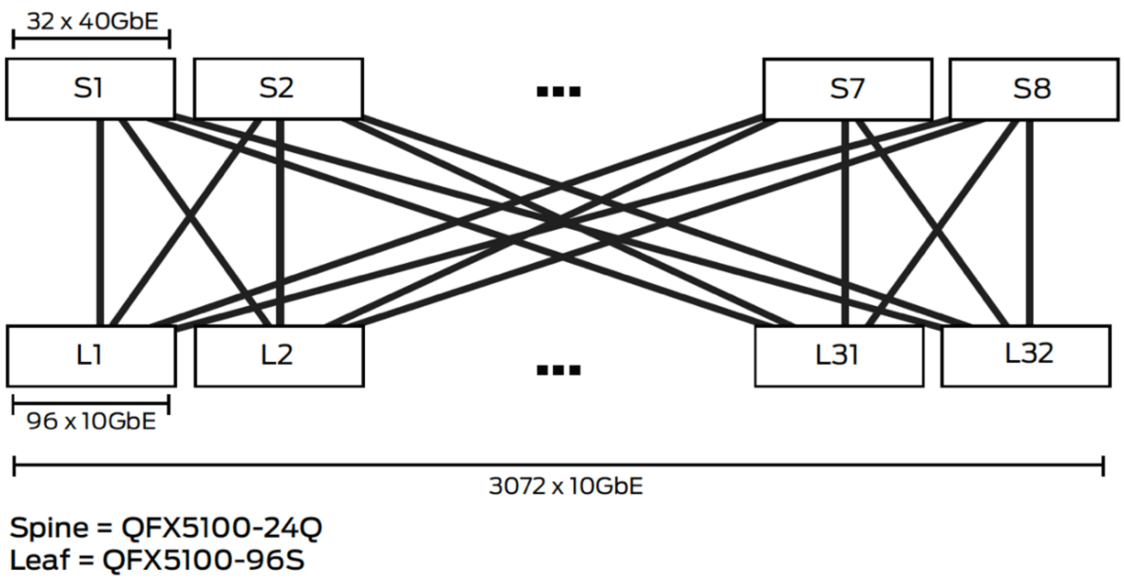


Figure 13-2 underlay DCN topology (Primary and Secondary DCs) using Juniper IP Fabric

The leaves are constructed using the QFX5100-96S, and their 8 x 40GbE interfaces are used as uplinks to the spine. Each leaf has 8 uplinks into the spine, so the maximum width of the spine is 8.

13.2.1.1 Oversubscription

In a leaf-spine topology oversubscription occurs at the leaf layer. Oversubscription is equal to the total amount of bandwidth available to all servers connected to a leaf switch divided by the amount of uplink bandwidth.

$$Oversubscription = \frac{total\ bandwidth}{uplink\ bandwidth} = \frac{96\ x\ 10Gbps}{8x40Gbps} = \frac{960}{320} = 3:1$$

13.2.1.2 Scalability and bi-sectional bandwidth

The presented solution supports up to 32 Leaf Switches (each spine switch has 32x40GbE ports), each with 96 10GbE Ports, which implies that a maximum of 96x32 = 3,072 active ports can be connected.

The total access bandwidth is 3,072 x 10Gbps=30,720Gbps=30.72Tbps. Each one of the 8 spine switches has 32 links @40GbE, which implies that the total bi-sectional bandwidth is 8x32x40Gbps = 10,240Gbps=10.24Tbps.

13.2.2 Routing Options for the underlay

The most common routing options for the control plane of an IP fabric are OSPF, IS-IS, and BGP. Every routing protocol can advertise prefixes, but the protocols vary in terms of features, convergence time, performances, scale, etc. OSPF and IS-IS use a flooding technique to send updates (link state packets) and other routing information. Creating areas can reduce the amount of flooding, but by doing this, one starts losing the benefits of an SPF routing protocol. In contrast, BGP, by design, supports many prefixes and peering points, as proven by the same existence of the Internet.

The ability to shift traffic around an IP fabric is a core feature. As an example, one could route traffic around a specific leaf when the switch is in maintenance. OSPF and IS-IS provide limited traffic manipulation abilities, and, again, BGP has been built to support extensive traffic steering and tagging.

Usually, a large IP fabric is built iteratively and over time, and it is common to see device from multiple vendors co-existing in a single IP fabric. Again, OSPF and IS-IS work well across multiple vendors, however, again, BGP is the best option when considering multi-vendor coexistence, as proven by the fact that Internet consists of a massive amount of equipment from different vendors, all using BGP.

So, to summarize, when selecting a control plane protocol for an IP fabric, due to its ability to scale up, tag traffic, and multivendor stability, BGP is the best option.

13.2.2.1 eBGP or iBGP

One of the first decisions to make is whether to use iBGP or eBGP. The ability to fully use all the available links, implies that the IP fabric must support equal cost multipath (ECMP). One of the key design point is how does each option support ECMP:

- eBGP handles ECMP without a problem.
- iBGP requires a BGP route reflector and the AddPath

If using eBGP in an IP fabric, each switch represents a different autonomous system number, and each leaf must peer with every other spine in the IP fabric, as shown in Figure 13-3

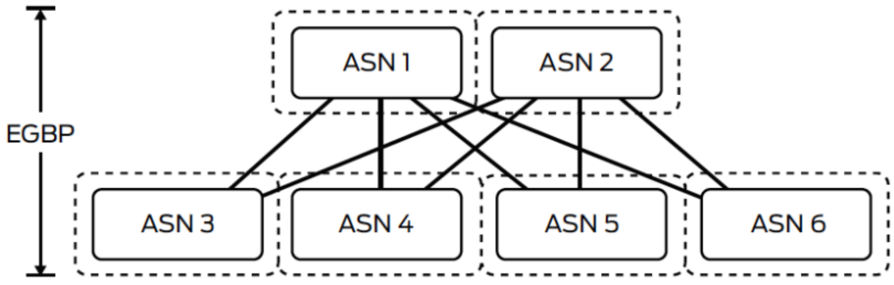


Figure 13-3 - Use of eBGP in an IP Fabric [106]

Using eBGP in an IP fabric is very simple and straightforward, and enabling the use of local preference and autonomous system padding also provides traffic capabilities.

The use of iBGP, though, is different since iBGP requires the peering between all switches. To mitigate impact of this *"full peering"*, the spines can be configured to support inline BGP route reflectors (RR), as illustrated in Figure 13-4. However, the issue with standard BGP route reflection is that it only reflects the best prefix and doesn't support well ECMP. To enable full ECMP, BGP AddPath feature must be used, which provides additional ECMP paths into the BGP advertisements between RR spines and leaves.

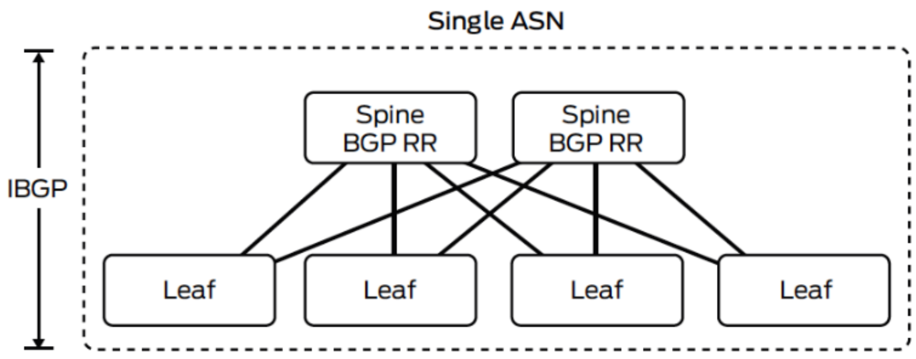


Figure 13-4 - Use of iBGP in an IP Fabric [106]

The QFX5100 Switches used in the underlay support both BGP design options of iBGP and eBGP. Both options work equally well. However, from the discussion above, it is quite clear that design and implementation using eBGP are simpler, therefore, eBGP is the routing protocol that will be used in the proposed underlay DCN.

13.2.2.2 eBGP Design

The main issue when using eBGP in the underlay is how many BGP autonomous system numbers will be consumed with the IP fabric. Each switch has its own BGP autonomous system number. Technically, the BGP private range is 64,512 to 65,535, which leaves with 1023 BGP autonomous system numbers. If the IP fabric is larger than 1023 switches, then the public BGP autonomous system number range must be considered or move to 32-bit autonomous system numbers.

13.2.2.3 Routing for the Primary and Secondary DCN underlay

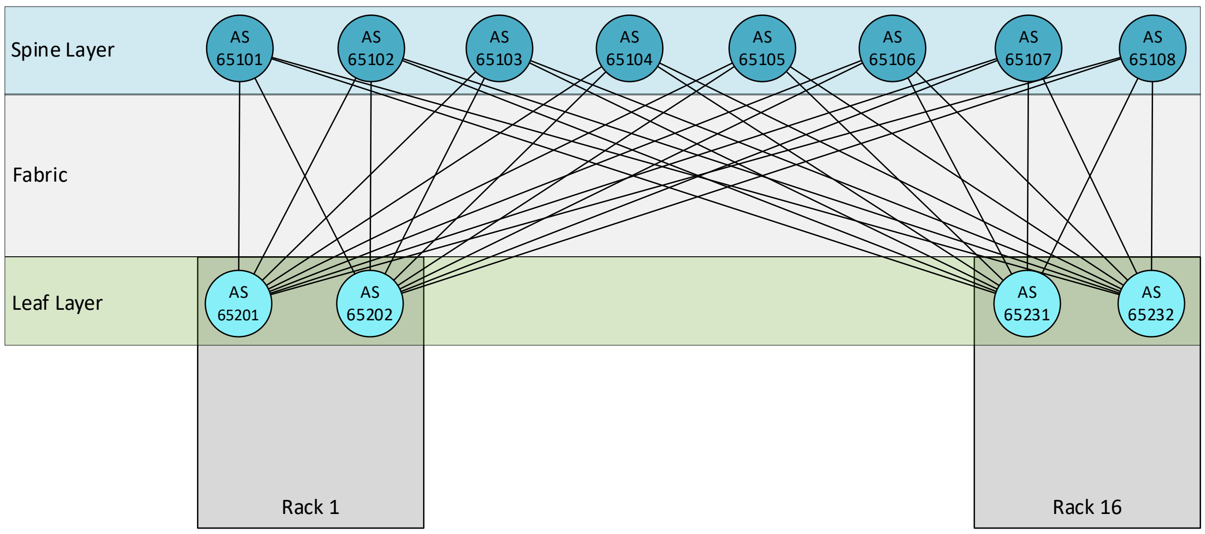


Figure 13-5 - Routing in the DCN underlay using eBGP and dedicated ASNs

For the reasons explained in 13.2.2, the routing protocol used in the DCN underlay will be eBGP with dedicated ASN number for every switch.

It is also a good practice to align the AS numbers within each layer. As shown in Figure 13-5 the spine layer will use AS numbering in the 651xx range, and the leaf layer will use AS numbering in the 652xx range.

13.3 DCN Overlay

As shown in Figure 13-1, the key components of DCN Overlay network are

- Virtual Network
- Network Policies
- Gateway Devices: vRouter, Gateway, etc.
- ToR switches and TSNs nodes, proxying virtual and physical workloads

13.3.1 Virtual network

Virtual networks are the domain where the workloads (Containers, VMs, physical nodes are connected). They are the equivalent of VLANs in classical networking.

13.3.2 Network policies

Network policies define how virtual networks are connected, and can be of two types:

- Virtual network policies: high level of abstraction, applied at the boundary of virtual networks

- Service policies: define how network services are connected and provided to the virtual networks (through NFC)

13.3.3 vRouter

Contrail implements overlay networking using a function called Contrail vRouter located in each hypervisor (e.g., KVM with QEMU), operating system supporting containers (e.g., Linux with Docker) or Contrail ToR Service Nodes. Contrail vRouters implement user-specified network policies issued by the Contrail Controller.

The Contrail Controller uses the Extensible Messaging and Presence Protocol (XMPP) to control the vRouters, and a combination of BGP and NETCONF protocols to communicate with physical devices (except TORs Service Nodes).

13.3.4 Integration with ToR Switches (using OVSDB)

Contrail networks can connect physical nodes via ToR switches running OVSDB [107]. Each physical switch has a set OVSDB tables storing the routes to host MAC addresses; OVSDB makes copies of the routes between switches via a Contrail element called the ToR Service Node (TSN).

The TSNs is essentially a proxy between XMPP and OVSDB on switches so that MAC routes can be exchanged between virtual and physical networks.

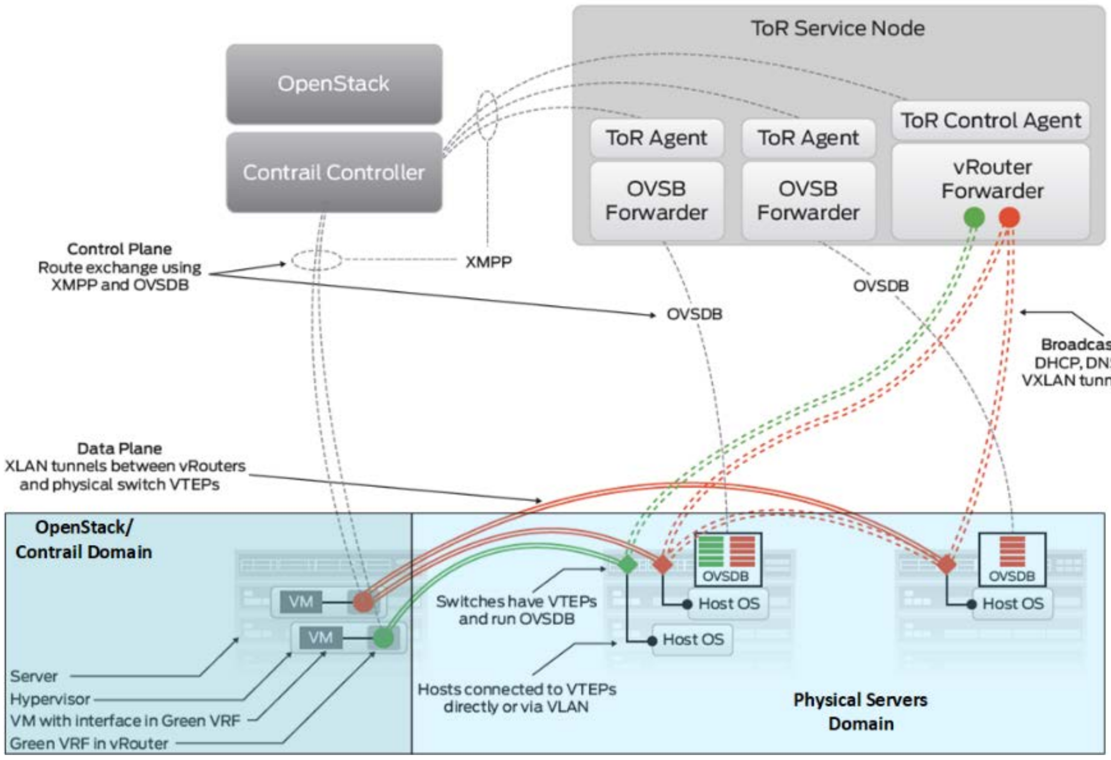

ToR Service Node
OpenStack
Contrail Controller
ToR Agent
ToR Agent
ToR Control Agent
OVSB Forwarder
OVSB Forwarder
vRouter Forwarder
Control Plane
Route exchange using XMPP and OVSDB
XMPP
OVSDB
OVSDB
Broadcast Traffic DHCP, DNS, ARP in VXLAN tunnels to TSN
Data Plane
XLAN tunnels between vRouters and physical switch VTEPs
OpenStack/ Contrail Domain
VM
VM
OVSDB
OVSDB
Host OS
Host OS
Host OS
Switches have VTEPs and run OVSDB
Hosts connected to VTEPs directly or via VLAN
Server
Hypervisor
VM with interface in Green VRF
Green VRF in vRouter
Physical Servers Domain

Figure 13-6 - Contrail ToR Service Node (TSN) [107]

A TSN contains four types of software components:

- OVSDB Client: send/receive Route updates and configuration changes
- ToR Agent: maintains an XMPP session with the Contrail Controller and mediates between the Contrail XMPP messages and OVSDB.
- vRouter Forwarder: traps and responds to broadcast packets that VTEPs in switches direct towards the IP address of a TSN inside VXLAN encapsulation.
- ToR Control Agent: a vRouter providing proxy services (DHCP, DNS, and ARP) for broadcast traffic arriving over VXLAN from servers attached to switches. Response data is provided by either the ToR Control Agent itself, or by the Contrail Controller via an XMPP session.

When a physical switch learns a new MAC address on an interface configured in a VTEP, it creates a bridge table entry for the MAC for that interface. A corresponding entry in the OVSDB table is created, causing a MAC route to be sent via OVSDB protocol to the TSN. The route specifies a VXLAN encapsulation tunnel where the next hop is the IP address of the switch, and the VNI will be that of the VTEP on the switch to which the server is connected. When a route arrives at the TSN, it is converted to an XMPP message that is sent to the Contrail Controller, which sends the route to vRouters that have VRFs with matching VNI. The TSN also sends the routes to other switches that are running OVSDB and have VTEPs with the same VNI.

Similarly, when VMs are created using an orchestrator platform like OpenStack or VMware vCenter/NSX, routes to the new VMs are sent by the Contrail Controller via the TSN to each switch with a VTEP with matching VNI. The routes specify VXLAN tunnels with the next hop being the IP address of the vRouter where the destination VM is running, and where the VNI value of the Contrail virtual network is being used.

When VXLAN traffic arrives at a vRouter, the VNI is used to identify which VRF should be used for MAC lookup to find the virtual interface to which the inner Ethernet frame should be sent.

13.4 Logical view of the Enterprise private DCN using Contrail

13.4.1 Primary Datacenter

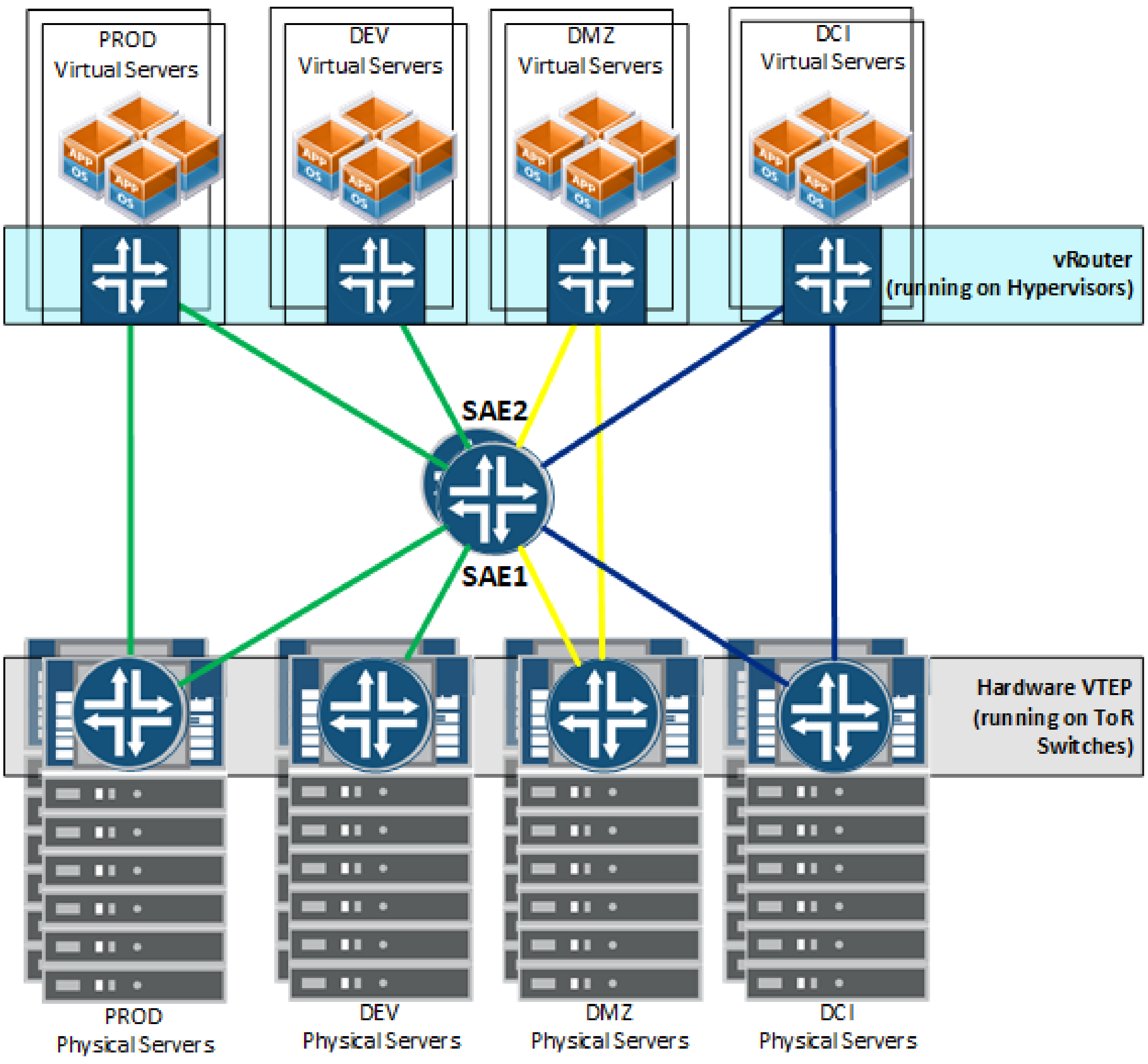


Figure 13-7 - Primary DCN with Contrail – Overlay Topology Logical View

13.4.1.1 Intra-tenant routing

In every tenant, the corresponding vRouter (and Hardware VTEPs running on ToR switches in case of workload deployed on legacy or physical servers) performs Intra-tenant routing (aka inter-VLAN routing). A service insertion policy could be deployed in the tenant router to intercept inter-VLAN traffic and send it to a NFC (that could be virtual, physical or hybrid, see 13.5.3) performing, for instance, Firewall or load balancing functions.

13.4.1.2 Inter-tenant routing

The inter-tenant routing will be performed through a hybrid NFC containing, amongst the others, also a Juniper (Virtual) SRX Firewall

13.4.2 Secondary Datacenter

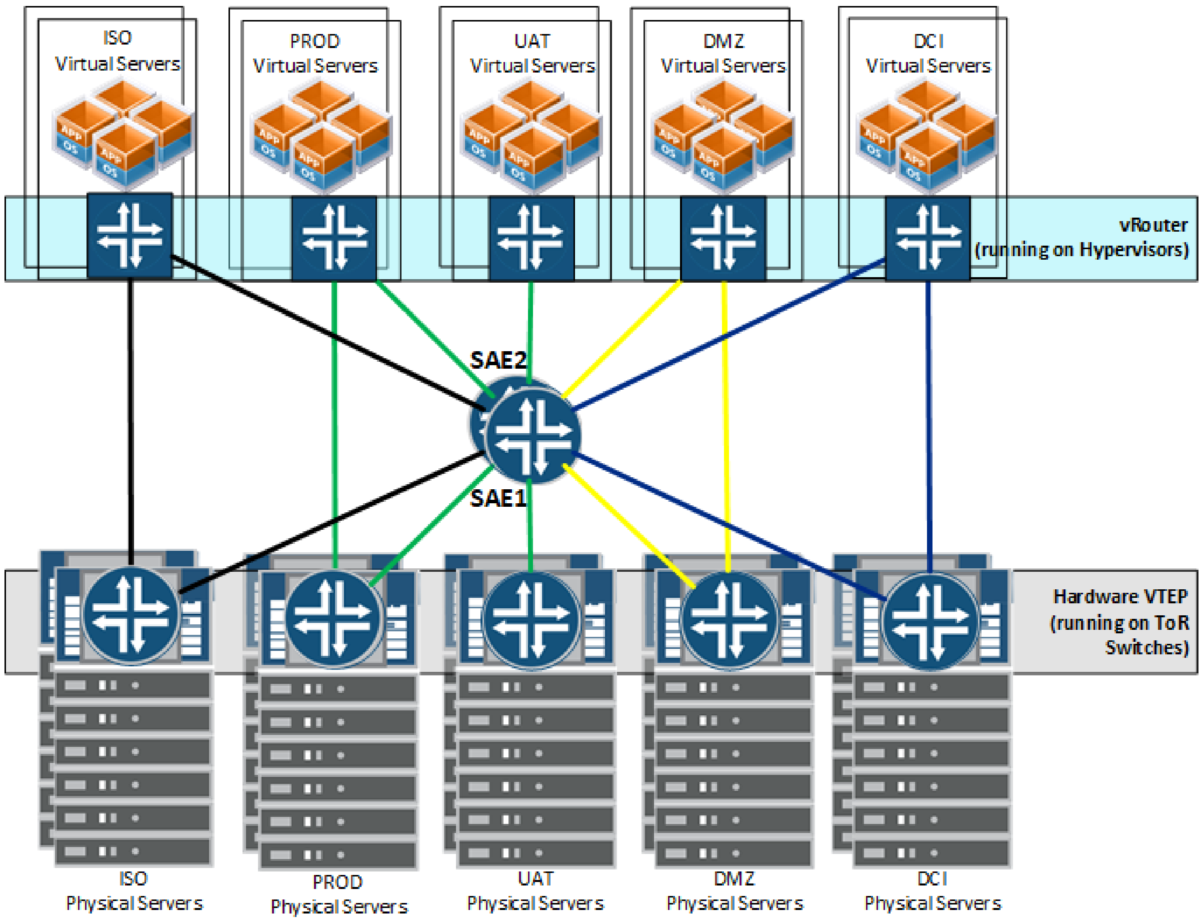


Figure 13-8 - Secondary DCN with Contrail – Overlay Topology Logical View

13.4.2.1 Intra-tenant routing

In every tenant, the corresponding vRouter (and Hardware VTEPs running on ToR switches in case of workload deployed on legacy or physical servers) performs Intra-tenant routing (aka inter-VLAN routing). A service insertion policy could be deployed in the tenant router to intercept inter-VLAN traffic and send it to a NFC (that could be virtual, physical or hybrid, see 13.5.3) performing, for instance, Firewall or load balancing functions.

13.4.2.2 Inter-tenant routing

The inter-tenant routing will be performed through a hybrid NFC containing, amongst the others, also a Juniper SRX Firewall

13.5 Network Function Chaining

Service chaining can be offered in two ways

- Virtual Network Function (VNF) Chaining: dynamic chains of virtual services running on virtual machines
- Physical Network Functions (PNF) Chaining: chains physical appliance-based services
- Hybrid Network Functions Chaining: chains that include a combination of VNFs and PNFs.

NFC using Juniper Networks contrail is described in [108].

13.5.1 NFC Modes

Contrail Release 3.0 and greater supports also the creation of service chains that include a combination of VNFs and PNFs.

Services can be configured in the following modes:

- Transparent or bridge mode: this mode is used for services that do not modify the packet. Also known as bump-in-the-wire or Layer 2 mode. Examples include Layer 2 firewall, IDP, and so on.
- In-network or routed mode: Provides a gateway service where packets are routed between the service instance interfaces. Examples include NAT, Layer 3 firewall, load balancer, HTTP proxy, and so on.
- In-network-nat mode: Like in-network mode, however, return traffic does not need to be routed to the source network. In-network-nat mode is particularly useful for NAT service.

13.5.2 NFC Elements

Service chaining requires the following configuration elements in the solution:

- Service template: an object describing the characteristics of all the service instances, their domain, etc.
- Service instance: an object describing the concrete" implementation object" of a template
- Service policy: an object describing the service insertion rules

13.5.3 Types of Service Chaining

13.5.3.1 VNF Chaining (V-NFC)

Services are offered by instantiating service virtual machines to dynamically apply single or multiple services to virtual machine (VM) traffic. Figure 13-9 shows the basic service chain, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

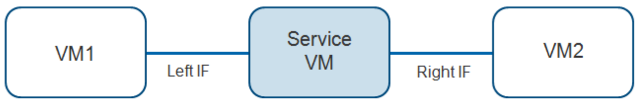


Figure 13-9 – Contrail basic V-NFC

When a virtual service chain is created, Contrail software creates tunnels on the underlay network that span through all services in the chain. Figure 13-10 shows two end points and two compute nodes, each with one service instance and traffic going to and from one endpoint to the other

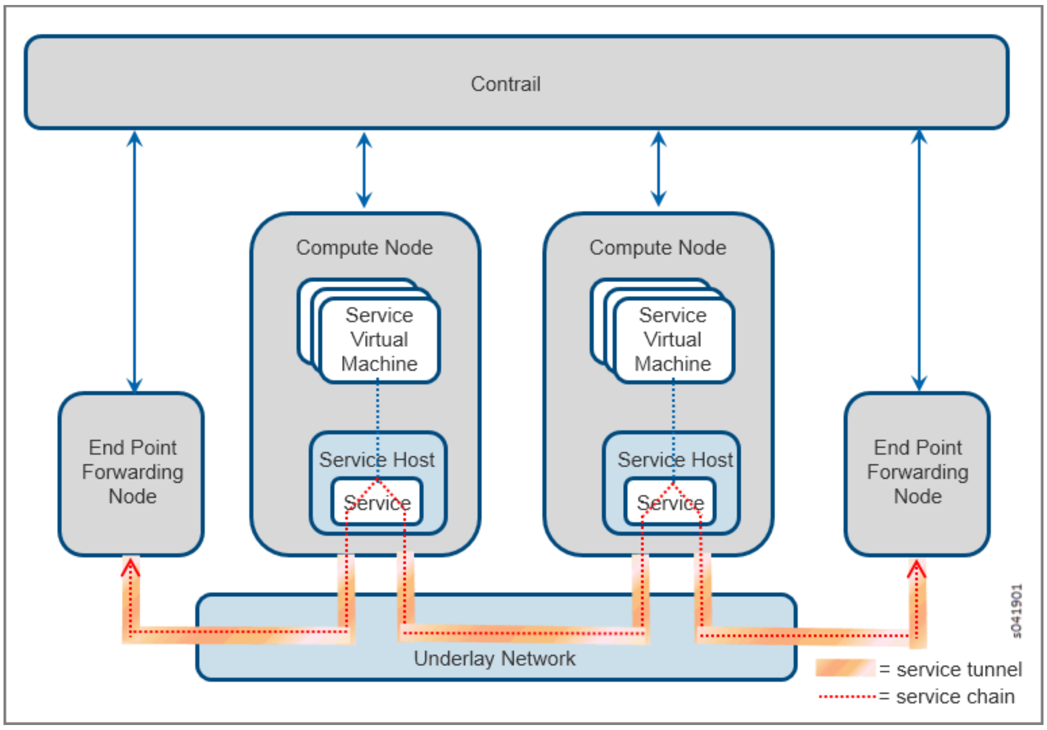


Figure 13-10 – Contrail V-NFC

13.5.3.2 PNF Chaining (P-NFC)

Contrail Release 3.0 support PNFs in service chains, including:

- service appliance (SA)—represents a single physical appliance
- service appliance set (SA set)—represents a collection of functionally equivalent SAs, all running the same software with the same capabilities

A service appliance is associated with a physical router that has physical interfaces for the left, right, management, or other interfaces. There can be more than one service appliance and associated physical router and physical interface objects representing it.

A physical appliance can host more than one service appliance through a logical system or other virtualization capability.

The service template object supports a physical network function service template (PNF-ST). The PNF-ST is associated with a service appliance set, which represents a pool of service appliances that can be used when the PNF-ST is instantiated. Only the transparent service mode (see 13.5.1) is supported for PNF-STs.

To implement a p-NFC with Contrail the following conditions must be met:

- Before the controller can use a PNF SA, the controller must be connected to a service control gateway (SCG) router, such as an MX Series router; the Contrail Device Manager must manage the SCG router.
- The PNF SA must be configured and must operate as an Ethernet bridge. The Contrail controller does not automatically implement PNF SA configuration.
- Infrastructure interfaces (physical interfaces or aggregated Ethernet interfaces) on the SCG facing the SA must be preconfigured. The interfaces must be able to support VLAN-based units.

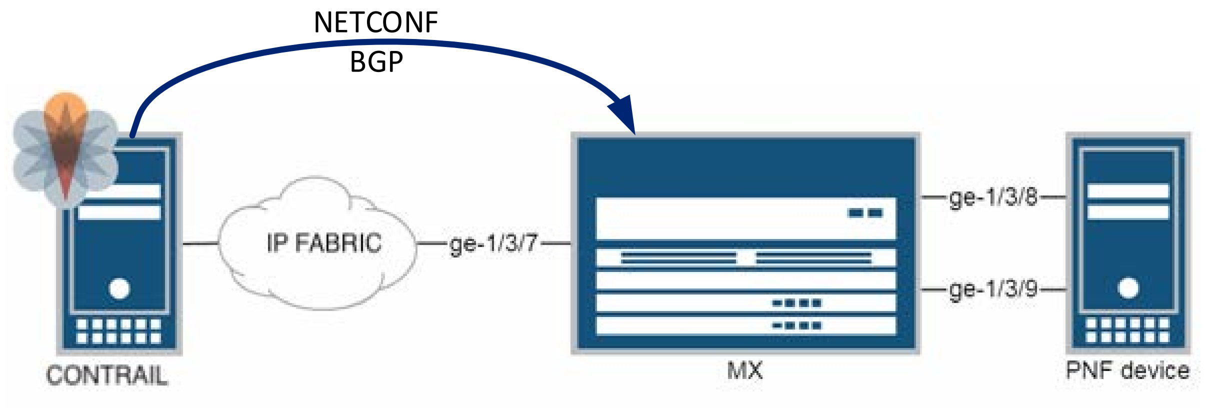


Figure 13-11 – Contrail P-NFC

13.5.3.3 Hybrid NFC (H-NFC)

VNFs and PNFs can be combined in a hybrid service chain connecting services provided on a virtual environment and network services provided through physical appliances.

13.6 Integration with VMWare vSphere

The vCenter integrated Contrail solution has the following modes [109]:

- vCenter-only
- vCenter-as-compute

13.6.1 vCenter-only mode

In the vCenter-only mode, vCenter is the main orchestrator, and Contrail is integrated with vCenter for the virtual networking.

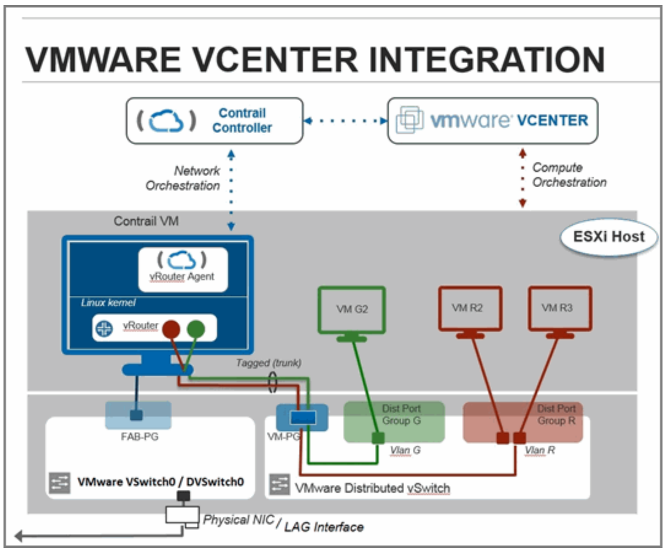


Figure 13-12 Contrail – vCenter-only Mode [109]

13.6.2 vCenter-as-Compute Mode

In the vCenter-as-compute mode, OpenStack is the main orchestrator, and the vCenter cluster, along with the managed ESXi hosts, act as a nova-compute node to the OpenStack orchestrator.

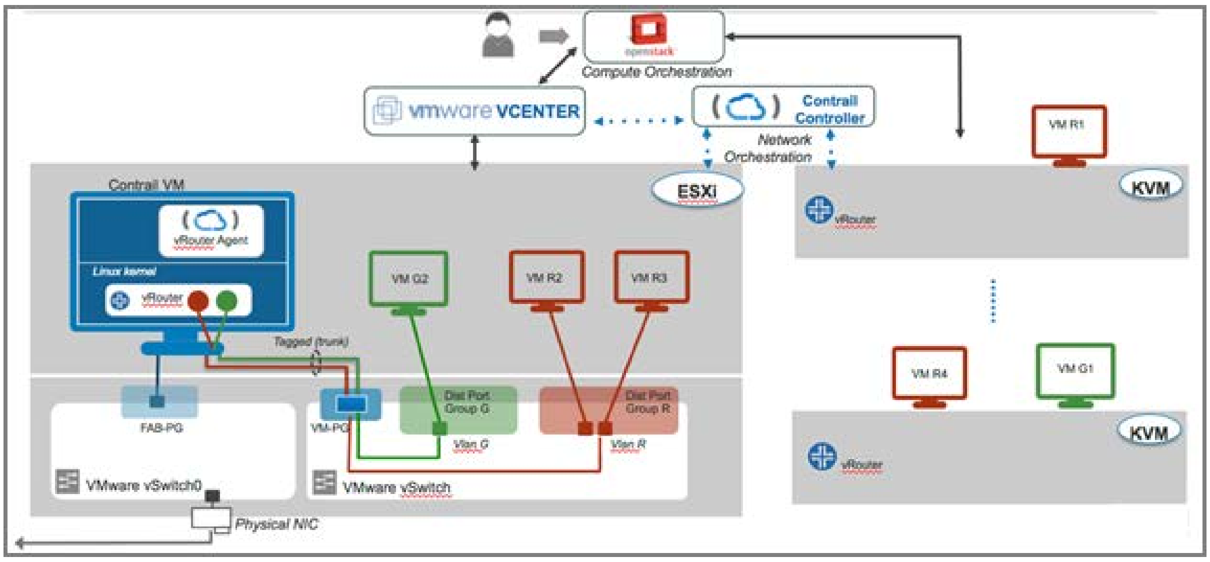


Figure 13-13 Contrail – vCenter-as-Compute Mode [109]

14 Intent-based design using Cisco ACI

14.1 Abstract

This paragraph presents the SDN policy-based high-level design for enterprise private datacenter using Cisco ACI Architecture (see Par. 7.7.4) which relies on Cisco APIC Controller for the SDN Control plane and Cisco Nexus switches for the network plane. An introduction on APIC, ACI and Intent based networking can be found in [85], [110] and [111]

14.2 Key components

14.2.1 Cisco ACI Operating System

To develop Cisco ACI OS, Cisco has taken the Nexus OS (NX-OS), developed for the Datacenter Networks, and trimmed to the features essential for a modern datacenter. Cisco made also profound structural changes so that the Cisco ACI Fabric OS can render policies defined in the APIC into the physical infrastructure:

- a Data Management Engine (DME) provides the framework that handles the I/O requests to a shared, object oriented, lockless data store. Each object stored as portions of data, with each portion owned by one ACI process. Any ACI process can read any data but only the process owner of the portion can write on it. Simultaneous, concurrent access to the data simultaneously is provided through API, CLI, or SNMP calls.
- A local policy element (PE) implement the policy model directly in the ACI Fabric OS, as illustrated in Figure 14-1

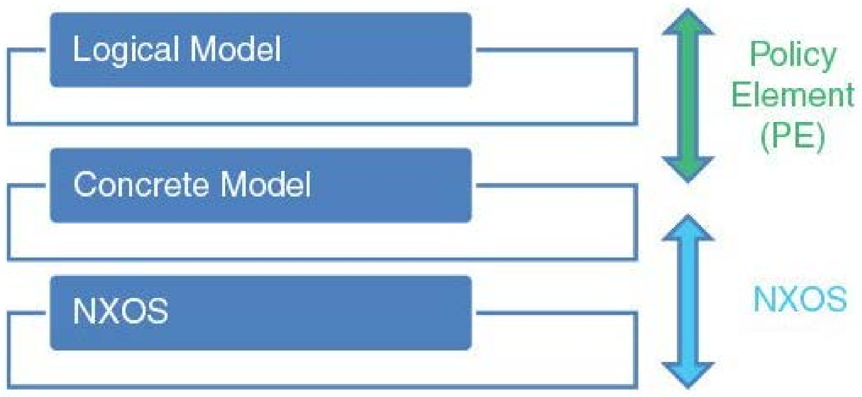


Figure 14-1 - ACI OS and NX-OS

14.2.2 Physical Topology

The mandatory DCN topology adopted by Cisco ACI is a spine-leaf providing zero-touch provisioning, auto-discovery, and an integrated cable plan. The Cisco ACI topology is made by a set of leaf switches connected to a set of spine switches in a full bipartite graph using 40/100-Gigabit Ethernet links. All leaf switches are connected to all spine switches, all spine switches are connected to all leaf switches, and links between spines switches or between leaf switches are disabled if present.

Leaf switches connect any network device or host and enforce the network policies. Leaf switches are also able to route and bridge external networks: in this case they are referred to as border leaf switches.

14.2.2.1 APIC Controller Connectivity

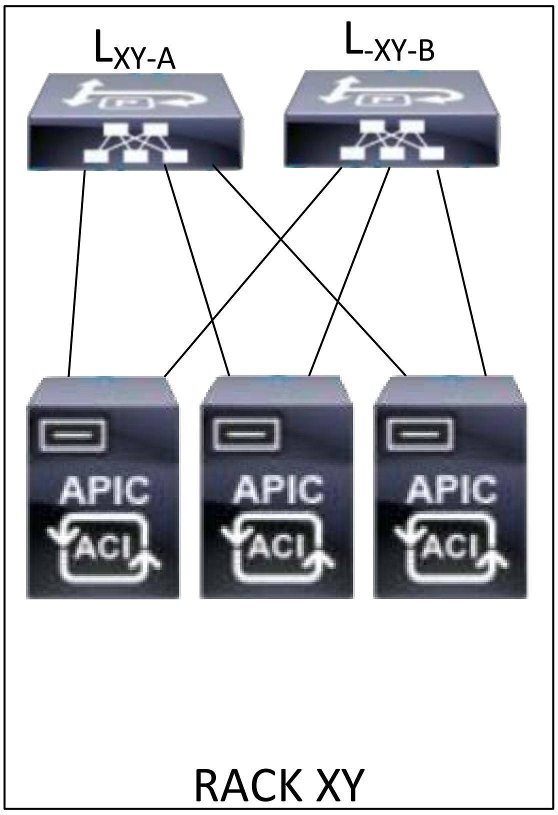


Figure 14-2 – APIC Controller Connectivity

A typical ACI infrastructure requires 3 APIC controllers, due the fact that all the components of ACI are datasets generated and processed by the Distributed Policy Repository and that data for those APICs functions are partitioned into logically banded subsets called shards (like DB shard). A Shard is then broken into three copies, one for each APIC, but only one APIC is the master for a specific copy/shard.

This strategy evenly balances work load and processing across the cluster, also providing a failsafe in case an APIC goes down. If one of the three APICs goes down, the remaining two will negotiate who will now be the master for the shards that the down APIC oversaw. The workload will be then load balanced between the two and the cluster becomes fully operational again.

Working with 2 APICs is not recommended due to the risk of split-brain that occurs when both APIC 1 and APIC 2 assume to be master of a shard and cannot agree so the shard is in contention and the cluster is unfit ("data layer partially diverged").

In case of only 1 APIC, that APIC does all the work, it is the leader for all shards but if it goes down no changes can be made at all: data plane will continue forwarding but due to the absence of the APIC, there will be no way to create new policies or changes.

14.2.2.2 External connectivity (Border Leaf)

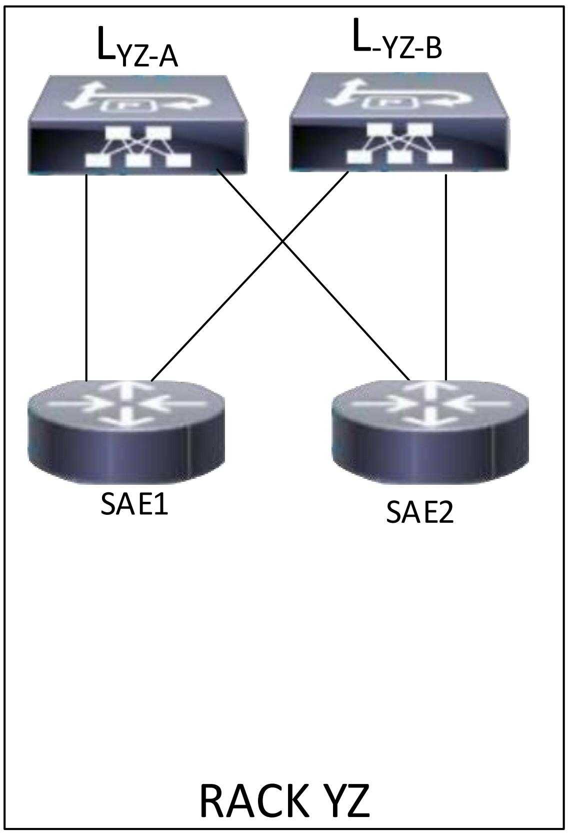


Figure 14-3 - Border leaf and external connectivity

From the physical standpoint, external connectivity is provided to the ACI through internetworking devices (i.e. routers, firewalls) connected to a pair of leaf switches. As shown in Figure 14-3, in the Enterprise Private datacenters the connectivity to the SAE1 and SAE2 could be provided through two gateways connected to two border leaf switches.

At the routing/logical level, Cisco ACI refers to external Layer 3 connectivity as a L3Out connection: in a standard configuration, route peering and static routing are performed on a per-VRF basis. External prefixes are learned on a per tenant and per-VRF and are redistributed in the forwarding tables of the leaf nodes only if the specific VRF is deployed on that leaf

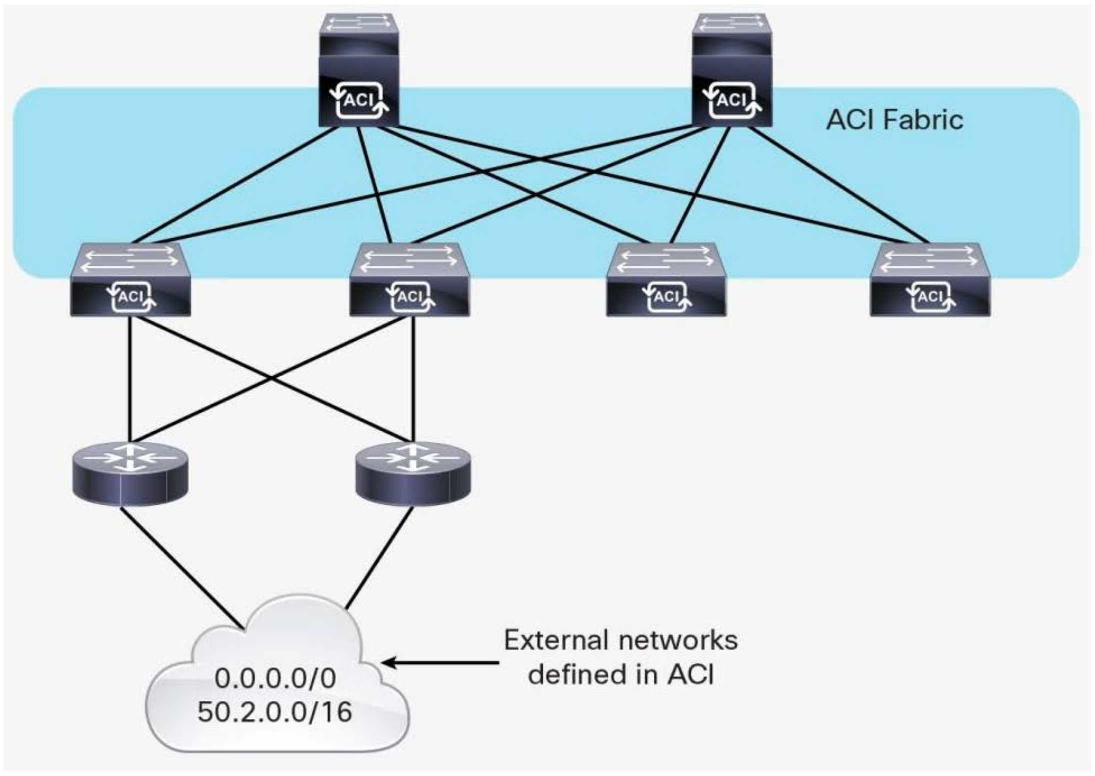


Figure 14-4 - External routing on ACI

14.2.2.3 Border Leaf Switch Design Consideration

Any Cisco ACI leaf switch can be a border leaf, and there is no limitation on the number of leaf switches that can be used as border leaf switches[17]. The border leaf can also be used to connect to computing, IP storage, and service appliances. In large-scale design scenarios, for greater scalability, it may be beneficial to separate border leaf switches from the leaf switches that connect to computing and service appliances.

Border leaf switches support three types of interfaces to connect to an external router:

- Layer 3 (routed) interface.
- Sub-interface with IEEE 802.1Q tagging: With this option, multiple sub-interfaces can be configured on the main physical interface, each with its own VLAN identifier.
- Switched virtual interface (SVI): the same physical interface can be used for Layer 2 connections (called *L2out*) as well as an Layer 3 connection (called *L3out*).

Besides providing support to routing protocols, and exchanging routes with external routers, the border leaf switches can also enforce traffic policies between internal and external endpoints.

[17] The only limitation is physical: in a bipartite Clos Networks, the spine ports must be enough to connect all the leaf switches.

14.2.3 The policy Object model

From a high-level perspective, the Cisco APIC policy model could be identified as a policy enforcement engine abstracting the underlay network functionalities, focused on the application. The Cisco APIC policy model is an object-oriented model based on the promise theory (see par. 7.7.4.1). Promise theory relies on an ordered hierarchy of objects handling configuration changes originated by the APIC controller and triggering exceptions and faults back to the controller, when and if needed. It should be noted that this approach reduces the load and complexity of the control plane and allows for greater scale, but it also moves back the control plane, or at least a part of it, to the underlying network devices[18].

14.2.3.1 Tenants

At the top level, the Cisco APIC policy model defines the Tenant class providing segregation capabilities for both network administration and traffic[19]. These tenants can be used by customers, business units, groups, or management teams, depending on the requirements.

As an example, an enterprise might use one tenant for the entire organization, while a cloud provider might have customers using one or more tenants to represent their organization. A tenant is a logical container or a folder for application policies. It can represent an actual tenant, an organization, or a domain, or can just be used for the convenience of organizing information. A normal tenant represents a unit of isolation from a policy perspective, but it does not represent a private network.

A special tenant named *common* has sharable policies that can be used by all tenants.

14.2.3.2 Contexts

Tenants further break down into private Layer 3 networks, called contexts, directly related to Virtual Route Forwarding (VRF) instances or separate IP space.

Each tenant may have one or more private contexts depending on the needs. IP addressing, routes, etc., can be duplicated in different contexts for multitenancy thanks due to the isolation of the forwarding instances provided by the contexts. A context is a unit of routing and policy isolation in the Cisco ACI framework and can be declared within a or in the mentioned "*common*" tenant.

[18] The more abstract the policies triggered by the controller, the more "intelligent" must be the underlay device implementing them.
[19] Multi-tenancy in ACI is enforced at the packet level (see Par. 7.6.2)

This approach provides both multiple private Layer 3 networks per tenant and shared Layer 3 networks that can be used by multiple tenants (for the contexts defines in the common tenant). The context object also provides a segregation at the policy level: the endpoint policy specifies a common Cisco ACI behavior for all endpoints defined within a given virtual ACI context.

14.2.3.3 Endpoint groups

Inside the context, the Object model provides a class that defines the applications: the objects are in these class are called endpoint groups (EPG) and represent a collection of similar endpoints representing an application tier or set of services. Policies interconnect the EPGs and include a collection of inbound/ outbound filters, traffic quality settings, marking rules/ redirection rules, and Layers 4– 7 service graphs. The relationship between EPG and Policy is shown in Figure 14-5: in this environment there are two contexts inside a tenant, each context containing a series of applications defined as set of EPGs and policies.

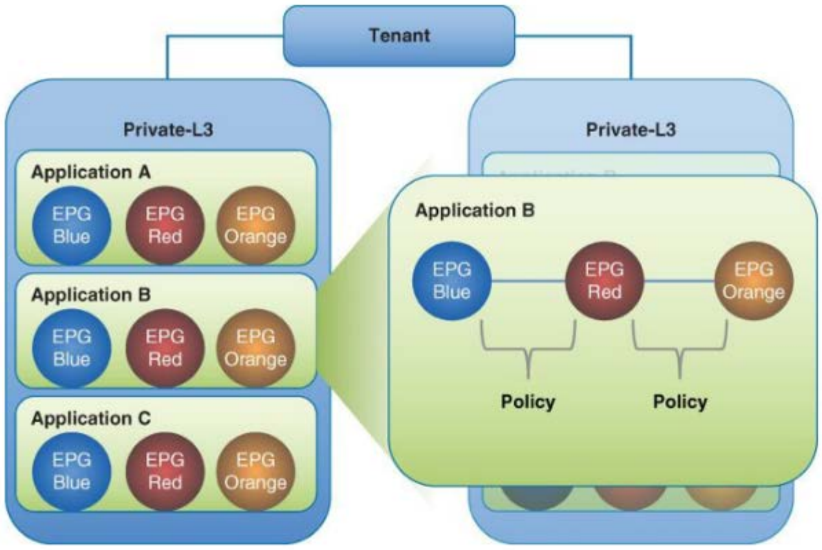


Figure 14-5 - Cisco APIC Logical Object Model [85]

14.2.3.4 Contracts

Contracts are ordered set of actions (i.e. permit, deny, QoS, redirect, service graphs) that can be applied both inbound and outbound to EPGs. Contracts are the ACI equivalent of ACLs and define the rules based on which a given EPG can communicate with other EPGs. Cisco ACI, as any intent-based infrastructure based on promise theory, explicitly separates the "*what*" and "*where*" of policy application allowing the creation of a policy independently from its application or reuse. The policy configured in the fabric is based on the high-level policy (intent) defined as a contract (*the what*) and the intersection of EPGs and other contracts with those policies (*the where*).

14.2.4 The southbound protocol: OPFlex

OpFlex [112] is the southbound protocol used to propagate policies and other information such as endpoint reachability between the APIC Controller and the leaf switches.

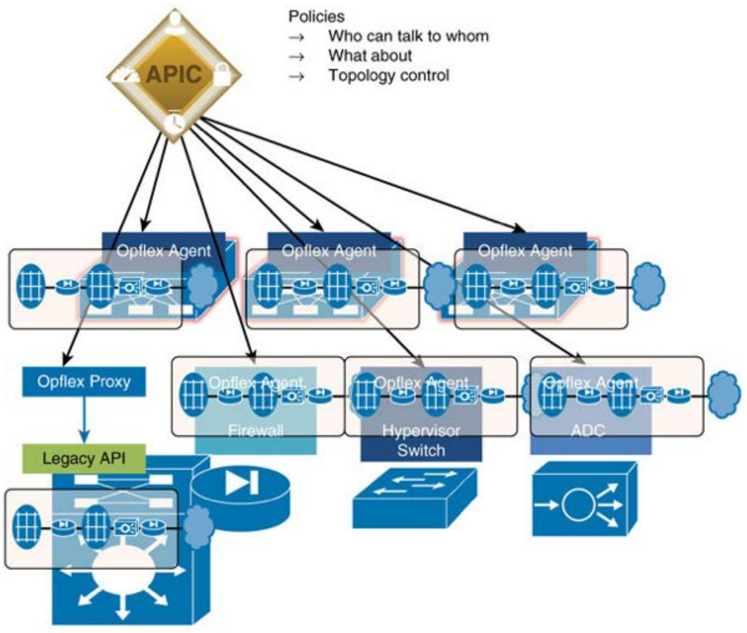


Figure 14-6 - Cisco ACI Propagates Policies to All the Leaf Devices [85]

Through OPFlex, the APIC controller propagates the policies to the leaf switches[20], which then render the policies based on to their local capabilities, as shown in Figure 14-7

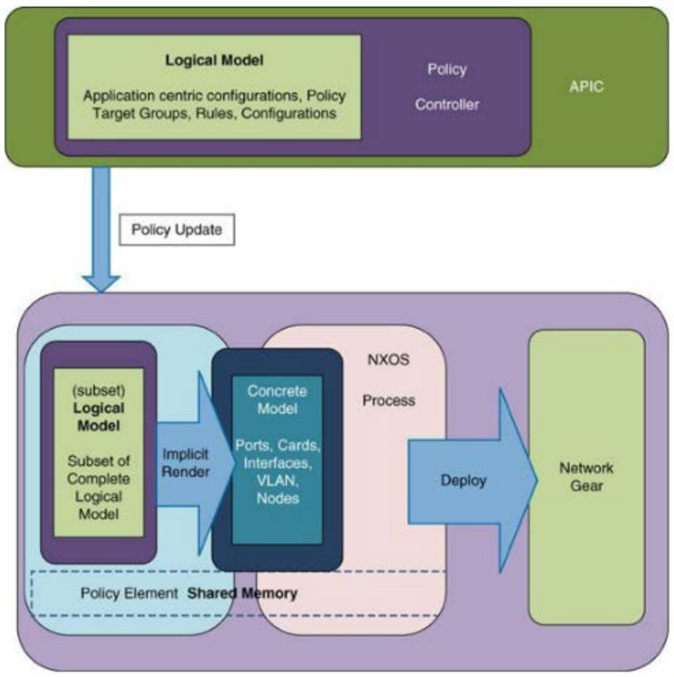

Logical Model
Application centric configurations, Policy Target Groups, Rules, Configurations
Policy
Controller
APIC
Policy Update
(subset) Logical Model
Subset of Complete Logical Model
Implicit Render
Concrete Model
Ports, Cards, Interfaces, VLAN, Nodes
NXOS
Process
Deploy
Network Gear
Policy Element Shared Memory

Figure 14-7 - Policy Rendering by the Network Devices [85]

[20] In Cisco ACI only the leaf switches render the policies, the spines do not implement any policy

14.2.5 ACI Routing and Pervasive Gateway

ACI fabric uses the idea of a pervasive gateway, which is an Anycast gateway. The subnet default gateway addresses are programmed in all leaves with endpoints present for the specific tenant subnet.

ACI uses multiprotocol BGP (MP-BGP) [56] [113] between leaf and spine switches to propagate external routes. All the leaf and spine switches are in the same Autonomous System (AS): when a border leaf learns external prefixes, it can then redistribute them to a MP-BGP address family (VPN4 or VPN6). The border leaf advertises through MP-BGP the prefixes to one of the spine switches (which is also one of its BGP neighbors), configured as a BGP route reflector (RR). The receiving RR then propagates the learned prefixes to all the leaves where the VRFs (or contexts in APIC terminology) are instantiated.

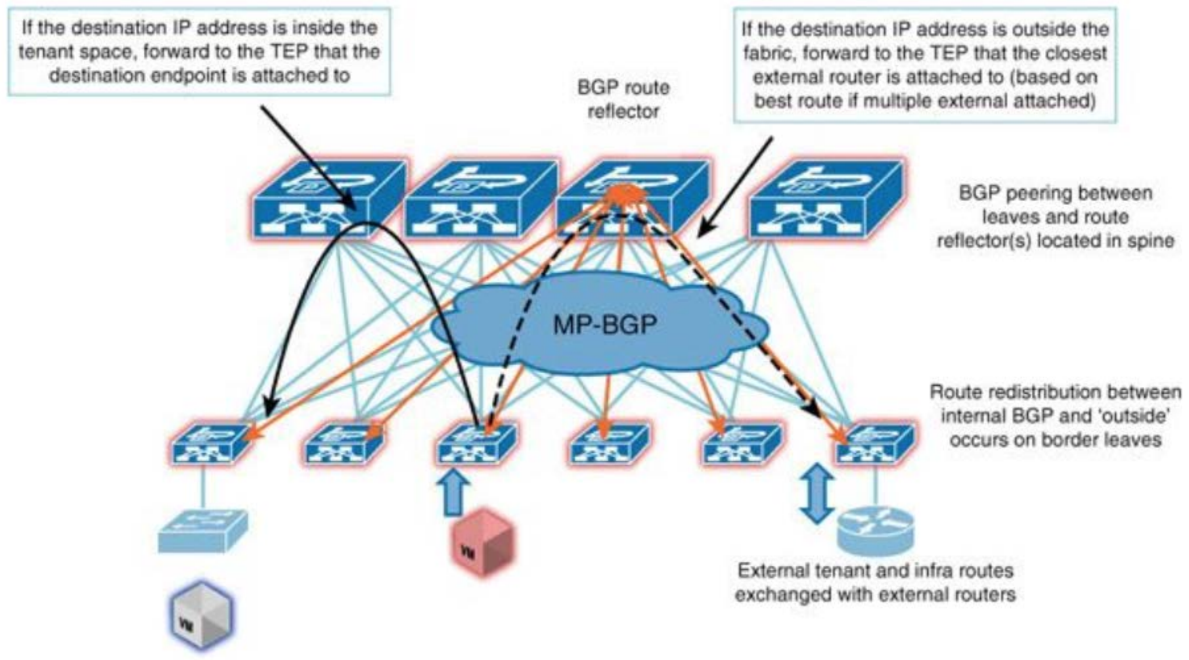

If the destination IP address is inside the tenant space, forward to the TEP that the destination endpoint is attached to
BGP route reflector
If the destination IP address is outside the fabric, forward to the TEP that the closest external router is attached to (based on best route if multiple external attached)
BGP peering between leaves and route reflector(s) located in spine
MP-BGP
Route redistribution between internal BGP and 'outside' occurs on border leaves
External tenant and infra routes exchanged with external routers

Figure 14-8 - MP-BGP routing in ACI for External Networks [85]

14.2.6 ACI Forwarding

Forwarding in ACI is based on VXLAN encapsulation, with some customization applied to the original VXLAN protocol:

- standard VXLAN uses multicast in the transport network to emulate Layer 2 flooding for BUM traffic. Unlike traditional VXLAN networks, the ACI preferred mode of operations does not rely on multicast for learning and discovery but on a mapping database that is populated upon discovery of endpoints in a way that resembles the LISP Protocol [114]

- The ACI VXLAN header is an extension of the LISP protocol (to which adds policy groups, load and path metrics, counters and ingress ports, and encapsulation information) and it can identify properties in the frames forwarded through the fabric.
- The VXLAN header is not connected to specific segments of the network and provides a general-purpose overlay used in the ACI fabric.

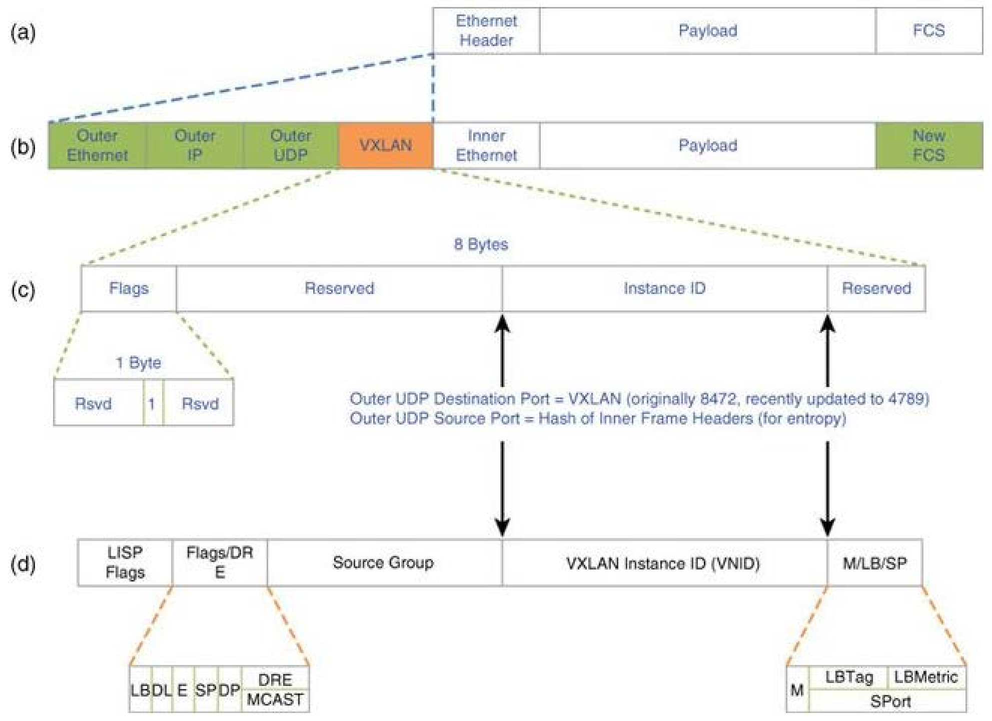


Figure 14-9 - ACI VXLAN format

The ACI fabric separates the endpoint address, its "*identifier*," from the location of that endpoint, which is defined by its "*locator*", or VTEP address. The ACI uses the VXLAN policy header shown in Figure 14-9 and, as shown in Figure 14-10, the fabric forwards packet between VTEPs. The mapping of the internal tenant MAC or IP address to location is performed by the VTEP using a distributed mapping database.

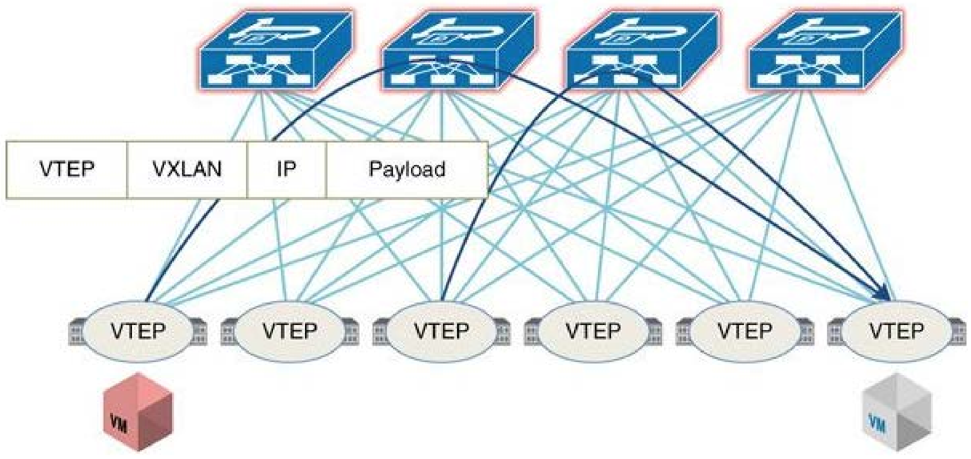


Figure 14-10 - ACI Forwarding [85]

Traffic from all the hosts attached to leaves can be tagged with network virtualization generic routing encapsulation (NVGRE, see [115]), VXLAN, or VLAN headers and then at the edge is normalized to the ACI VXLAN (see Figure 14-11)

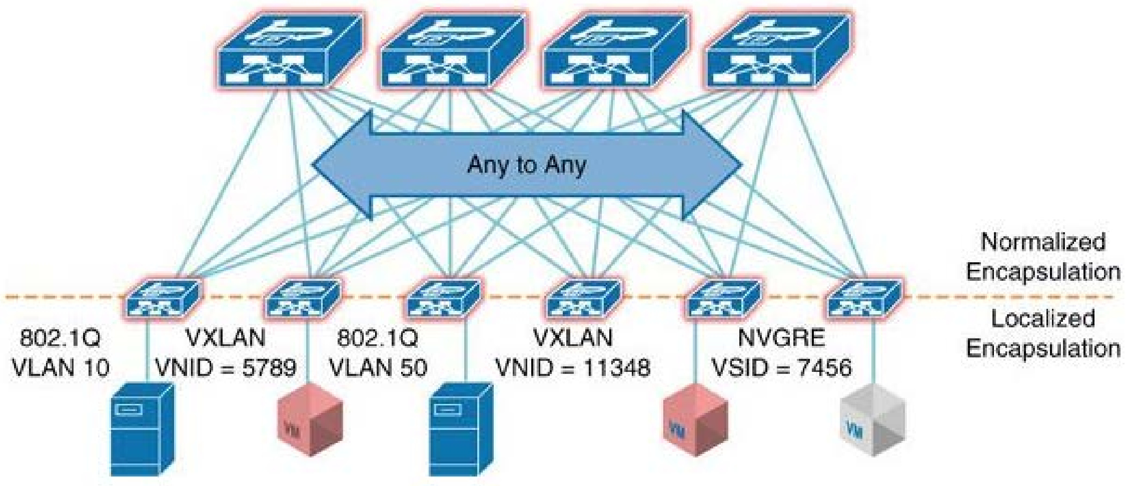


Figure 14-11 - Encapsulation normalization in ACI

14.2.7 Service Function Chaining

14.2.7.1 Service insertion

As explained in par. 14.2.3.4, a contract connects two or more EPGs, but it also offers Service insertion capabilities, i.e. the ability to insert in the path L2-L7 network functions such as traffic filtering, traffic load balancing, and SSL offloading (see Figure 14-12). Cisco ACI can locate the devices that provide these functions and insert them into the path as defined by the service graph policy (for details on SFC with ACI see [116])

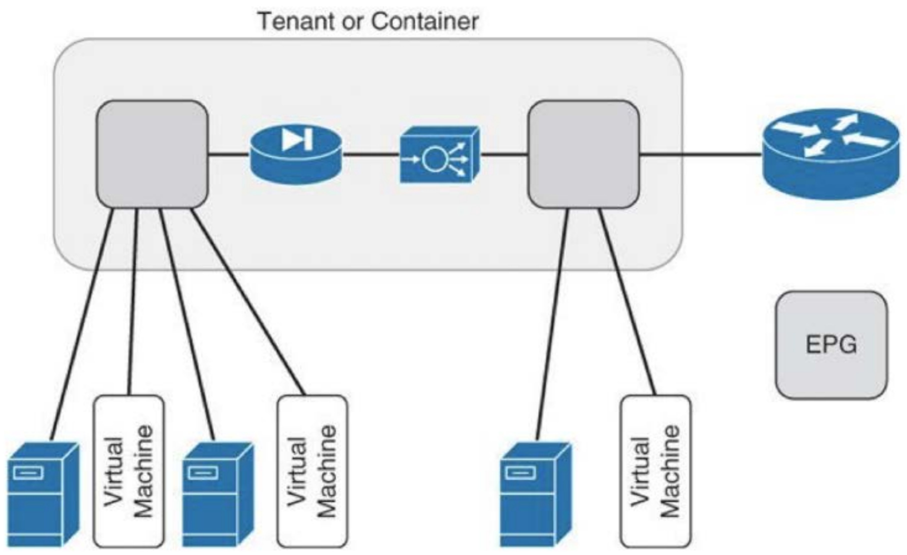


Figure 14-12 – ACI Service Insertion (Firewall and load balancer) [116]

Virtual appliances can be automatically inserted into the Cisco ACI fabric by the Cisco Application Policy Infrastructure Controller (APIC).

14.2.7.2 Service graph

The concept of *service graph* is more general than service insertion. A service graph is the Cisco ACI term for SFC: a concatenation of functions. The service graph identifies the path from one EPG to another EPG through certain VNFs or PNFs. The Cisco APIC translates (*renders)* the definition of the service graph into a path through network functions (i.e. firewalls and load balancers).

As shown in Figure 14-13, the Cisco APIC is aware of the availability of resources connected to the Fabric (i.e. load balancers and firewalls) and can use them to translate the user intentions defined in the service graph.

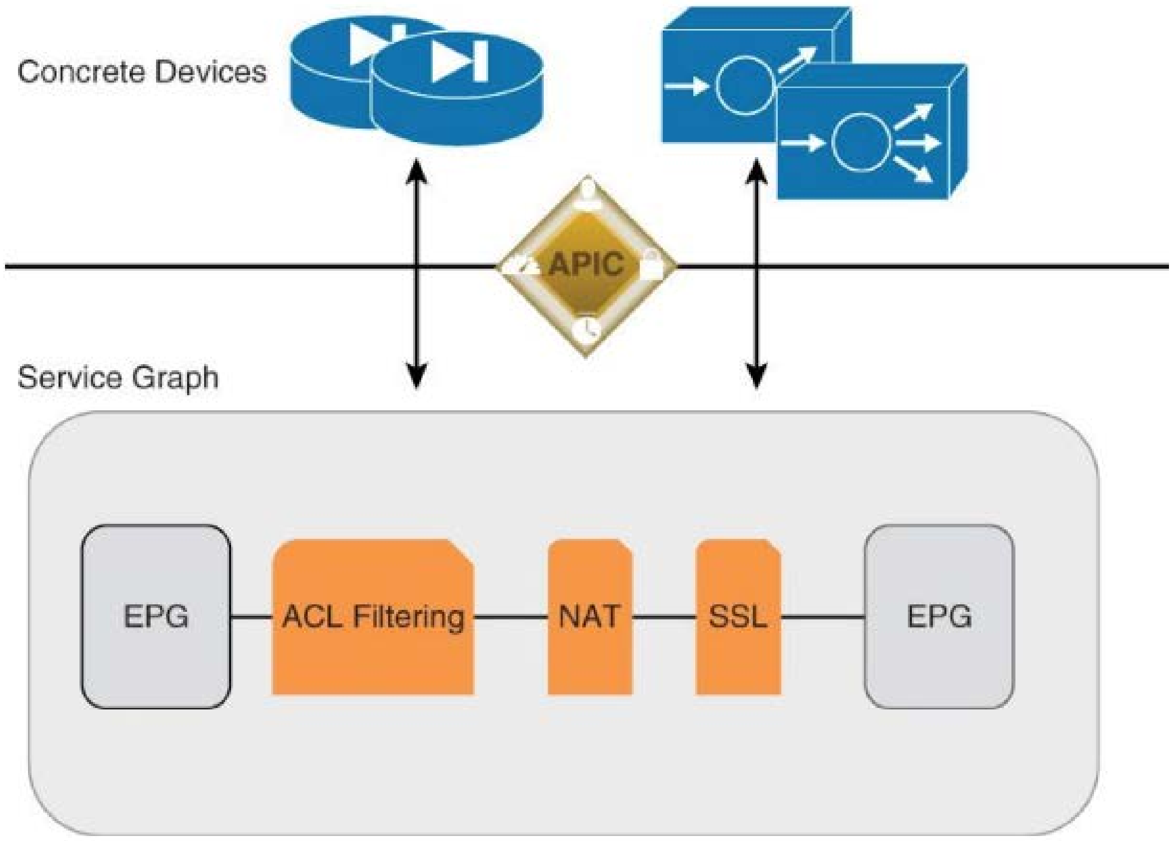


Figure 14-13 – ACI Service graph concept [116]

The service graph is a sort of template that can be used in different datacenters and *rendered* with the resources available on site.

The APIC Controller communicates with the VNFs or PNFs to render the service graph defined by the user. To communicate with the network appliances (physical or virtual) and connect them through the service chain designed in the service graph, Cisco APIC needs to use a so-called "*device APIs*": a capability provided by a plug-in (called device package and provided by the appliance vendor) that must be installed by The APIC Administrator.

14.3 DCN Physical Topology

14.3.1 Design options

The high-level design is based on a choice of the Cisco Nexus 9396 leaf switch and three different design options for the spines:

- Design option “A”: Cisco Nexus 9336: with 36x40G ports,
- Design option “B”: Cisco Nexus 9508, able to provide up to 288x40G ports (line cards with 36 ports each)
- Design option “C”: Cisco Nexus 9516, able to provide up to 576x40G ports (16-line cards with 36 ports each).

As explained in chapter 5, there is an immediate correlation between the type of spine and the potential number non-blocking leaf ports shown in the formula displayed in Figure 14-14. The interconnect between spine and leaf here uses a 40-GE speed. It is expected to have 40-GE at the leaf port facing level and 100-GE in the spine leaf interconnect in the future.

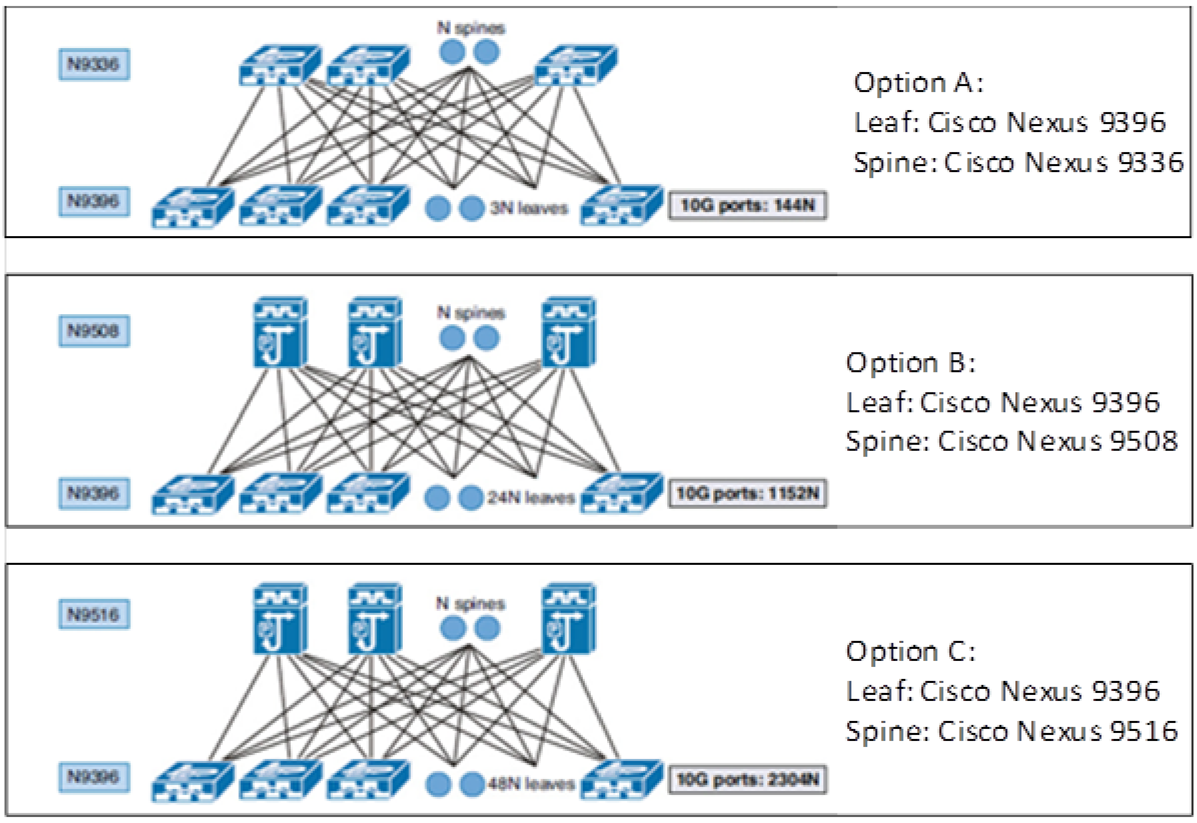


Figure 14-14 – ACI DCN Design Options with 40-GE Interconnect

14.3.2 Oversubscription

The presented design uses as leaf a cisco nexus 9396 providing 48 10G ports and 12 40G ports, and it assumes that all the 12 spine uplinks on the leaf switches are connected to (at least two[21]) different non-blocking spine switches.

$$Oversubscription = \frac{total\ bandwidth}{uplink\ bandwidth} = \frac{48\ x\ 10Gbps}{12\ x40Gbps} = \frac{480}{480} = 1:1$$

The oversubscription ratio is 1:1, so there is no oversubscription and the topology is non-blocking.

14.3.3 Scalability and bi-sectional bandwidth

If we define

- N_s as the number of spine switches
- P_s as the number of ports per spine switch
- P_u as the number of uplink ports per leaf switch
- P_a as the number of access ports per leaf switch

In a maximum occupation scenario, all the spine ports are connected to all the uplinks ports in the leaf switches and the oversubscription is always 1:1, so if $N_{l,s}$ is the maximum number of leaf switches per spine, we have $N_{l,s} = \frac{P_s}{P_u}$

If we define $P_{a,s}$ as the maximum number of access ports that can be connected to a single spine, we have that $P_{a,s} = P_a \cdot N_{l,s}$ therefore the total number of 10G access ports $P_{a,tot}$ is

$$P_{a,tot} = N_s \cdot P_{a,s} = N_s \cdot P_a \cdot \frac{P_s}{P_u}$$

Table 14-1 shows the scalability (in terms of $P_{a,tot}$) of the proposed design for the three spine models and for different number of spines. The table considers that the leaf switch is a Cisco Nexus 9396 providing $P_a = 48$ and $P_u = 12$, and that $2 \le N_s \le 12$ (spines must be redundant, and all the leaf switches must be connected to all the spines)

Design Option	Spine Switch	P_s	$N_{l,s}$	$P_{a,s}$	$P_{a,tot}$			
					$N_s = 2$	$N_s = 3$	$N_s = 6$	$N_s = 12$
A	Nexus 9336	36	3	144	288	432	864	1,728
B	Nexus 9508	288	24	1,152	2,304	**3,456**	6,912	13,824
C	Nexus 9516	576	48	2,304	4,608	6,912	13,824	27,648

Table 14-1- ACI DCN scalability

[21] Spine chassis must be redundant to provide resiliency at the spine layer

The optimal configuration for the requested size (3,072 access ports) is Option "*B*" with 3 spine switches. This solution supports the requested number of hosts and the architecture can scale from 2,304 (using 2 spines) up to 13.824 10G access ports (using 12 spines)

The bi-sectional bandwidth in a non-blocking network is the same of the access bandwidth, therefore the bi-sectional bandwidth in the proposed design is 3,456 x 10Gbps = 34,560Gbps = 34.56Tbps

14.3.4 DCN Design

The private datacenters must be able to support up to 1,536 dual homed servers with 10G access, and the optimal Design option is "B" with 3 spines. This architecture scales up to a total amount of 3,456 Access ports, 34.56Tbps of bisectional bandwidth and up to 3,456/48=72 Access switches, i.e. 36 pairs of Leaf ToR Switches (see Figure 14-15)

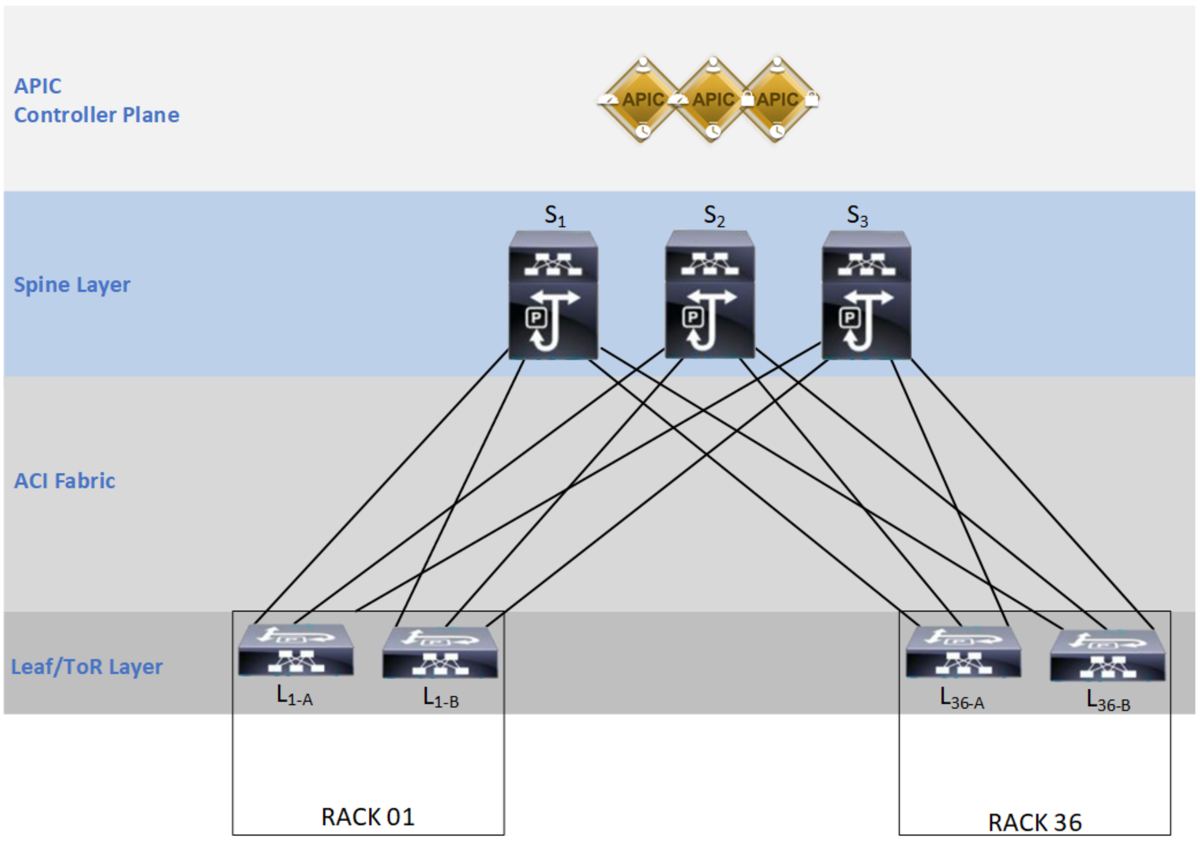


Figure 14-15 – ACI DCN Topology (Primary and Secondary DCs)

14.3.5 ToR to Spine Connection

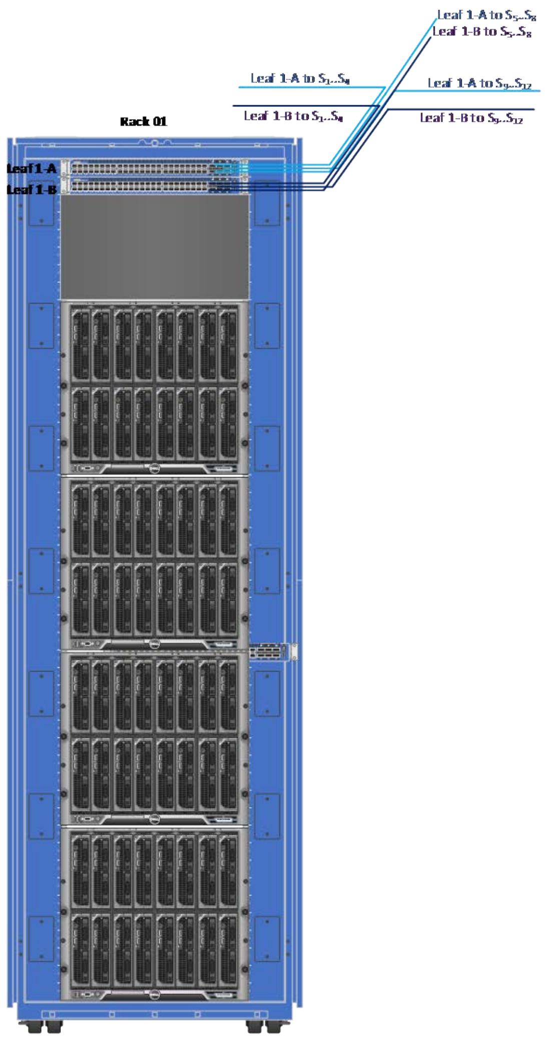


Figure 14-16 – ACI DCN ToR to Spine connection

Each rack contains 2 ToR switches (Cisco Nexus 9396), and 48 dual homed servers, each connected to both ToR with 2x10G links.

Each ToR switch has 12 uplinks connecting it to the Spine Layer (Cisco Nexus 9516), with the uplinks equally distributed on the 3 spines (as shown in Figure 14-16).

In terms of internal and external throughput, per rack/PoD, this design provides

- a total uplink throughput per rack of 2x12x40G=960Gbps
- access throughput per rack of 2x48x10G=960Gbps
- both the intra-pod and uplink connectivity are provided through non-blocking networks

As expected, access throughput and uplink throughput are the same.

This architecture, as shown in Table 14-1 , can scale up to 13,824 access ports: scaling up would require increasing the number of spines switches, and to re-cable the leaf switches according to the number of spines. This cabling activity, though, in modern datacenter could be fully automated.

14.4 Logical View

14.4.1 Tenants

Each private datacenter will have, initially, two tenants:

- Common Tenant: connecting all the tenants
- Production Tenant, processing all the traffic
- Test Tenant: a copy of the production tenant, useful for testing purposes (i.e. test new features, configurations, etc.)

The connection between Production and Common Tenant will go through the inter-tenant firewall, able to provide high level of granularity in access to the Test Tenant.

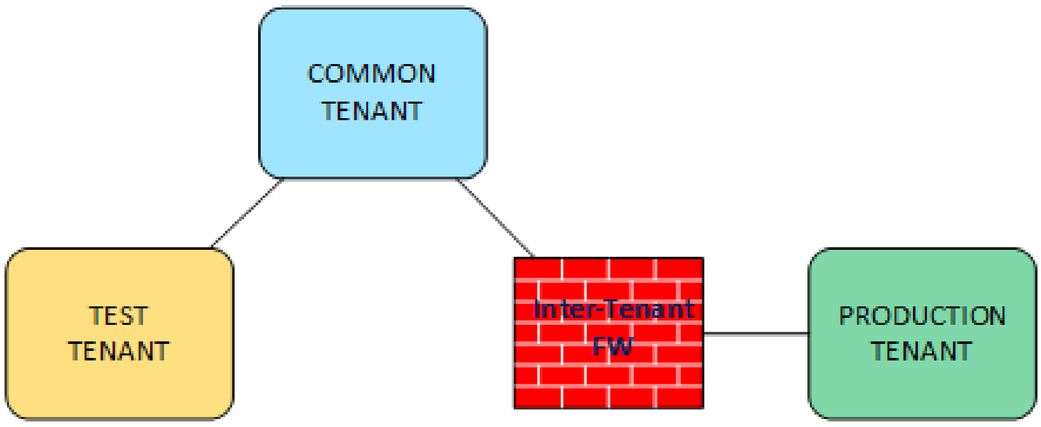


Figure 14-17 – ACI Tenant configuration

The objects described in the paragraphs below will be configured in the production Tenant, and a copy of the production Tenant will be deployed in the Test Tenant during the test invocation.

The common Tenant, as shown in Figure 14-17, will be used essentially as a Transit Tenant between Test and Production Tenants.

14.4.2 Primary Datacenter

As shown in Figure 14-18, the private Primary datacenter will have the following contexts:

- DCI
- PROD
- DEV
- DMZ
- CORE

The primary datacenter will be connected to the Distributed datacenter network through SAE1 and SAE2 gateways, physically connected to a border leaf, and logically terminated on a L3out,

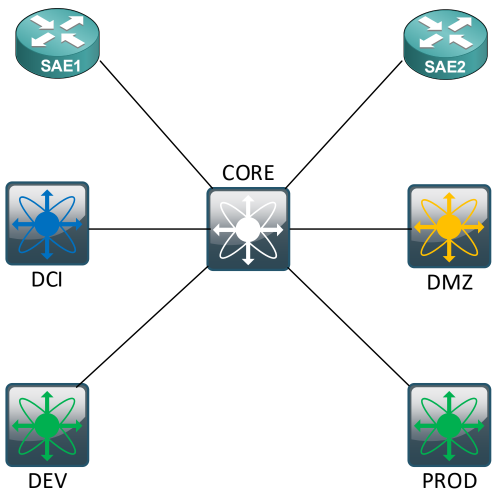


Figure 14-18 – ACI Contexts in the Primary Datacenter

The internal VRFs (DMZ, DCI, PROD and DEV) will be connected to the CORE VRF through a SFC providing application inspection and visibility.

14.4.3 Secondary Datacenter

As shown in Figure 14-19, the private Secondary datacenter will have the following contexts:

- DCI
- PROD
- UAT
- DMZ
- ISO
- CORE

The secondary datacenter will be connected to the Distributed datacenter network through SAE1 and SAE2 gateways, connected to a border leaf, and routed through the CORE Context.

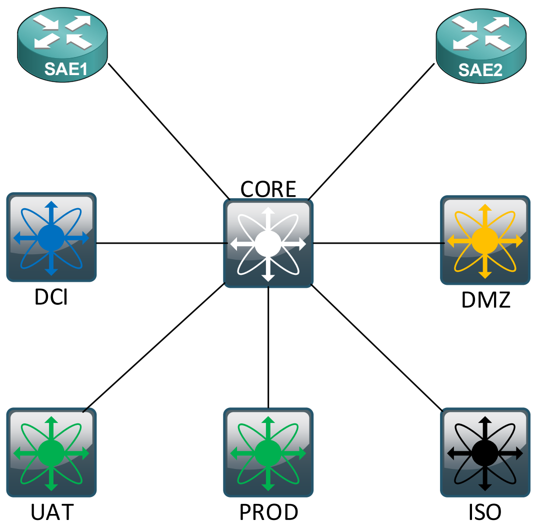


Figure 14-19 – ACI Contexts in the Secondary Datacenter

14.4.4 PROD and DEV Web Application

PROD, DEV and UAT VRFs can host many different workloads and applications. One of the key design tool provided by ACI is the Application Network Profile (ANP). This paragraph introduces a ANP for a generic Web application: we assume that the Enterprise is using a three-tier web application based on a set of Apache servers providing static content and acting as reverse proxies for various instances of application servers (i.e. Tomcat, Apache+PHP, etc.) which, in turn, access a database cluster (i.e. MySQL).

The application must be available to external users and protected through a NFC providing Application firewalling and load balancing. Figure 14-20 depicts the high level "design" of this application, showing the intent of the white-list model allowing only the protocols and ports strictly required.

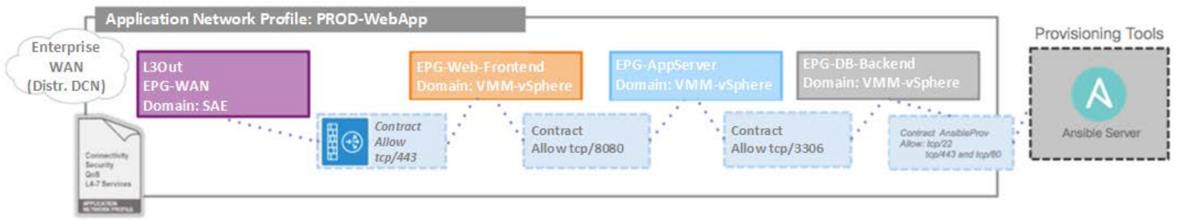

Application Network Profile: PROD-WebApp
Enterprise WAN (Distr. DCN)
L3Out EPG-WAN Domain: SAE
EPG-Web-Frontend Domain: VMM-vSphere
EPG-AppServer Domain: VMM-vSphere
EPG-DB-Backend Domain: VMM-vSphere
Provisioning Tools
Ansible Server
Contract Allow tcp/443
Contract Allow tcp/8080
Contract Allow tcp/3306
Contract AnsibleProv Allow: tcp/22 tcp/443 and tcp/80
Connectivity Security QoS L4-7 Services
Application Network Profile

Figure 14-20 – Web Application network profile

In ACI terminology, the high-level design presented above is called Application Network Profile, and basically describes the expected dataflow of an application through the ACI network.

A Firewall VNF will perform application level inspection and probably NAT towards internal addresses, and the load balancer VNF will be able to off load SSL and provide advanced load balancing features. It is very possible that these devices are physical and shared for other applications as well.

The only IP address that needs to be accessible from the outside networks is the VIP on the load balancer for this application. The load balancer cannot access the database, even if they are on the same segment, but only to the web machines that are part of the load balancing pool. Only the Web servers should have access to the DB, and they do not need to communicate with one another.

14.4.4.1 End-to-end access to DEV and PROD

Figure 14-21 shows the north-south (User-to-Service) and East-West (Service-to-service) dataflow for an end user accessing PROD and DEV workloads from the WAN to the Primary Datacenter

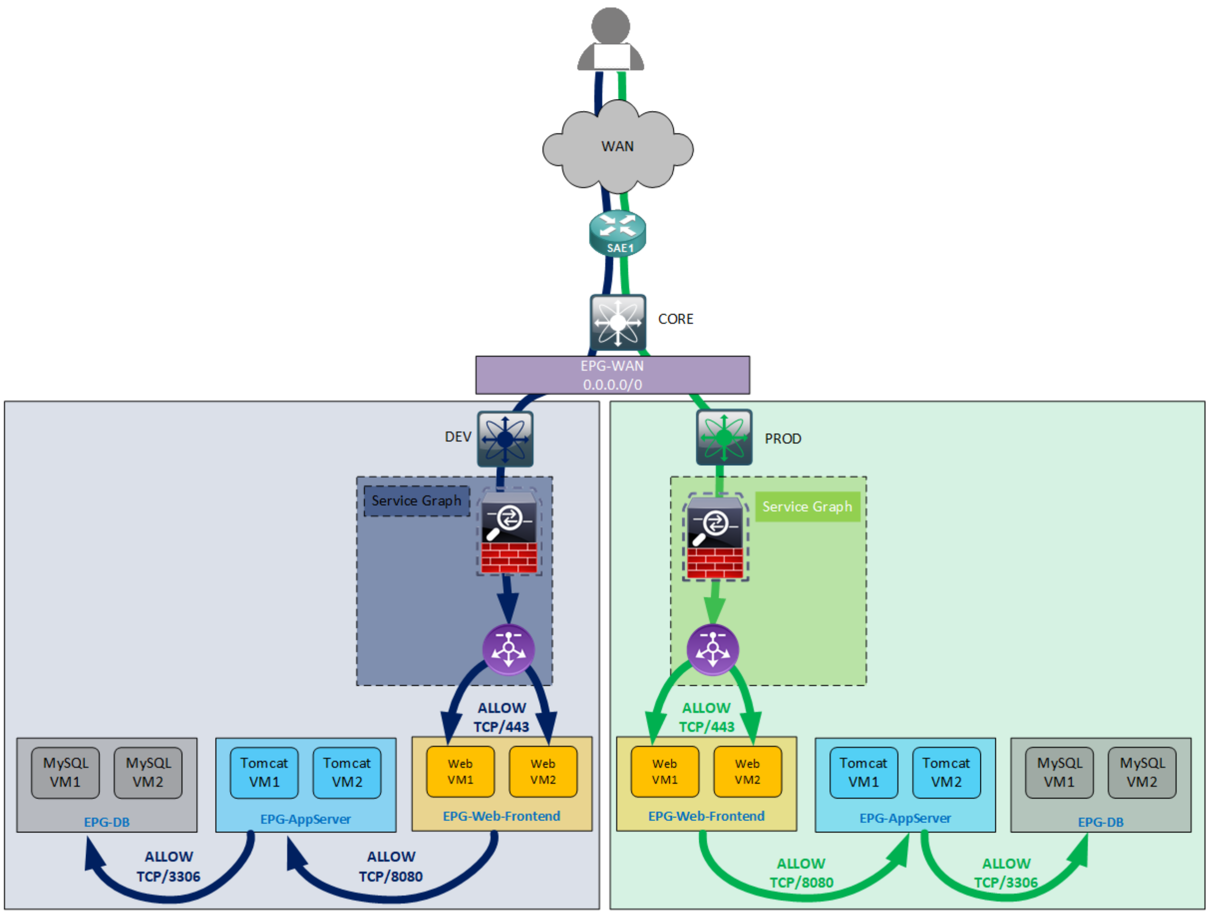


Figure 14-21 - End-to-end access to PROD and DEV Workloads

In this example an external EPG represents a “default” route with all hosts external to our application. A contract between that external EPG and the Web-Prod and Web-DEV EPG will enable users to access the application.

This contract allows any external host to access the VMs in Web-Prod EPG on port TCP/80 by having the traffic flow before through a NGFW and a LB. This contract allows any external host to access the VMs in Web-Prod EPG and Web-DEV on port TCP/443 by having the traffic flow before through a SFC containing a NGFW VNF and a LB VNF. These VNFs are inserted in the path through a Service Graph.

14.5 Integration with VMWare vSphere

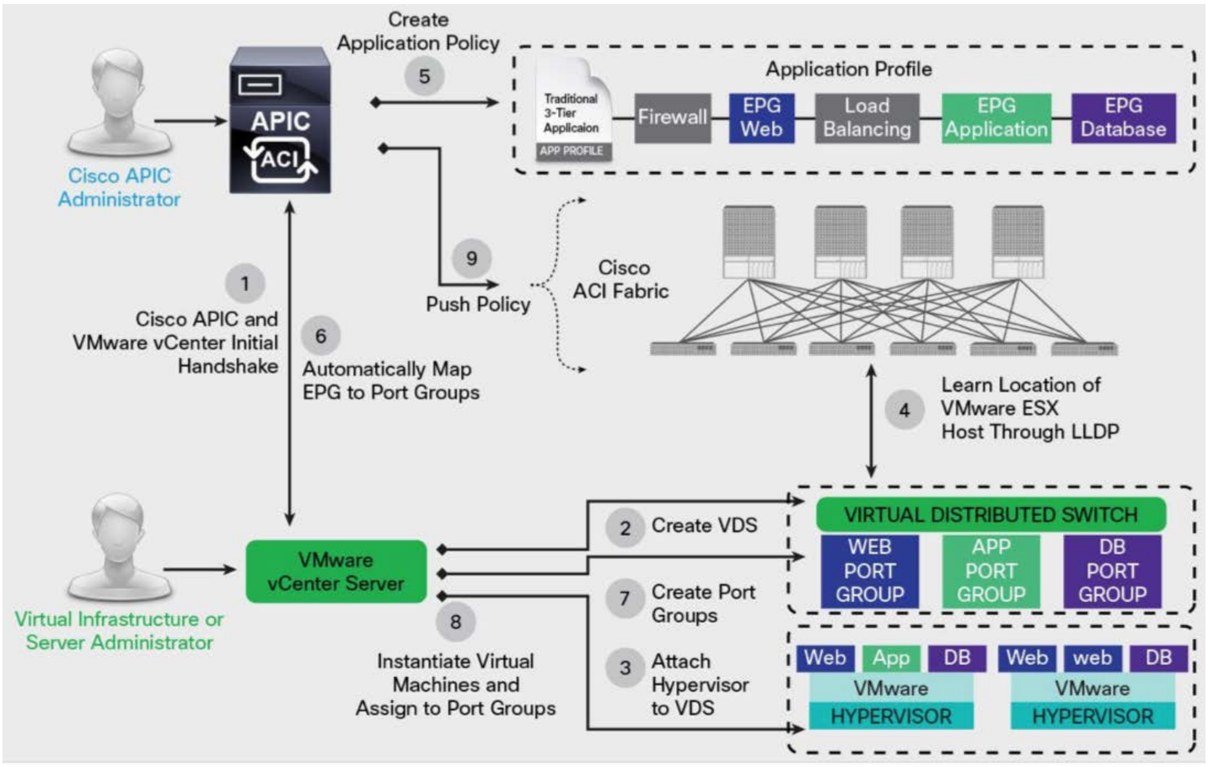


Figure 14-22 – Cisco ACI integration with VMWare vSphere [117]

Cisco APIC integrates with the VMware vCenter instances [117] and can extend the ACI policy to VMware vSphere workloads. Cisco APIC creates a distributed virtual switch (DVS) in VMware vCenter, through this DVS the APIC can create the virtual networks to which the VMs are connected. Once the DVS is created on the VMware environment, Cisco APIC can manage the virtual network infrastructure components.

The APIC administrator creates EPGs and can push them to VMware vCenter as port groups on the DVS. Server administrators can then associate the virtual machines and provision them accordingly (Figure 14-22).

15 Comparison of the high-level designs

15.1 Abstract

This chapters defines the comparison criteria for the three high level designs presented in chapters 12,13 and 14, compares their models with a theoretical reference model and show the benchmarks against the defined criteria. The last paragraph presents a summary of the Gartner report [118] on datacenter network, showing a different standpoint.

15.2 Criteria

The comparison criteria consider

- Physical infrastructure
- Network Control plane (SD-Model)
- NFV support (L2, L3, physical and virtual SFC)
- Orchestration
- Micro-segmentation, container support

15.3 Physical infrastructure

15.3.1 Evaluation Parameters

The comparison of the physical infrastructure for the three HLDs assumes that

1. The network should be able to support (at least) 1,536 dual homed hosts connected with 2x10G interfaces
2. Oversubscription can vary from 3:1 to 1:1

To benchmark the physical network design of the presented HLDs, a reference model will be introduced.

The evaluation will consider the following criteria:

- Network size / complexity
    - Spine (number of switches, port per switch, total number of ports)
    - Leaf (number of switches, port per switch, total number of ports)
    - Total number of switches
    - Total amount of 10G equivalent ports: every 40G ports will be counted as 4x10G ports.
- Design Quality
    - Oversubscription and bi-sectional bandwidth
    - Maximum scalability

- Cost and ports
  - Cost
  - Cost per port
  - Overall cost
  - Percentage of the total cost spent for the fabric

15.3.2 Reference model for the physical DCN

The model must satisfy the following requirements:

1. Scale up to 3,072 ports @10G
2. Non-blocking network, bi-sectional bandwidth 30.72Tbps

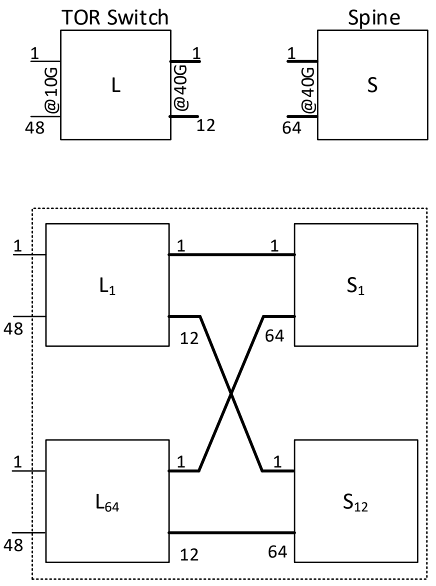


Figure 15-1 – Reference Leaf-Spine topology model

The reference topology shown in Figure 15-1 has the following characteristics

- Leaf/ToR: 48ports @10G + 12P@40G (non – blocking). This implies 64 leaf switches as ToR for 32 racks with 48 dual homed hosts each
- Spine: 12 Spine switches with 64 ports each
- 3,072 access ports @10G and 64 Leaf Switches
- 768 spine ports@40G and 12 Spine switches, 30.72Tbps Bi-sectional Bandwidth
- 9,216 10G-Equivalent ports (3,072 + 2x768x4)

15.3.3 Design performances

This paragraph introduces the design performances evaluating the parameters described in Par.15.3.1. The relative weight of the parameters depends on how much the Enterprise values an aspect (i.e. oversubscription vs overall Price) and it is out of the scope of the present paper. The price values are taken from the Gartner Report introduced in Par. 15.7

	Description		BCF	Contrail	Cisco Opt "B"	Reference Model
	Max Dual homed hosts number		1536	1536	1536	1536
Network Size	Spine Switches	number	6	4	3	12
		ports per switch	64	64	288	64
		total Spine ports	384	256	864	768
	Leaf Switches	number	64	32	64	64
		40G uplinks	6	8	12	12
		total 40G Uplinks	384	256	768	768
		10G ports per switch	48	96	48	48
		total 10G ports	3072	3072	3072	3072
	Total Switches		70	36	67	76
	10G ports equivalent		**6144**	**5120**	**9600**	**9216**
Quality of the Solution	Access Bandwitdh		30720	30720	30720	30720
	Bi-sectional BW		**15360**	**10240**	**30720**	**30720**
	Maximum scalability		**3072**	**3072**	**3456**	**3072**
	Support to 100G spine		**YES**	**YES**	**YES**	**N/A**
Cost	**Unitary cost per per 10G port**		**$ 250**	**$ 330**	**$ 375**	**$ 165**
	Total Cost		**$ 1,536,000**	**$ 1,689,600**	**$ 3,600,000**	**$ 1,522,299**
	% of overall cost needed for Fabri		**50%**	**40%**	**68%**	**67%**

Table 15-1 - Comparison of the physical topologies for the three HLDs

Table 15-1 shows how the three solutions perform on the selected parameters and depicts also the (hypothetical) performances of the reference model. The cost per-port of the reference model is the average cost indicated in the mentioned Gartner Report. Next paragraph will score the three solutions against the refence model.

In terms of access ports, the three solutions provide, by design, the same value, so the differences in the number of 10G equivalent ports (which is an indirect measure of the network size or network complexity) are connected only to the Fabric layer (leaf-spine connectivity) and related to the over-subscription value (in case of BCF and Juniper) or to the line-card size on the spine switches (Cisco)

15.3.4 Benchmarking

Table 15-2 below shows the score of the three designs respect to the reference model

Description	BCF	Contrail	Cisco Opt B	Reference Model (Actual Values)	
10G ports equivalent	67%	56%	104%	9216	ports
Bi-sectional BW	50%	33%	100%	30720	Gbps
Maximum scalability	100%	100%	113%	3072	hosts
Cost per 10G Port	151%	200%	227%	$ 165	USD
Total Cost	101%	111%	236%	$ 1,522,299	USD

Table 15-2 – HLDs Benchmarks for the Physical topology

15.3.4.1 Big Switch BCF

BCF network size is 67% of the Reference model but applies an oversubscription ratio of 2:1, providing half of the reference bi-sectional throughput. In terms of access hosts and scalability, the BCF solution can scale up to the same value of the reference model and its cost is aligned with the cost of the model.

15.3.4.2 Juniper Contrail

Juniper network size is 56% of the Reference model but applies an oversubscription ratio of 3:1, providing one third of the reference bi-sectional throughput.

The solution can scale up to the same value of the reference and its cost is 10% more expensive than the reference cost. It is worth noting that the cost per port is comparable to the Cisco solution, so is quite high, but the overall number of ports (network size) is smaller because of the choice of providing only 1/3 of the reference Bi-sectional bandwidth.

15.3.4.3 Cisco ACI

From a pure network complexity, it is quite clear that Cisco fabric is much more complex than Juniper and BCF, and even more complex than the reference model, providing 4% more ports than the reference (1,632 ports against 1,536). From the Design quality standpoint (bi-sectional BW, scalability), though, the complexity mentioned above can be explained by the fact that Cisco solution offers better performances than the reference model: it is the only design providing the maximum theoretical throughput (30.72Tbps) and even better scalability (up to 3.456 access ports) than the reference model.

Again, network complexity and improved performances have also another effect: from the cost perspective, Cisco costs 2.36 times more the (hypothetical) reference cost, 2.34 times more than BCF and 2.11 times the price of Juniper solution.

15.4 Network Control Plane (SD-Layer)

15.4.1 Evaluation parameters

- Control plane separation, simplified device (*brite* box), centralized control, multi-vendor devices support through Southbound protocols.
- Network automation and virtualization, Openness
- Intent-based policies, Zero Touch provisioning

15.4.2 Benchmarking

Description	BCF	Contrail	ACI
Control plane separation	100%	25%	25%
Simplified device (brite box)	100%	0%	0%
Centralized control	100%	100%	100%
Multi vendor devices (through Southbound)	100%	75%	0%
Network automation and virtualization	100%	100%	100%
openness	100%	100%	100%
Intent-based policies	0%	0%	100%
Zero-touch provisioning	100%	100%	100%

Table 15-3 – HLDs Benchmarks for the Network Control Plane

15.4.2.1 Big Switch BCF

BCF approaches to SDN adopts a "classical" SDN approach with full separation between control plane and data plane, centralized control, and support of third party devices (even though it provides a Hardware compatibility list, or a certification for some platform, see Par 12.2.4)

At the time of the writing (Dec 2017) BCF doesn't support intent-based policies.

15.4.2.2 Juniper Contrail

Juniper Contrail control plane still resides on the switches and Contrail does not directly program the data plane, therefore Contrail approach is not a classical/Open SDN. Contrail supports integration with multivendor environments [83], but not brite box switches

At the time of authoring this paper, Contrail does not support intent-based networking.

15.4.2.3 Cisco ACI

Cisco ACI approach to SDN relies on a high level/intent-based API controller through a standard southbound protocol (OPFlex) controls Cisco Nexus 9k switches running in ACI Mode. The adoption of OPFlex, at the time of this paper, is limited to only Cisco Nexus 9k switches, therefore this solution supports only this hardware platform.

15.5 Network Function Chaining and micro-segmentation

15.5.1 Evaluation parameters

Support to

- VNF: Layer 2, Layer 3 and Layer 4-7
- NFC: Physical, Virtual and Hybrid
- Micro-segmentation: Layer 2, Layer 3-7

15.5.2 Benchmarking

Description		BCF	Contrail	ACI
VNF	Layer 2	0%	100%	100%
	Layer 3	100%	100%	100%
	Layer 4-7	100%	100%	100%
NFC	Physical	100%	100%	100%
	Virtual	100%	100%	100%
	Hybrid	100%	100%	100%
μ-segm	L2	0%	0%	100%
	L3-7	100%	100%	100%

Table 15-4 – HLDs benchmarks for the support to NFC and μ-segmentation

15.5.2.1 Big Switch BCF

BCF supports layer 3 service insertion, does not provide any layer 2 functionality (layer 2 segregation can be achieved through another solution from Big Switch Networks: Big Monitor Fabric). BCF support network layer 3 service function chaining with both virtual and physical appliances, and provides μ-segmentation functionalities only through the integration with orchestration/overlay platforms (i.e. OpenStack/OpenShift, VMware NSX)

15.5.2.2 Juniper Contrail

Juniper Contrail supports layer 2 and layer 3 VNF and connecting them through physical, virtual and hybrid NFC. Juniper fully supports micro segmentation from layer2 upwards

15.5.2.3 Cisco ACI

Cisco ACI supports layer 2 and layer 3 VNF and connecting them through physical, virtual and hybrid NFC. Juniper fully supports micro segmentation from layer2 upwards

15.6 Orchestration

15.6.1 Evaluation parameters

- Orchestration platforms:
    - VMWare
    - Open Stack
    - Open Shift
- API
    - North bound REST API support
    - South bound Open Protocols (multivendor)

15.6.2 HLD performances

	Description	BCF	Contrail	ACI
Orchestration	VMWare	75%	100%	75%
	Openstack	80%	100%	100%
	Openshift	80%	100%	100%
API	Northbound REST	100%	100%	100%
	Southbound Open Protocols	100%	100%	25%

Table 15-5 – HLDs Benchmarks for the support to Orchestration platforms

15.6.2.1 Big Switch BCF

BCF integrates with VMware, OpenStack and OpenShift but lacks some Layer 2 abilities (as pointed out in the previous paragraph)

Both the northbound (REST) and southbound APIs (OpenFlow with some extensions) are fully supported from different vendors.

15.6.2.2 Juniper Contrail

Juniper contrail offers two different integration modes with VMWare vSphere and fully integrates with Openstack/OpenShift.

Its REST northbound API is fully documented and the integrates with different vendors. It supports multiple southbound protocols (OVSDB, XMPP, EVPN, etc.), which, theoretically opens to different vendors.

15.6.2.3 Cisco ACI

Cisco ACI fully supports and integrates with VMWare, OpenStack and OpenShift. Its northbound API is open and supported by different vendors.

As already pointed out, the southbound protocol (OPFlex) is supported only by Cisco, and at the moment of the writing of this paper, only on the Nexus 9k platform.

15.7 Gartner Report for Datacenter Networking (2017)

15.7.1 Market Directions

15.7.1.1 The CLI Is Dead; the API Is Cool

In the 2017 report for Datacenter networking [118], Gartner recognizes that the market is shifting from using device-by-device command line interface (CLI)-driven configurations to a centralized policy-based mode of operations: the estimation is that, by 2020, only 30% of network operations teams will use the CLI as their primary interface, down from 85% in 2016. API brings the real innovation on the management plane because it enables complete automation of repetitive tasks, but also integrates with higher-level infrastructure. Device-level automation tools (like Ansible, Puppet or Chef) and APIs can also facilitate the implementation of a DevOps model.

15.7.1.2 Value Continues to Shift Toward Software

Although there is consensus on the growing importance of software, there are still vendors investing in proprietary hardware (like Cisco and Juniper ASICs). Anyway, most vendors are now focusing the innovation on software and leveraging white box devices to build their switching products. Many vendors rely on more than one chip supplier, and the competition in the merchant silicon market is increasing. This creates a need for the software to be portable across different chipsets. Gartner believes that merchant-based platforms can meet the needs of at least 80% of enterprises.

15.7.1.3 Fabrics Are the New Normal

Most vendors offer fabric architectures based on different (and usually proprietary) solutions, but can provide similar advantages as SDN, at least in terms of centralized point of control and programmability, if not in terms of lower vendor lock-in and cost. Examples are Arista Networks Cloud Vision, Juniper Networks Junos Fusion, Cisco ACI and NFM.

15.7.1.4 Analytics and Intent-Based Networking

All vendors put great emphasis on network analytics and streaming network telemetry data as a new way to feed analytical tools, and scales beyond legacy SNMP. These capabilities enhance troubleshooting and Application Visibility and Control.

Analytics together with automation are key factors for intent-based networking, a principle that represents the next frontline for vendors. With intent-based networking, high-level policies are translated into actual network configuration changes and implemented through automation. Real network behavior is then compared against the desired behavior to enable any necessary remediation and ensure that the correct policy is actually enforced. The idea is to build a self-driving network.

Implementation of intent-based networking systems requires data from sensors (network analytics) to feed the calculation of abstract models built with artificial intelligence algorithms, and then actuators to implement the intent-based policies.

15.7.1.5 Open Networking

The inclination to move away from proprietary solutions is continuously growing. Gartner end-user survey indicated that 42% of clients consider open standards and multivendor interoperability support a mandatory requirement, 34% consider it very important, and 20% consider it somewhat important, so openness is a relevant buying criterion for 96% of the end users.

These results are quite impressive, but in addition, 75% of the end users indicated that they expect an increase in relevance of open networking in their purchasing decisions in the next 24 months.

15.7.1.6 Disaggregation/Brite Box

Disaggregation of hardware and software can be a first step toward more vendor independence. Adoption of white-/brite-box switching has increased significantly within hyperscale datacenters over the past several quarters, and Gartner predicts it to reach 22% of the total datacenter switch market by 2020.

Enterprise adoption of brite box grows, and Gartner estimates about 1,000 enterprise customers as of March 2017. This includes some very large accounts in finance and the public sector. However, established vendors (for example, HPE and Dell) did not generally lead in enterprise with solutions based on brite box and third-party software so adoption of disaggregated solutions in mainstream enterprises is still limited.

15.7.1.7 Hyper Converged Integrated Systems (HCISs)

HCISs tightly couple compute, network and storage hardware in a system and are gaining popularity since they streamline operations and reduce provisioning times. The networking components of an integrated system are largely prescribed, which results in the transition of the physical access layer network buying decision to an integrated server/storage/network decision. Examples are Nutanix and HPE SimpliVity.

15.7.1.8 Containers

Container networking is at the maturity stage where VM networking was about five years ago; it is fast-evolving and fragmented, with all vendors starting to provide solutions and launching new initiatives. It is difficult to determine which vendors and architectures are best-suited for their usage scenarios, but at this time, containers have very limited production deployments in enterprise datacenters.

However, this will be an important decision in the next three years, as containers are widely used for development of *Mode 2* applications[22] that sooner or later will need to be deployed in production

15.7.2 Magic Quadrant

Network offering for datacenters is evolving to improve agility and provide better cloud architectures, with enterprises needing more integration and orchestration between the network and the rest of the infrastructure. The datacenter market is being driven by replacement of legacy switches, expansion of network infrastructure, and the adoption of new solutions that increase cost-effectivities of the infrastructure and easiness of operations.

According to Gartner price is not the main driver for most enterprise buyers, and the availability of lower-cost alternatives is frequently not sufficient to cause a supplier change.

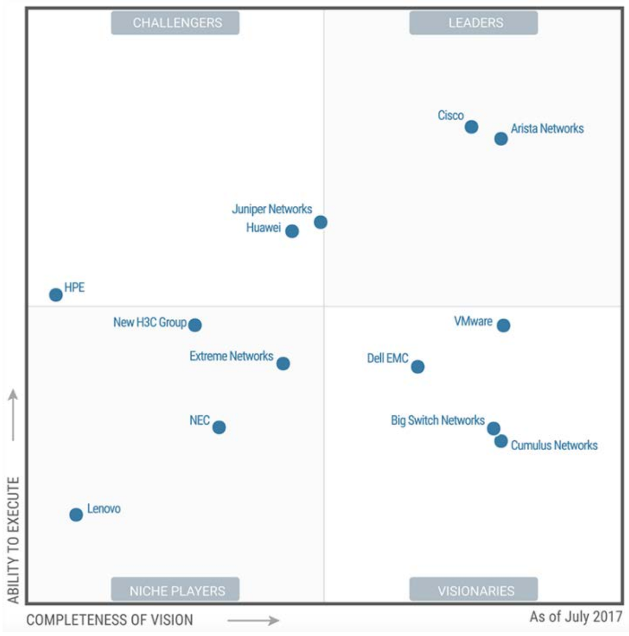


Figure 15-2 - Gartner Magic Quadrant for Datacenter Networking (2017)

[22] Bimodal IT is the practice of managing two separate, coherent **modes** of IT delivery, one focused on stability and the other on agility. **Mode** 1 is traditional and sequential, emphasizing safety and accuracy. **Mode 2** is exploratory and nonlinear, emphasizing agility and speed.

15.7.3 Big Switch BCF

15.7.3.1 Description

Big Switch Networks in 2014 published Big Cloud Fabric (BCF), described in Chapter 12, Big Switch used its network packet broker (NPB) solution called Big Monitoring Fabric (BMF) as Trojan horse for accounts that were not ready for a white /brite box as a primary datacenter network but were open to testing technology for a specific application (NPB). Big Switch contributes to open source communities and its philosophy is to combine open source components with their modules to provide business-class solutions.

Big Switch should be considered by organizations interested in a commercial fabric business solution that leverages the white / brite-box hardware platforms and can scale in very large environments.,

15.7.3.2 Strengths

Big Switch provides a commercial cost-effective SDN fabric solution based on industry-standard white-/brite-box hardware that is 30% to 60% less expensive than top-brand hardware, resulting in three years total cost of ownership (TCO) savings that can reach 50%.

BCF can be integrated with leading cloud orchestration platforms (like VMware or Red Hat Enterprise Linux OpenStack) to provide a highly automated network solution.

15.7.3.3 Cautions

Big Switch is a small company with limited footprint outside North America and limited experience in integrating with complex business environments with a large legacy estate. However, it has channel partnerships with companies such as Dell EMC and HPE to extend market coverage.

While being one of the first Classical SDN and OpenFlow supporters, Big Switch uses some customization of this protocol within its BCF. Therefore, third-party switches cannot be integrated into BCF.

In large multi-tenant environments with more logically isolated pods, role-based access control (RBAC) to enable per-tenant administrative control is limited. We expect this gap to be resolved by the end of 2017. Big Switch does not support interfaces for converged storage networks (FC or FCoE); it is suitable for environments using IP storage.

15.7.4 Juniper Networks

15.7.4.1 Description

Juniper Networks has a large portfolio of datacenter networks under the Unite Cloud framework. Contrail and Juniper Networks Overlay are described in Chapter 13

In VMware environments, Juniper switches can be controlled by NSX, enabling physical and virtual integration. Juniper is positioned as a supplier n. 3, based on the market share of 2016 revenue, and increased its share in corporate datacenters by introducing new products into the QFX, DCI, and analytics family and optimizing their marketing messages on Juniper Unite Cloud. As a result, the supplier recorded strong growth in 2016 at higher market rates.

All major organizations should consider Juniper in their networking lists for datacenters, especially those environments that want to have more vendors and solutions that can interact in multivendor environments.

15.7.4.2 Strengths

Juniper has a portfolio of datacenter solutions based on common building blocks belonging to two product families (QFX5K and QFX10K), and experience in supporting mission-critical infrastructures. The vendor favors open solutions that leverage standards (like EVPN) and enable multivendor interoperability. Juniper's Contrail is a popular commercial SDN controller for OpenStack, with increasing deployments in the enterprise. The vendor provides cost-competitive solutions and is an effective option for clients that want to have at least two network suppliers for their datacenter.

15.7.4.3 Cautions

Juniper still lacks some focus and market presence in the enterprise segment, although it made significant progress in 2016. Juniper has a partnership with VMware for integration with NSX but is also competing with Contrail. The two companies pursue their sales strategies (partnering versus competing) on an account-by-account basis and that might confuse clients. The vendor makes most of its overall revenue with service providers and large enterprise clients. Midsize clients should assess its market coverage level and the capabilities of its local channels. Juniper is a networking and security company and lacks the full datacenter portfolio of some larger players. Juniper does partner with channel and industry partners that can provide compute and storage components (for example, for hyper converged systems).

15.7.5 Cisco ACI

15.7.5.1 Description

Cisco is the most visible provider in the datacenter network market, and in 2016 it has driven the market with a revenue more than five times higher than that of its closest competitor but had a -3% of revenue share and -6% quarterly share portfolios over 4Q16. The Cisco Datacenter's peak offering is represented by the Cisco Nexus 9000 switches running Cisco ACI software, described in Chapter 0, though offering Nexus Fabric Manager (NFM), an alternative fabric solution, and BGP EVPN.

Gartner has seen a consistent adoption of these offers over the last year and Cisco now reports over 3,500 ACI customers; however, many use ACI for network automation and few use the policy-based features. Over the last year, Cisco has released a new line of switches based on its cloud-based ASIC, a move against the industry's trend toward merchant silicon

Cisco should be considered for all datacenter networking opportunities globally, especially from those customers who prefer few strategic vendors.

15.7.5.2 Strengths

Cisco has a deep and broad portfolio familiar to its channel partners, global support capabilities and a large loyal installed base of customers. Cisco ACI provides fabric automation, policy-based orchestration and service chaining capability, and supports integration with dozens of partners. Cisco is a full-stack datacenter infrastructure vendor, providing networking, compute, storage and security, which is desirable for enterprises looking to reduce their number of suppliers. Cisco supports multiple options for automation including ACI, Puppet/Chef/Ansible/Python, OpenStack, Yang, and RESTful APIs.

15.7.5.3 Cautions

Cisco customers report that migrating to Cisco ACI is complex, and most clients have not implemented the ACI policy-based model. The vendor's overall proposals are the most expensive observed by Gartner: Cisco's cost per 10G port ($307.40) is much higher than the market average for 2016 ($165.18). Some clients have reported issues on software stability and on ACI-based micro-segmentation. Cisco currently lacks a fixed form factor ACI spine switch based on the Cloud Scale ASIC with 25/50/100GbE ports. This forces customers to make a suboptimal hardware choice when a fixed form factor ACI spine switch is desired, which Gartner believes is a good fit for most mainstream enterprises. Gartner anticipates that the vendor will address this in 2017.

16 Future Developments

16.1 Abstract

This chapter presents some challenges, (possible) evolutions and lessons on the Software-defined Datacenter network domain learned by the SDN implementations of Facebook, Google and Microsoft. The focus of the chapter is more on the Software-defined part of the SDN paradigm, or the control overlay. The underlay seems to be a done deal, being basically defined as a variation of a (mostly non-blocking) folded Clos network, Pod-based, where Amazon first, and now Facebook, use 25G on the access and 50G/100G on the uplink.

The idea behind the chapter is to discuss some of the lessons learned by the biggest (early) adopters of SDN and apply some of those concepts in the Enterprise domain.
This chapter is organized in six paragraphs

- The four pillars of Google SDN, based on the information presented in [119]
- The need for hierarchy with H-SDN and Microsoft ([120]) [121] and [122])
- Disaggregation and Data Plane programming in Facebook SDN ([123], [129]
- SD-WAN for the Enterprise (with reference to MSDC, see [124] et al)
- Stitching all together: a future for the Enterprise Global DCN

16.2 Google SDN

16.2.1 The four pillars

Amin Vahdat, Fellow & Technical Lead for Networking at Google, during his keynote at the Open Networking Summit 2017 [125] stated that the question if SDN is a good idea or not is closed: Software Defined Networking is simply how they do networking, Google builds its cloud architecture disaggregating storage and compute, spreading them across the entire datacenter. This approach increases substantially the bandwidth and latency requirements for accessing anything anywhere, pushing further the pressure on the network infrastructure within the datacenter.

At Google, SDN started its journey (see Figure 16-1) in 2013 with B4, a wide area network interconnect for their datacenters, then in 2014, followed Andromeda, their network virtualization stack that form the basis of Google Cloud. In 2015, Google added Jupiter for datacenter networking (see Par. 5.4.3).

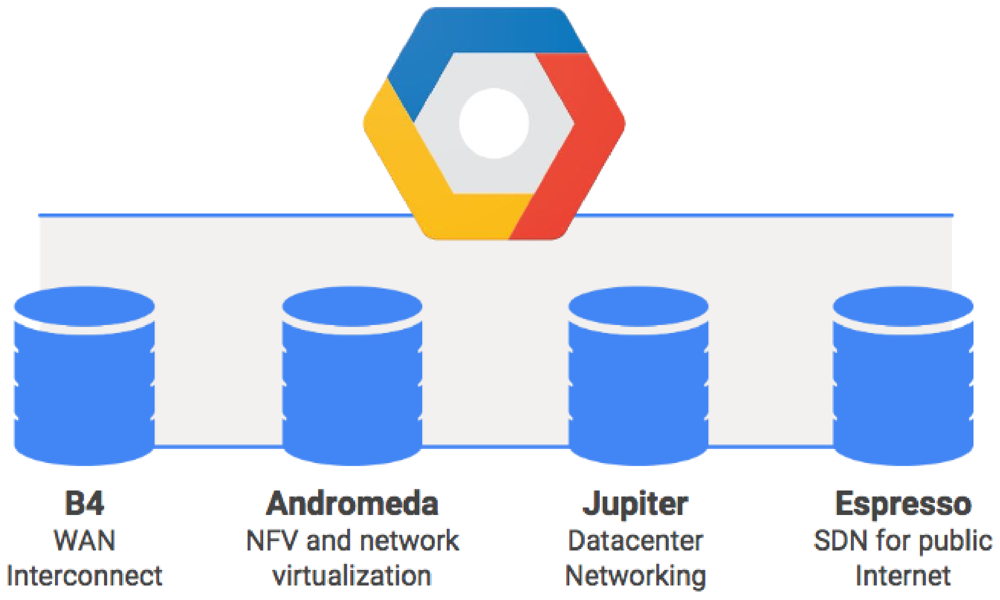


Figure 16-1 – The four pillars of Google SDN [119]

16.2.2 Google Espresso

Nowadays, Google has Espresso, which is the SDN for the public Internet with per-metro global views and real-time optimization across many routers and many servers (see Figure 16-2)

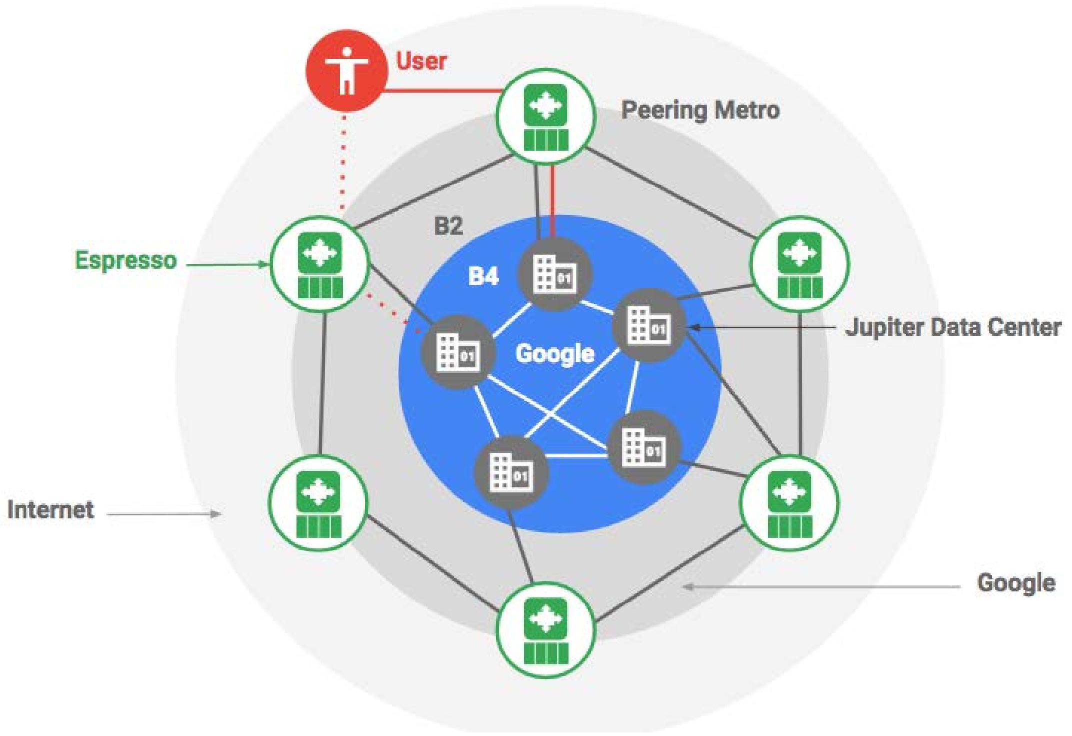


Figure 16-2 – Google Espresso Metro [119]

"What we need to be doing, and what we will be doing moving forward, is moving to Cloud 3.0. And here the emphasis is on compute, not on servers", Vahdat says. According to Vahdat, Cloud 3.0 implies:

- Storage disaggregation: the datacenter is the storage appliance
- Seamless network telemetry and scale up/down
- Transparent live migration
- Open Marketplace of services, securely placed and accessed
- Applications + Functions (not VMs)
- Policy (not middleboxes)
- Actionable Intelligence (not data processing)
- SLOs (not placement/load balancing/scheduling)

The high-level challenges above are consistent with an agile/proactive/software-defined view of the infrastructure: datacenter as a code, from servers to containers, from boxes to function virtualization, from static access control lists to intent-based dynamic policies, and so on.

16.2.3 The six high availability principles

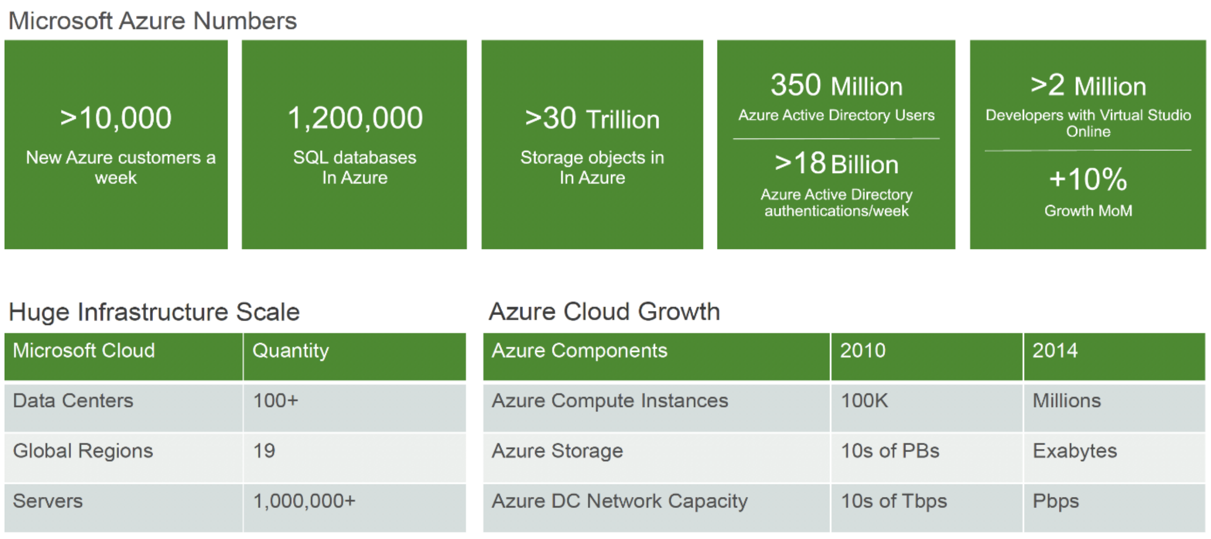
Even though Google's context is very specific, there is no doubt that its expertise in the domain might provide some advice in the context of this paper (Enterprise SDN). The main strategy, from the network infrastructure perspective, could be summarized in the statement "*Evolve or Die*" which is also the title of one of the research paper [126] where the authors define the Six high availability principles, and the challenges, behind Google's (Software Defined) Network:

1. Use Defense-in-depth
2. Maintain consistency within and across planes
3. Fail open
4. An ounce of prevention
5. Recover Fast
6. Continuously upgrade the network

16.3 Microsoft Azure and HSDN

16.3.1 Challenges

The cloud is growing at an unprecedented rate, and this growth demands an ability to scale the underlay network to tenths of millions of endpoints and hundreds of millions of VMs and VNFs (see Figure 16-3)


Microsoft Azure Numbers

>10,000 New Azure customers a week	1,200,000 SQL databases In Azure	>30 Trillion Storage objects in In Azure	350 Million Azure Active Directory Users >18 Billion Azure Active Directory authentications/week	>2 Million Developers with Virtual Studio Online +10% Growth MoM

Huge Infrastructure Scale

Microsoft Cloud	Quantity
Data Centers	100+
Global Regions	19
Servers	1,000,000+

Azure Cloud Growth

Azure Components	2010	2014
Azure Compute Instances	100K	Millions
Azure Storage	10s of PBs	Exabytes
Azure DC Network Capacity	10s of Tbps	Pbps

Figure 16-3 – the scale and growth of Microsoft Cloud [127]

This growth adds pressure on the infrastructure, with reference to

- Brite boxes switches
- ECMP forwarding
- Overall cloud scaling ability

16.3.1.1 Brite boxes switches

The use of brite box switches to scale the underlay network requires solutions to the potential explosion of the routing/forwarding tables in the network nodes as the size of underlay network grows. The FIBs and RIBs of current commodity switches are relatively small: they typically contain only 16K or 32K entries and are clearly inadequate to support all the hosts of a hyper-scale cloud. The common strategy to reduce the number of entries is the IP Address/prefix summarization; however, it doesn’t completely remove the scalability challenge many reasons:

1. The addresses may not be summarized in a way fully matching the DCN structure
2. The dynamic environment in the DC/cloud might require the handling of granular prefixes in the network. For example, VM and VNF migration may cause pollution in the routing table if their addresses are carried in the move, with the result of having many host-based routes stored in the tables.

3. In a DCN based on Clos topology, the ECMP support itself may be a major factor. For example, if the individual outgoing paths belonging to an ECMP group carry different outgoing labels, a single destination contributes F entries in the LFIB rather than just one, with F being the size of the ECMP group corresponding to that destination. Traffic Engineering tunnels are of course another major contributing factor to the FIB/LFIB size since each tunnel may require its own entry in the table.
4. Moreover, commodity switches use relatively small buffers, which worsens the challenge of accomplishing high resource utilization in the DCN. Indeed, since the DC must be able to support a wide range of traffic loads, it is usually designed with a lot of spare capacity, and its fabric is not meant to run at full load. At hyper scale, however, the assumption that "*capacity in the DC is plenty and cheap*" does not hold anymore and designing with that principle in mind can make scaling exorbitantly expensive.

16.3.1.2 ECMP forwarding

Congestion avoidance and traffic management in modern DCN depend deeply on ECMP forwarding. This solution, however, presents many challenges:

- ECMP approach becomes difficult with certain traffic patterns and does not protect properly QoS flows.
- To minimize packet re-ordering, ECMP is usually applied per-flow rather than per-packet, therefore hot spots may still occur for relatively long durations.

Unfortunately, today, any-to-any, end-to-end (server-to-server) Traffic Engineering (TE) does not scale for several reasons:

1. TE tunnels may cause routing/forwarding table explosion;
2. TE path and bandwidth allocation computation is a NP-complete problem;
3. Establishing TE tunnels takes considerable time, in that it generates a considerable volume of control traffic and network state that needs to be maintained and synchronized. TE rapidly becomes unaffordable at scale, and it is hardly applicable to rapidly changing traffic mixes.

For these reasons, today, TE is used primarily in the DCI/WAN and in general in networks with relatively small numbers of endpoints, to set up the "highways" between selected destinations; in the DC, it is used very sparingly for few, sensitive flows that are known to have long duration

16.3.1.3 Cloud Scaling

Scaling the cloud at low cost requires minimizing operational and computational complexity throughout the network. Many other important design objectives and considerations help reducing cost and complexity. They include:

1. Removing vendor dependency, using open interfaces;
2. Minimizing number and complexity of protocols and technologies;
3. Unifying forwarding in the DC and DCI, as well as in the underlay and overlay;
4. Simplifying management and reducing likelihood of configuration errors by unifying toolsets and increasing automation.

16.3.2 Hierarchical SDN goals

Hierarchical SDN (H-SDN) is a solution to the challenges presented in the previous paragraph. The goals of H-SDN are:

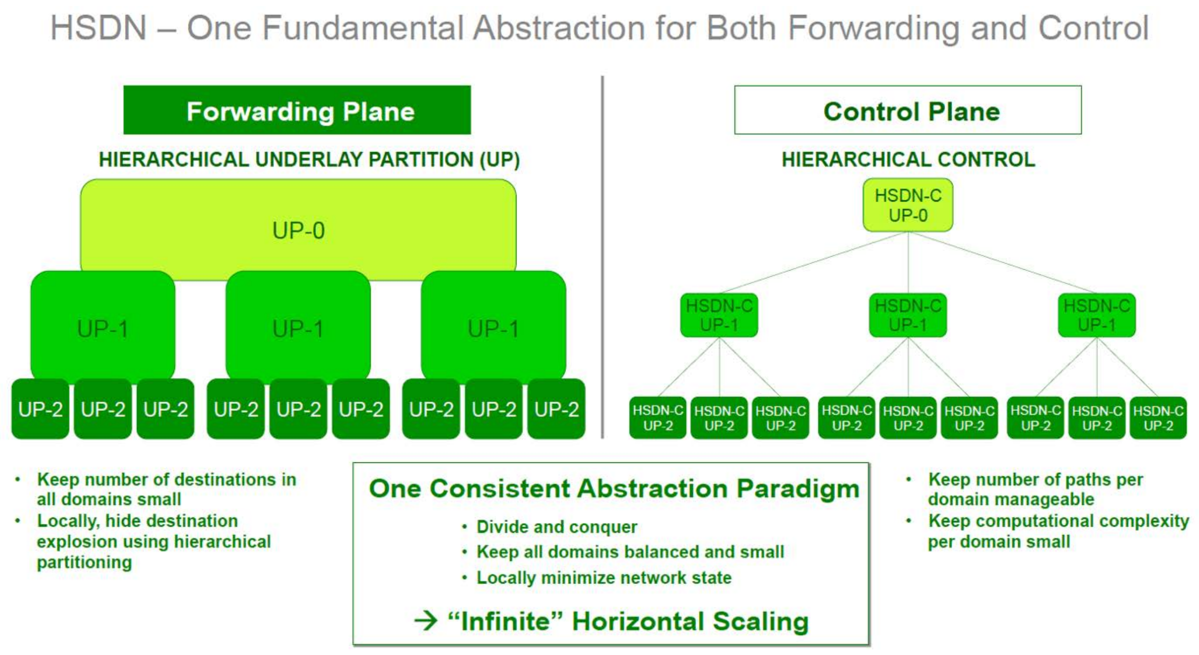
- Scale at low-cost, use commodity HW with relatively small FIBs and LIBs in all network nodes
- Achieve high resource utilization supporting efficiently ECMP and any-to-any, server-to-server Traffic Engineering
- Scale at low operational and computational complexity, locally minimizing complexity and network state, without information loss
- Scale while improving cloud elasticity and service velocity, overcoming current challenges of NFV scalability and VM/NFV mobility

16.3.3 Microsoft H-SDN Architecture

H-SDN is a framework to decompose many complex hyper-scale problems into more manageable ones:

- Forwarding
- Control
- Traffic Engineering
- Operation and Management
- Overlay

16.3.3.1 Forwarding


HSDN – One Fundamental Abstraction for Both Forwarding and Control
Forwarding Plane
HIERARCHICAL UNDERLAY PARTITION (UP)
UP-0
UP-1
UP-2
Control Plane
HIERARCHICAL CONTROL
HSDN-C UP-0
HSDN-C UP-1
HSDN-C UP-2
One Consistent Abstraction Paradigm
→ "Infinite" Horizontal Scaling

- Keep number of destinations in all domains small
- Locally, hide destination explosion using hierarchical partitioning

- Divide and conquer
- Keep all domains balanced and small
- Locally minimize network state

- Keep number of paths per domain manageable
- Keep computational complexity per domain small

Figure 16-4 – H-SDN Forwarding [127]

H-SDN tries to resolve the issues presented in the previous paragraph applying the *divide and conquer* principle to the DCN and the DCI/WAN, organizing the DCN in a hierarchically partitioned structure.

This design is based on three design steps:

1. Identify the DCN topologies that need to be connected through DCI (see Figure 16-5)
2. Define a hierarchical Underlay partitioning (see Figure 16-6)
3. Assign roles and partitions (see Figure 16-7)

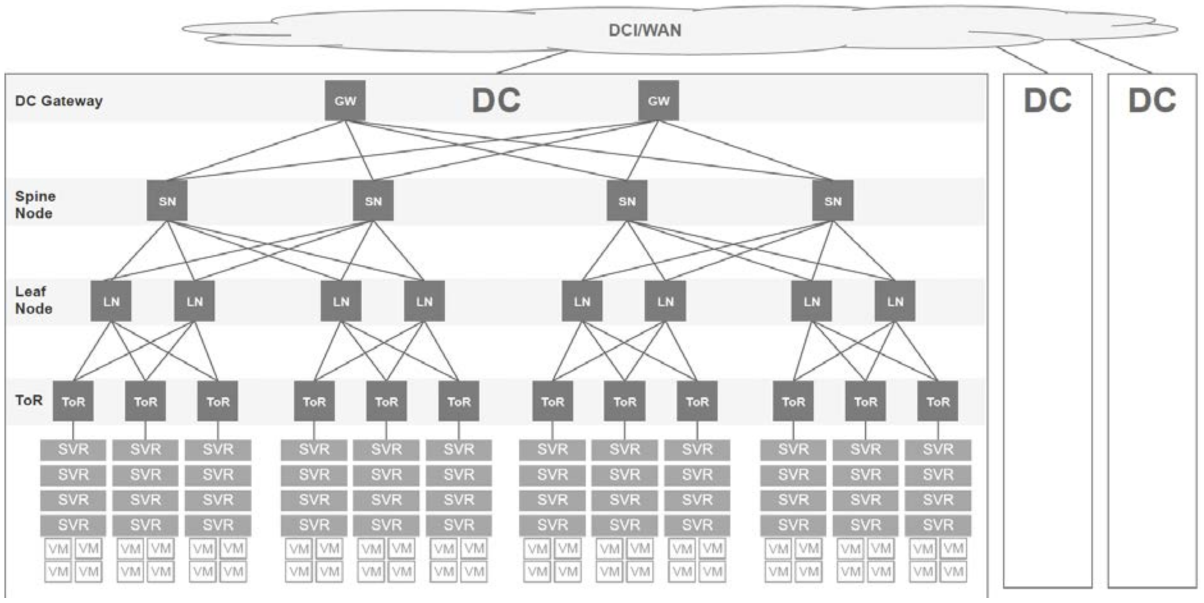


Figure 16-5 – H-SDN design step 1 [120]

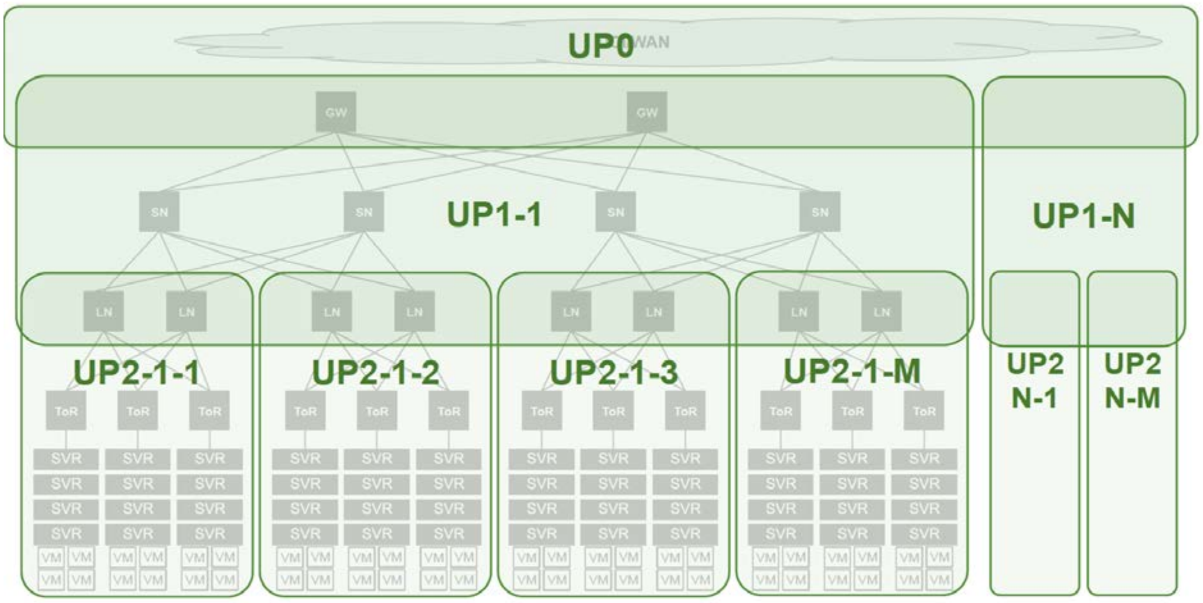


Figure 16-6 – H-SDN design Step 2 [120]

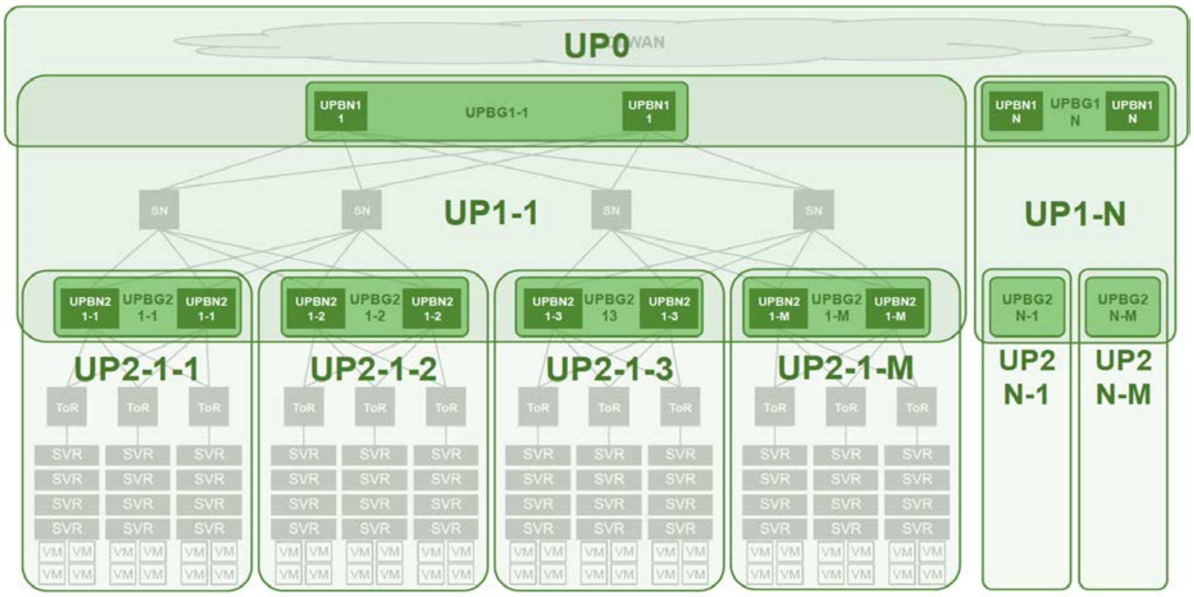


Figure 16-7 – H-SDN design Step 3 [120]

Once the design is completed, the MPLS label can be constructed by stacking all path labels (one per level of underlay partitioning) plus one VN label(see Figure 16-8)

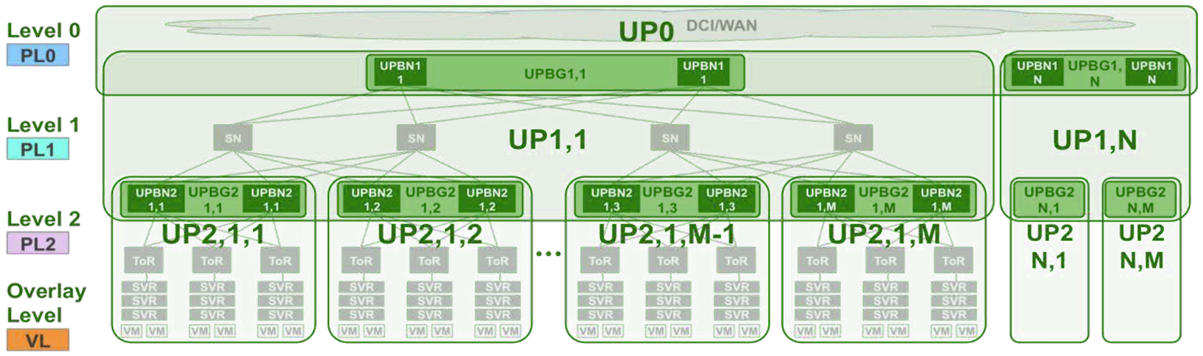


Figure 16-8 – UPBNs, UPBGs and HDSN label stack [120]

16.3.3.2 Control plane

The HSDN Controller (HSDN-C) is horizontally scalable, implemented as a set of local partition controllers (HSDN-C-UP) following the HDSN hierarchy. Each HDSN-C-UP can operate independently within the partition, configuring the LFIBs in the network nodes in the corresponding UP. The individual UP controllers can exchange the labels and construct the label stacks. In HSDN the labels are static and configuration updates are needed only when the physical topology changes or endpoints are added or permanently removed, and thus they are not too frequent. The critical tasks are monitoring network state and assigning traffic to the most suitable paths, thus steering traffic away from those paths that are experiencing failures or congestion.

Each HDSN-C-UP at the lowest level of the hierarchy is also in charge of providing the label stacks to the server's NICs in the corresponding partition. For this purpose, several label servers (that may also be arranged in a hierarchy) map between IP addresses and label stacks.

The hierarchical arrangement of the HSDN-C-UP does not simply allow to scale the "global" SDN controller but also inherently decomposes the SDN control plane in individual UP control planes largely independent. Redundancy is superimposed to the structure, with each HSDN-C-UP shadowing controllers in other UPs.

The HSDN-C-UP may also oversee Traffic Engineering and can also be built with a hybrid approach, in which a distributed routing or label distribution protocol is used to distribute the HSDN labels. HDSN supports both controller-centric SDN approach and traditional distributed routing/label distribution protocol approach: this capability can be useful during the migration from legacy approaches to full SDN.

16.4 Facebook SDN

This paragraph briefly presents some of the outcomes of a conference hosted by Facebook where FB and its partners describe the challenges and the opportunities presented by a modern, holistic SDN approach [128]. The basic concepts, and principles, we want to highlight in this paragraph is how Facebook and its partners plan to disaggregate the network architecture, unleashing the full power of a dynamic infrastructure, software defined at any layer. Quoting from the presentation "*At Facebook, we build our datacenters with fully open and disaggregated hardware. This allows us to replace the hardware or the software as soon as better technology becomes available. Because of this, we see compute, storage, and networking gains that scale with our business*" [128].

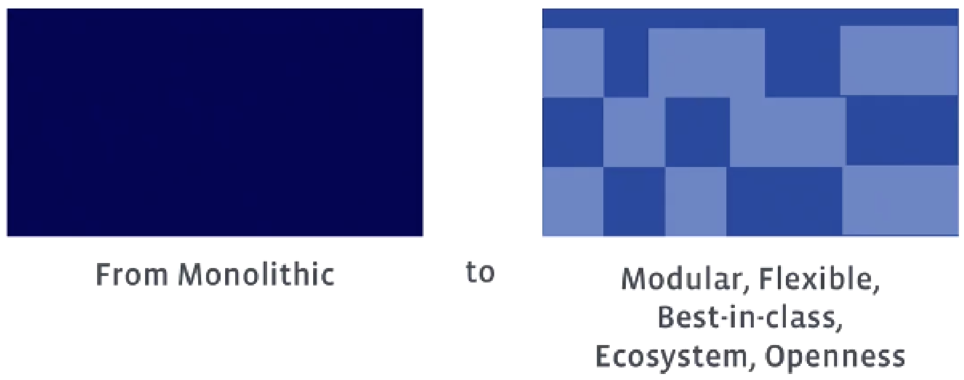


Figure 16-9 – Disaggregate the network approach [129]

The principles behind the disaggregate approach could be summarized in:

1. Centralize the control (Software defined control plane)
2. Openness (open API, open systems, full compatibility)
3. Avoid vendor lock-in
4. Abstract the network
5. Program the Data/Forwarding Plane (*Software is eating the network)*
6. Provide full, end-to-end visibility

The first three principles are shared with Classic SDN (See Chapter 7), the last three are quite new, even though they were already implicit in some SDN architectures. Programming the Data plane, or the forwarding layer, was not part of the original SDN mandate: the original idea of SDN was to abstract and separate the control plane from the forwarding plane and centralize its control. Through this paradigm shift, FB and their partners move the network programmability one step forward, not only by abstracting and centralizing the control plane, but also dynamically defining the data plane behavior by programming it.

16.4.1 Abstract the network

The need of an abstracted model comes from the observation that when there is a lack of an abstracted process describing how a task could be performed, then this task could be completed only using dedicated systems built on-purpose. The typical example is what happened with Computer Graphics: until the definition of a reliable abstracted model (i.e. OpenGL, ActiveX) and the development of dedicated DSPs (GPUs), advanced computer graphics could only be performed by dedicated systems like SGI, IRIS, etc.

Applying this principle to networking, if we can identify a reliable abstraction model describing how a Protocol Data Unit (frame, packet, segment) is handled by the network, and we can write code using this model and compile it on dedicated DSP (or on a general-purpose CPU with a specific instruction set), then we are able to program the forwarding plane.

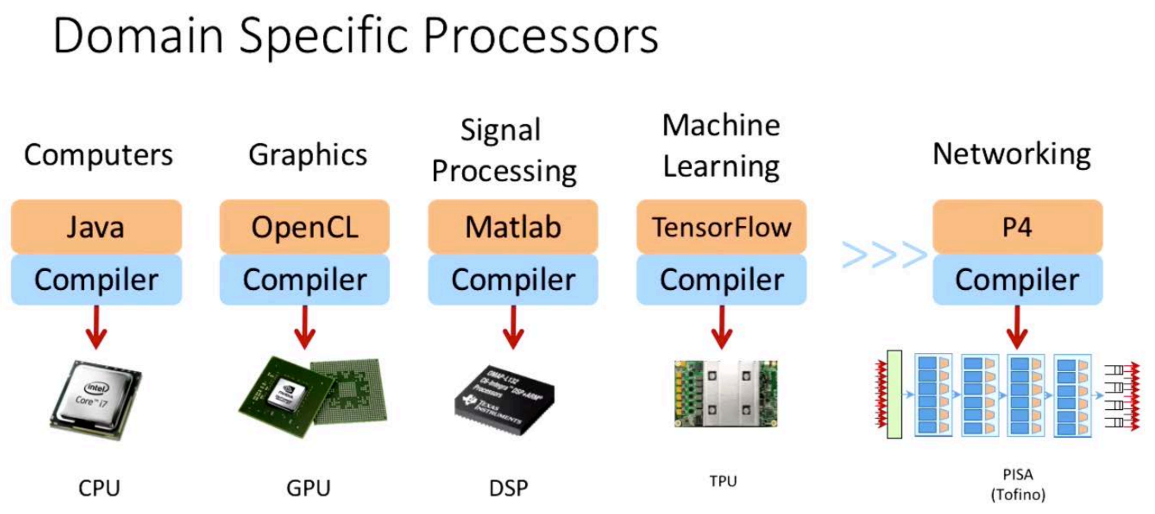


Figure 16-10 – Path to the network abstraction [129]

To enable this process, one needs to define

- An abstraction model, a high-level language able to use this abstraction, and its compiler
- An infrastructure able to run the code

With the right model for data parallelism and the basic underlying processing primitives, using domain specific language and domain specific tools [129] the conventional wisdom that *dedicated systems process packets faster than programmable processors* is not true anymore: as an example the Fixed Function Broadcom Tomahawk delivers 3.2Tbps, the programmable Cavium Xpliant achieves the same throughput and Barefoot Tofino can go up to 6.2Tbps

16.4.2 Forwarding plane programmability

The Protocol Independent Switch Architecture (PISA) is a general, high level architecture model, bringing in networking an abstraction layer like RISC is for computing, providing a simple and very fast pipeline.

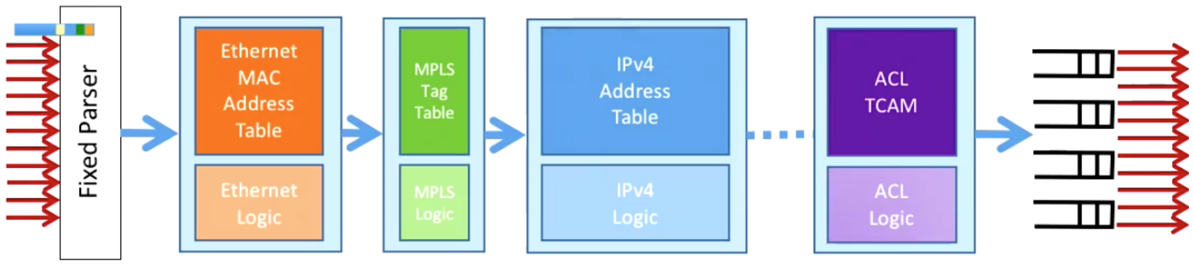


Figure 16-11 – Fixed switch model [129]

The goal of PISA is to establish an independent, efficient, reliable, architecture model (abstraction) that could be used to program the network behavior of inter-networking devices. In the PISA model, and in the SDN approach, the difference between L2, L3, L4 load balancer blurs: by using P4 programs, compiled in the PISA switch, it will be possible to granularly define the role of the device, establishing per-packet rules

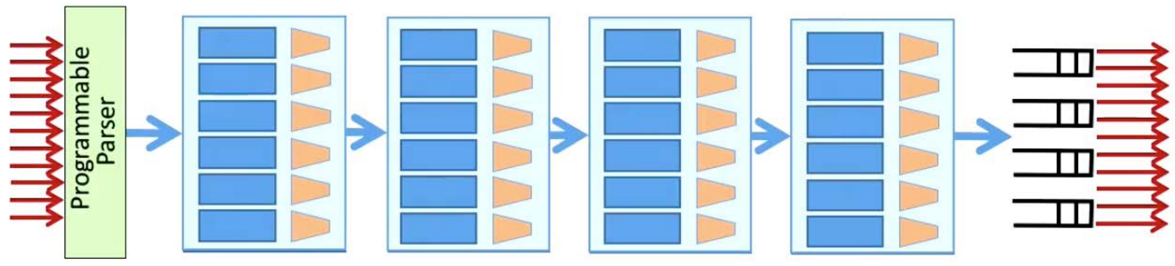


Figure 16-12 – Protocol Independent Switch Architecture [129]

Using PISA and P4 programs, it is possible to embed in the data plane network functions such, for instance, Network Visibility (In Band Network Telemetry) and L4 Load Balancing.

16.4.2.1 Network visibility (In-band network telemetry)

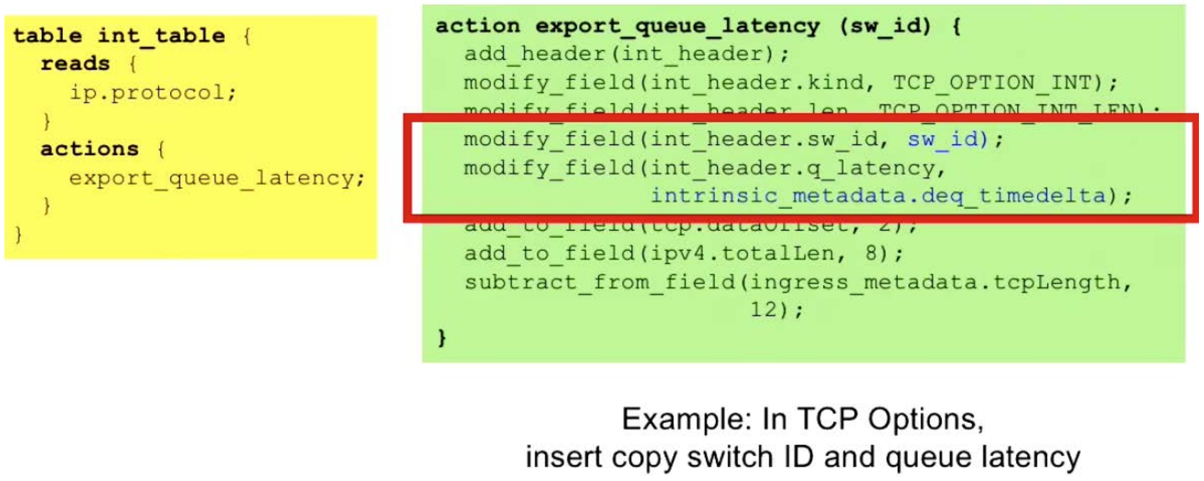
If one would compare the toolset that a storage, a system and a network engineer must use to troubleshoot the systems they manage, the network engineer would always be the one relying mostly on his own expertise and knowledge: during a support call he would be basically stuck with relatively primitive tools like ping, traceroute and NetFlow at the best. However, the questions that he would need to answer would be basically

- Which path did the packet take?
- Which rules did the packet follow?

- How long did it queue at each switch?
- With whom did the packet share the queue?

All those questions would be easily answered if the network infrastructure was able to program the packet processing at each device to provide the required information, performing what is called in-band network telemetry (IBT).

IF the data plane was programmable, though, IBT would be just a piece of code written in P4 running on a device (see Figure 16-13)



```
table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

```
action export_queue_latency (sw_id) {
  add_header(int_header);
  modify_field(int_header.kind, TCP_OPTION_INT);
  modify_field(int_header.len, TCP_OPTION_INT_LEN);
  modify_field(int_header.sw_id, sw_id);
  modify_field(int_header.q_latency,
               intrinsic_metadata.deq_timedelta);
  add_to_field(tcp.dataOffset, 2);
  add_to_field(ipv4.totalLen, 8);
  subtract_from_field(ingress_metadata.tcpLength,
                      12);
}
```

Example: In TCP Options,
insert copy switch ID and queue latency

Figure 16-13 – Example of IBT program written in P4 [129]

16.4.2.2 L4 Load Balancing

The authors in [130] show that a single modern switching ASIC can replace up to hundreds of software load balancer (SLB), potentially reducing the cost of load balancing by over two orders of magnitude.

They use switching ASICs to implement load balancers faster than before, and they built a system, called SilkRoad, defined in 400 lines of P4 code. When SilkRoad is compiled to a state-of-the-art switching ASIC, it can load-balance ten million connections simultaneously at line rate. The authors ran a simulation on FB Datacenter deploying SilkRoad on all the ToR switches. The simulation showed [131] that SilkRoad was able to:

- Sustain a line rate of multi-Tbps.
- ensure persistent session under frequent DIP pool updates.
- Provide 100-1000x savings in power and capital cost.

16.4.3 FB Disaggregated datacenter

The principles outlined in the previous paragraphs are implemented by Facebook in its SDN stack, which is based on the following components (see Figure 16-14):

- Datacenter: FBOSS, Wedge, and Backpack are the hardware and software that power the open switches running in the datacenter, a stepping stone on the disaggregated network.
- Backbone: Express Backbone and Open/R build the SD-WAN, a dedicated private backbone network using centralized traffic engineering in a global controller coupled with a new distributed routing system.
- Cross-domain automation/life cycle: Robotron handles every part of the network life cycle across all network domains, from datacenter to backbone and edge.
- End-to-end flow monitoring provided by fbflow
- Edge: Edge Fabric steers egress user traffic from FB PoPs around the world.

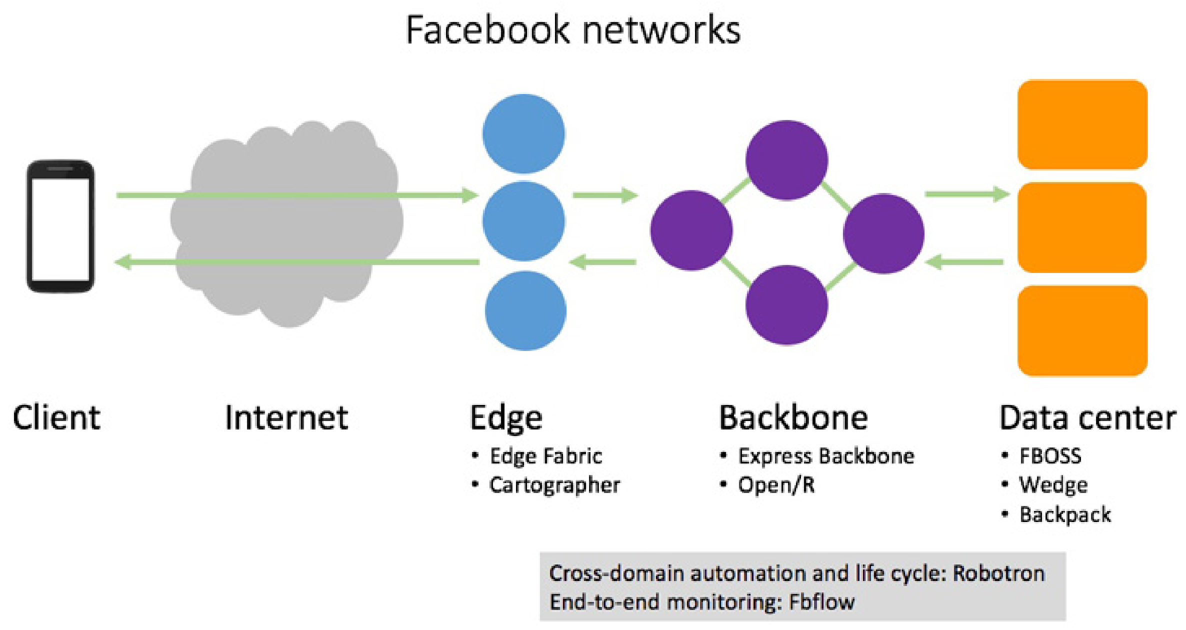


Figure 16-14 – Facebook software defined network components

16.5 SD-WAN

16.5.1 Introduction

Figure 16-15 shows the main challenges that Enterprise are facing regarding their WAN [132]

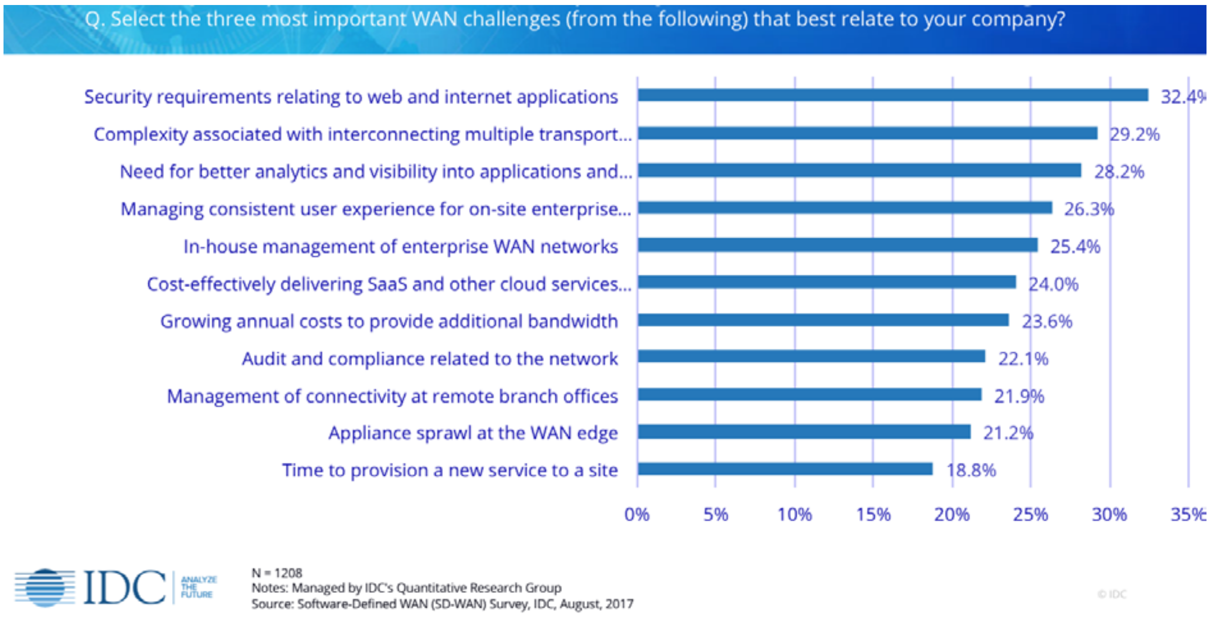


Figure 16-15 – WAN Challenges for the Enterprise [132]

The challenges presented above could be summarized in the following pain points:

- Cloud-unfriendly architecture: Enterprise WAN (EWAN) architecture historically is based on a hub and spoke topology, which enables centralized dataflows. However, Cloud access is, by its very nature, not centralized, and a hub and spoke topology poses some constraints and bottlenecks
- Complexity: EWAN is complex to manage, onerous to configure, maintain and transform
- Costly and (relatively) inefficient: hub and spoke topologies usually require the Internet/public SaaS traffic to be backhauled to the main datacenter where security policies are typically enforced by on-prem security kits. Moreover, with the current trend in bandwidth grow (30% yearly), expansion costs become quickly prohibitive

16.5.2 Software-Defined WAN

The issues presented in the previous paragraph could be addresses by adopting in the EWAN environment a software defined approach, so moving from WAN to Software-Defined WAN.

The main SD-WAN elements are:

- Leverage hybrid networks
- Centralized, application- based policy controller

- Application visibility and control (AVC)
- Network Telemetry (in band or out of band) per flow, application and user aware
- Use of a software overlay abstracting and securing the underlying networks (being them MPLS, LTE, plain Internet access, etc.)
- Dynamic path selection to optimize the WAN based on application requirements.

The typical use cases for SDWAN are simplification, cost reduction, improve application availability and visibility, reducing dependence on legacy MPLS (which is usually connected to a cost reduction), accelerating WAN deployment, and finally, improve public cloud access (IaaS, SaaS and PaaS).

16.5.3 SD-WAN as an enabler for the Cloud

According to Enterprise Management Associates (EMA) [133]. 44% of the enterprise network traffic traces to public cloud, and the hybrid/multi cloud architecture is one of the main drivers of data-center transformation.

This change in the traffic pattern is driving the provisioning of multiple connectivity options for the branches: 55% of the enterprise connect the branches directly to the cloud.

Branches connectivity is shifting from a traditional Hub & Spoke topology to a decentralized solution, with the branches becoming more “intelligent“ and connected to the Corporate networks, the cloud, and the external networks through a fabric of different links managed by a SD-WAN Controller (see Figure 16-16). Essentially, the branches themselves are becoming “Software-defined”, connected to the Software-Defined WAN where a SD-WAN controller:

- enforces the intent-based security policies on a flow-basis
- authenticates/validates endpoints (being them corporate, BYOD, mobiles, etc.)
- provides application visibility and control (AVC) capabilities, and advanced Traffic Engineering techniques
- decides which link should be used for which type of traffic (i.e. local Internet for Best effort, MPLS breakout to the CNF for connection to the private datacenter/public cloud)
- provides network segmentation end-to-end within and across IaaS environments

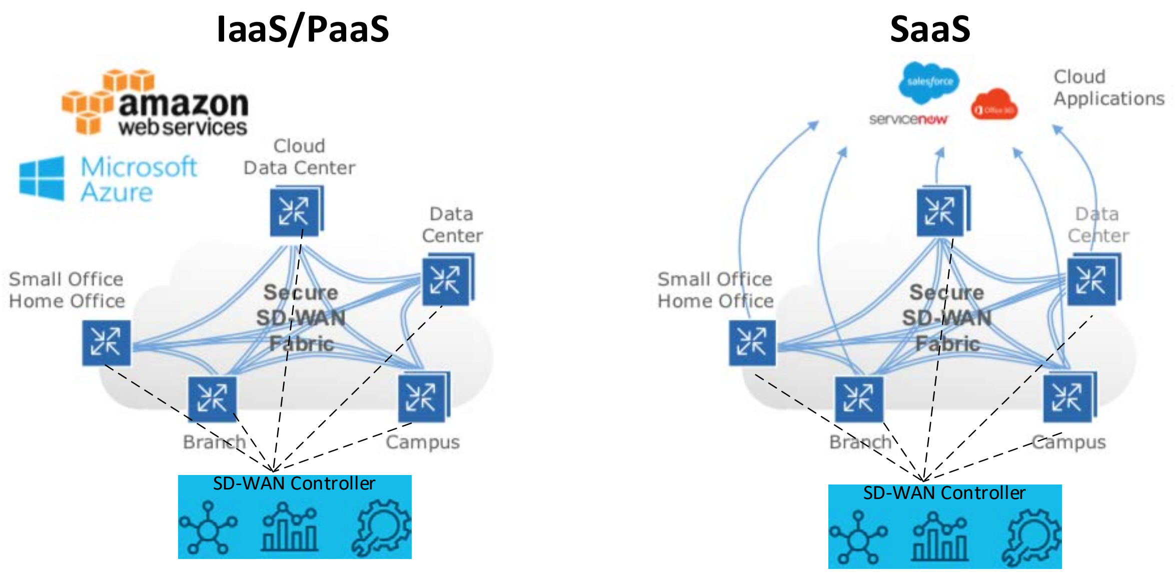


Figure 16-16 – SD-WAN as enabler for Cloud access

16.5.4 The future of SD-WAN

Some of the advances in SD-WAN are obviously directly connected to the development of the SDN world:

- intent-based networking
- standardization of the NBI of the controller,
- orchestration, self-provisioning and auto-sizing networks
- machine learning and AI triggered events
- Hierarchical SDN
- NFV orchestration and cooperation
- Data-plane programming

Specific to the Enterprise SD-WAN, though, there is a transformation of the branch: from a static branch to a SD-Branch containing multiple links controlled by the SD-WAN controller that, using NFV and SFC, will steer the traffic to the proper target, in a completed distributed multi-homed environment. Essentially, the control will be centralized, but the topologies will be distributed. The SD-WAN controller will be able to decide which link to use, for which user and application, and if needed apply a chain of network functions to inspect the traffic, validate it, enforcing local policies based on high-level intents.

This change will have impacts at different levels, it is worth noting that network security will move from a Castle-type defense (perimeter security) to a Casino-approach (distributed security, sensors based, with centralized control).

16.6 Stitching all together: a future Enterprise Software-Defined Network

In a rapidly and constantly evolving environment, dynamic in its very nature, it is quite hard to define how the Enterprise Datacenter network of the future will look like; however, even if the technical solutions will probably change, most of the requirements, trends and principles discussed in Section I and II of this paper will still be valid.

This paragraph, considering the points introduced previously in this chapter, and the overall discussion of this paper, briefly presents a possible future of the Enterprise Network, introducing a mesh topology and explaining its main building blocks.

16.6.1 Evolutions of the Enterprise SDN

As already discussed, SDN is another iteration of the patterns seen in several domains of the IT infrastructure (storage, graphics, compute, etc.), and, by looking at those domains, one could predict that software definition, virtualization, simplification will continue to grow, to eventually provide a dynamic (nearly) self-managed infrastructure able to change, adapt and, to certain extent, evolve.

As shown in Figure 16-17, the move to public cloud, and, in the cloud, the shift from IaaS to SaaS/PaaS consumption models, is another growing trend, but in highly regulated markets (like Finance for instance) some services/data will always require the Enterprise to directly/indirectly manage the infrastructure providing them.

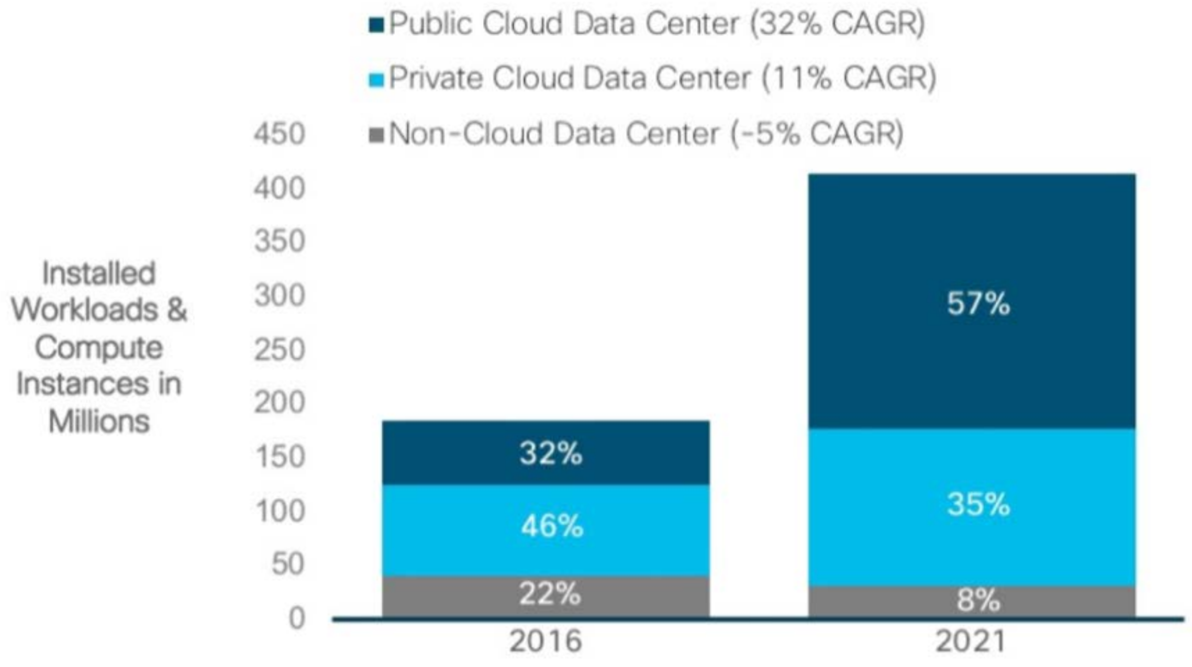


Figure 16-17 – Trend of Compute instances distribution in Private vs public vs non-cloud – source: Cisco© Global Cloud index 2016-2021

This implies that the Enterprise-owned datacenters (at least the virtual ones) are not going to disappear soon: their role is already changing, and will change even more, but the need of a privately/directly owned infrastructure is going to stay. This fact will trigger the need, from the network orchestration perspective, to interact with different DCN models, providing a common platform to manage/monitor/provision datacenter services.

Inside the DCN, the different SDN architectures (pure overlay, hardware or software based, classical SDN, API-based, etc.) will still coexist, but the push to software-define control, management and data planes will continue to grow, and with them the need to virtualize network and network functions will be even higher. The software definition paradigm will be applied not only to the control and management plane but also to the forward/data plane, as explained in 16.4. It is impossible to say which solution will "win" in the DCN market, however network virtualization, software-definition, end-to-end application visibility and control, policy abstraction using intents, interoperability, openness, agility and elasticity will become even more important in the future networks.

The Enterprise WAN is also changing, and this triggers a design change in the Enterprise Global network: from a hub-and-spoke topology, the Enterprise network is moving to hybrid mesh topologies, where Software-defined branches will be connected to cloud providers, Internet, and Service Access Exchanges interconnecting datacenters, 3rd parties, etc.

16.6.2 A possible topology for the future Enterprise DCN

The Future Enterprise DCN shown in Figure 16-19, will be probably based on the following building blocks:

- Physical and/or virtual Software-defined datacenters (typically, at least two per region)
- Software-Defined Service Access Exchanges (typically, at least two per region)
- A Software-defined WAN connecting SD-branches and SD-SAEs
- Intent-based Policies integrating SD-WAN and SD-DCNs
- Enterprise Network Orchestrator

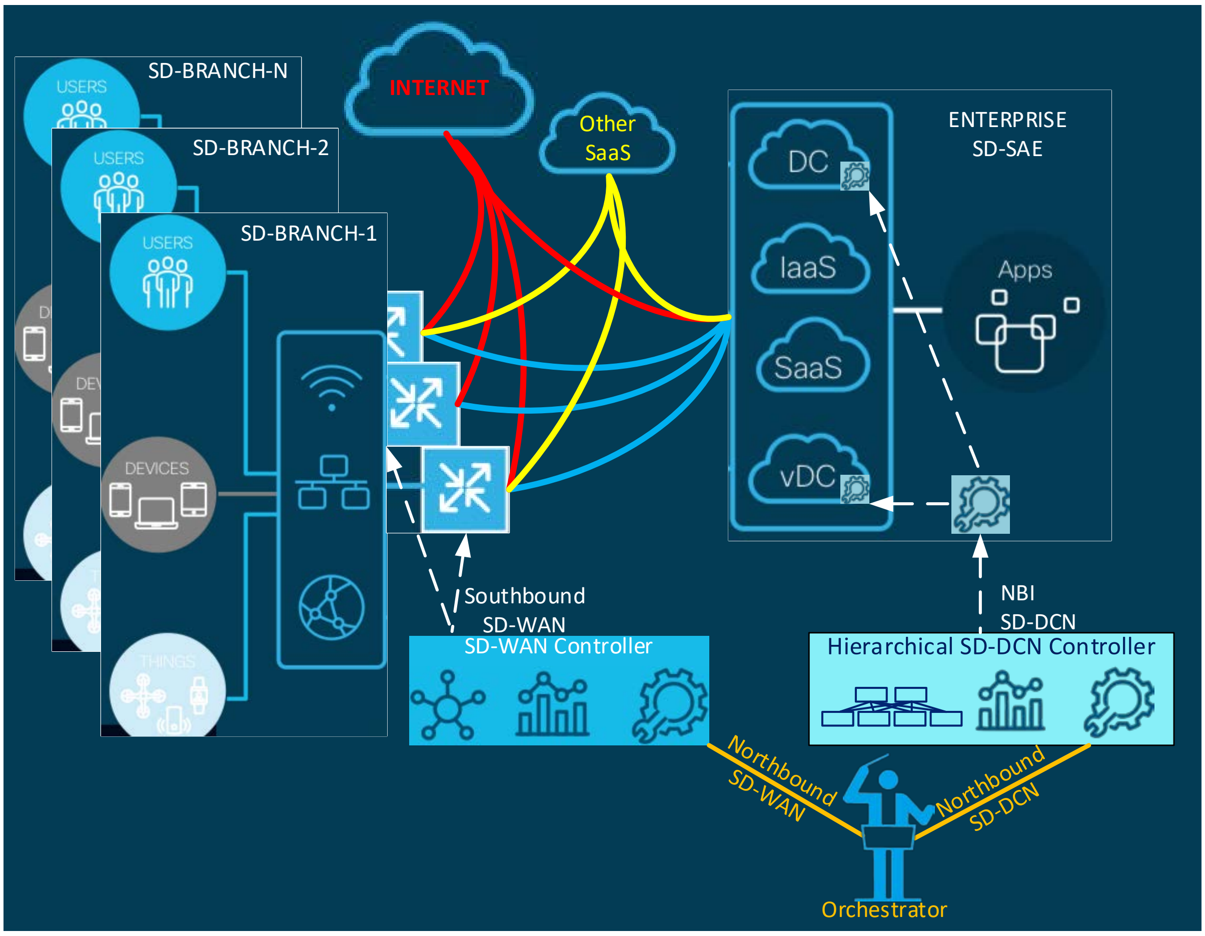


Figure 16-18 – A (possible) topology of the Enterprise Software-defined Network

16.6.3 SD-DCNs and SD-SAE

As described in Chapters 11,12, 13 and 14, regional Software-defined datacenter networks (SD-DCNs) could be managed by local SDN Controllers which can be logically connected together through a higher level regional SDN controller. The definition itself of DCN is expanding: some network manufacturers are already extending their Software-defined DCN to the public cloud, with the same SDN controller managing both physical and virtual leaf switches regardless of their location.

An example of this approach, adopted in Cisco ACI 3.2+ (released by Q2/2018) is shown in Figure 16-19, where a multi-site ACI appliance is controlling one private datacenter and three virtual DCNs located in Google Cloud, Azure and AWS.

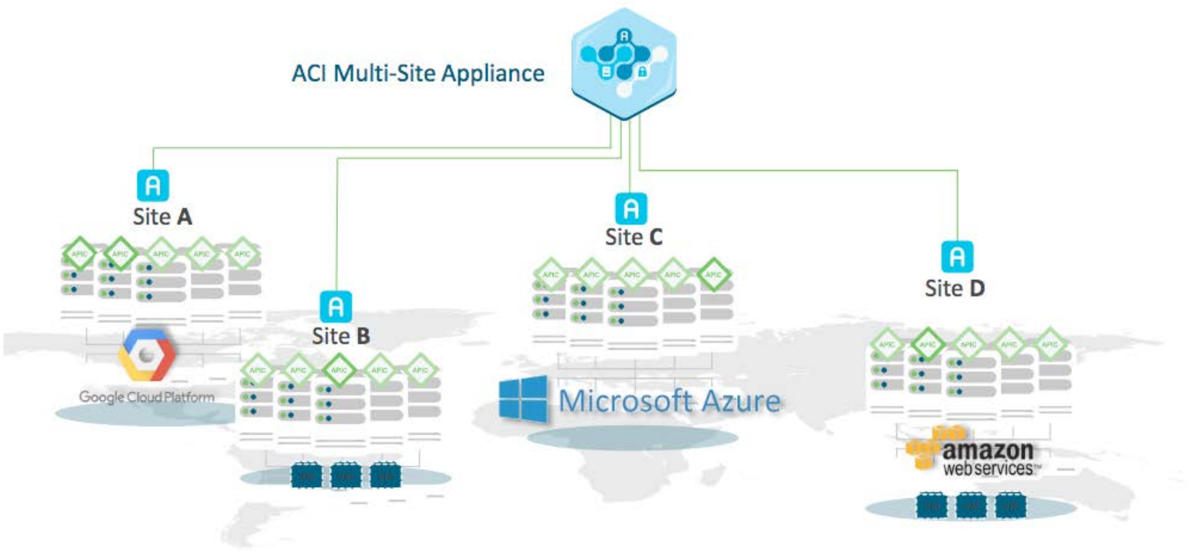


Figure 16-19 – Multi-site DCN with Cisco ACI for private and public DCNs

This solution, or its equivalent from other manufacturers, provides some insights on the requirements for the next generation software-defined datacenters:

- Consistent Network and Policy across private, public and hybrid clouds
- Seamless Workload Migration
- Single Point of Orchestration
- Secure Automated Connectivity

The Secure Access Exchange itself will be software-defined and could host a higher level SDN controller (see Chapter 10). The SD-SAE will interconnect private resources, third parties, shared infrastructures and the SD-branches coming from the SD-WAN links. Figure 16-20 shows an example of a SD-SAE using the Equinix Cloud Neutral Facilities solution.

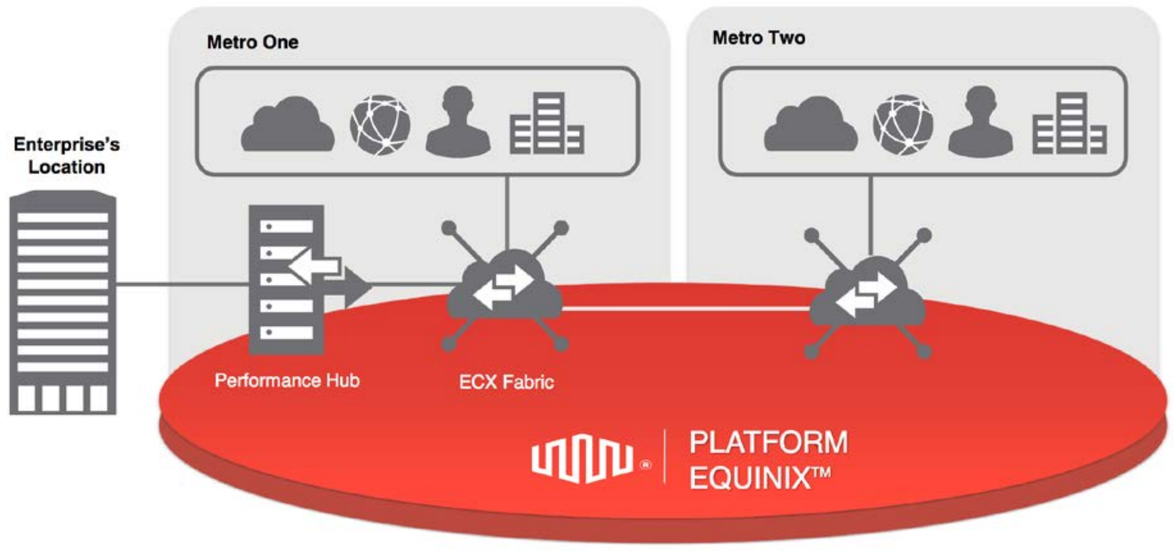


Figure 16-20 – Software Defined Cloud Connect using Equinix Platform (© Equinix, 2018)

16.6.4 SD-WAN

The (overlay) network of Software-defined branches is connected to hybrid links (private MPLS, Internet, LTE, etc.) through a SD-CPE where different VNFs are interconnected to one (or more) NFC.

Figure 16-21 details how the Software-defined branches connect to the WAN: every branch connects to a SD-CPE providing multi-link connectivity but also different network functions (i.e. security kit, multi-link capability, traffic shaping, application visibility and control). The SD-CPE is controlled by a SD-WAN controller able to define the policies for the different regional branches, providing application visibility and control for the different flows, and enforcing Traffic Engineering per branch and class of service.

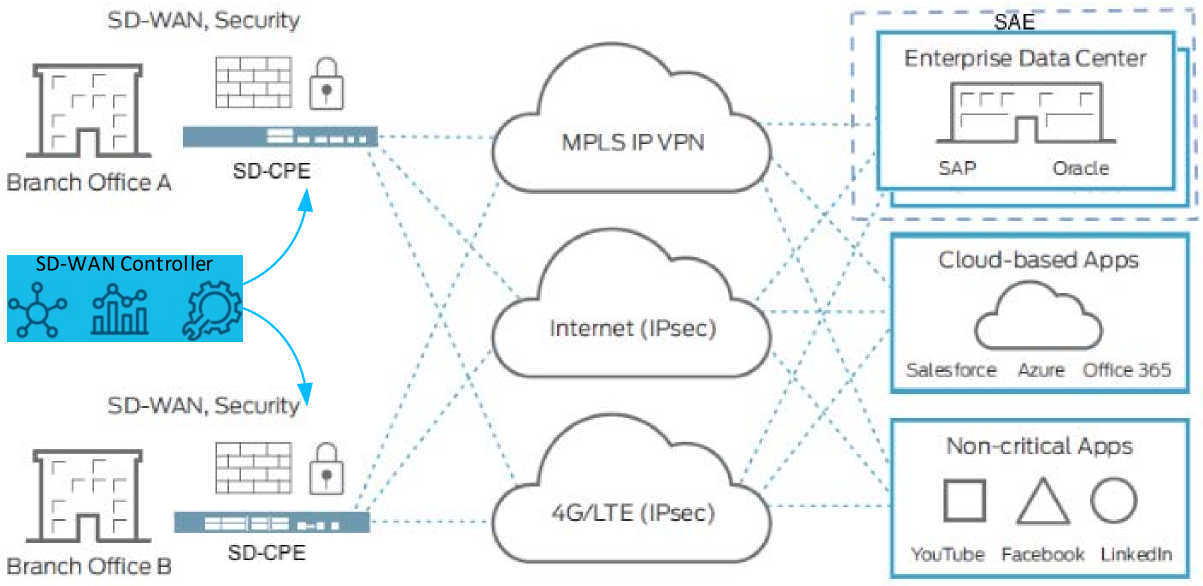


Figure 16-21 – Software-defined WAN and SAE

16.6.5 Integrate SD-WAN and SD-DCN policies

SD-DCN and SD-WAN are currently two disjointed SD domains: this might lead to unwanted segregation and inconsistencies between the WAN user policies and the DCN access policies (see Figure 16-22). The ability to control and monitor end-to-end traffic (in band network telemetry) and to monitor/maintain/update end-to-end flows and policies require an interaction between the SD-WAN controller and the distributed SD-DCN.

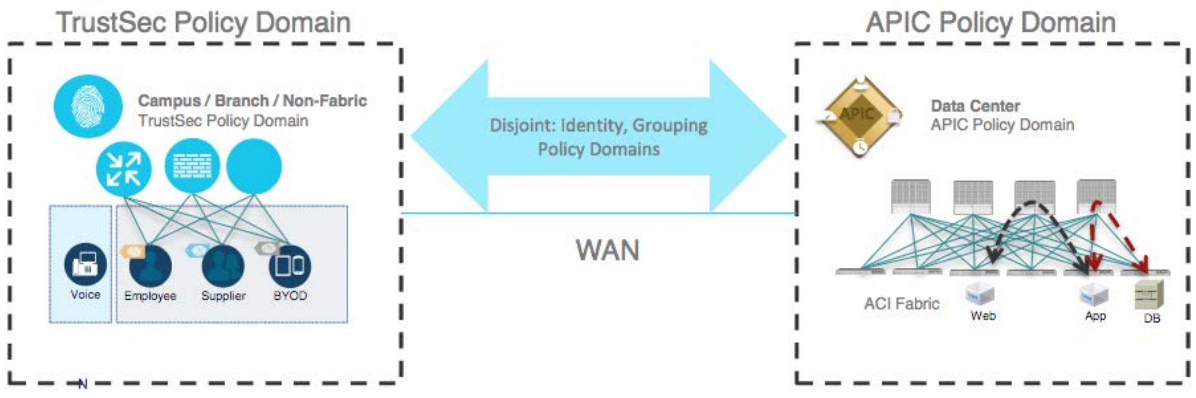


Figure 16-22 - Disjoint Polices between WAN and DCN (©Cisco Systems, Inc.)

This requirement could be implemented in different ways, and up to different extents. Currently the market seems to offer mainly routing, network-level integration (i.e. through MP-BGP). However, some manufacturers are starting to work on integrations at the NBI level and/or at the application layer (i.e. Identity and management).

One example of this integration between SD-DCN and SD-WAN is offered by Cisco: their Identity and Services Engine (ISE) could improve the consistency between SD-WAN and SD-DCN policies (Figure 16-23)

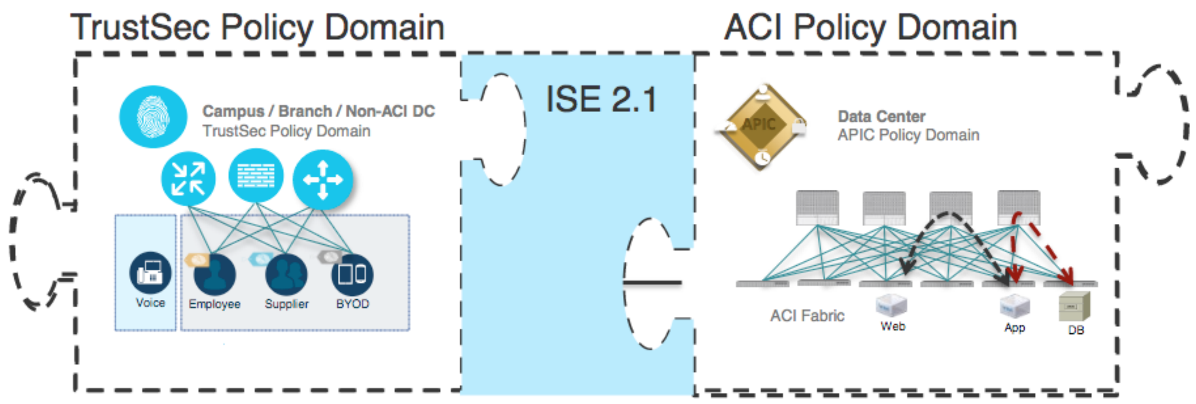


Figure 16-23 Cisco ISE as the glue between SD-WAN and SD-DCN (©Cisco Systems, Inc.)

In this scenario, the User Authentication/Authorization processes are managed by the ISE engine, which, depending on the Authentication/Authorization policies, pushes the corresponding configuration on the Network Access Device (NAD). The NAD is then responsible of enforcing the access policies, providing user segmentation at the access layer, and, finally, NADs are also in charge of marking the data flows using, for instance, Cisco TrustSec © technology. The marked flows can then be applied to define policies at the DCN level using, for instance, constructs inside the ACI contracts and Endpoint groups.

16.6.6 Enterprise Network Orchestrator

The Enterprise Network Orchestrator manages the Software Defined Datacenter (Storage, compute, network, and all the other resources) and/or to provide a common NBI to other applications like a SD-Datacenter management system, a Service Portal (i.e. ServiceNow), etc.

The orchestrator itself might support multiple “southbound” protocols from/to the SD-WAN, SD-DCN and SD-SAEs, and multiple northbound protocols to integrate the different Datacenter/service management solutions. The orchestrator will basically provide an interface to one or more systems, where the datacenter operators will decide/ react/ implement changes to the infrastructure.

With the massive advent of AI, neural networks and the application of predictive models, the orchestrator could be driven, at least partly, from one or more AIs, and this will bring us to the frontier of this paper: the self-managed software defined datacenter.

16.6.7 The frontier: Self-managed software-defined datacenter

As shown in this chapter, the demand for a generalized, consistent, standardized approach to software-defined infrastructures (“*Software is eating the world”* [134]) is growing fast, and in the medium/short term the trend will continue to add (artificial) intelligence to the datacenter orchestrator(s), which eventually will be able to predict the workload requests, place them in the optimal location, and correspondingly interact with the specialized controllers to configure the domain they manage.

In this fully dynamic, self-managed and automated scenario, the datacenter network, as any other part of the IT infrastructure, will be completely software-defined and its multiple SDN controllers will interact via their NBI with the orchestrator(s) providing the ability to dynamically define the network plane.

For the network infrastructure to be fully dynamic, the barriers between different domains (i.e. DCN, WAN, branches, CDN, etc.) will need to be broken, and the SDN controllers of different domains will need to converge, or at least to interact with one another. Soon, probably, an orchestrator (i.e. an expert system based on neural networks and/or other AI or autonomous processes) will be able to proactively manage the entire datacenter infrastructure, allocating resources, moving workloads across different domains, providing dynamic networks, chaining the corresponding network functions, pushing code to the data plane to customize the packet forwarding (i.e. load balancing, inspecting flows, etc.).

17 Bibliography

[1] **Evan Gilman et al.** *Zero Trust Networks: Building Secure Systems in Untrusted Networks.* s.l. : O'Reilly Media. , 2017, 2017.

[2] **Priestley, Angela.** Work': It's a verb not a noun, so find somewhere else to get it done. [Online] [Cited: 11 25, 2017.] https://womensagenda.com.au/latest/eds-blog/work-it-s-a-verb-not-a-noun-so-find-somewhere-else-to-get-it-done/.

[3] **IDG Enterprise.** *Consumerization of IT in the Enterprise Study .* s.l. : IDG Enterprise, 2014.

[4] **Morris, Kief. .** *Infrastructure as Code .* s.l. : O'Reilly, 2016.

[5] **Loukides, Mike.** *What is DevOps? .* s.l. : O'Reilly Media, 2012.

[6] **NIST.** *Minimum Security Requirements for Federal Information and Information Systems.* s.l. : NIST, 2006.

[7] **European Parliament.** GDPR Portal. [Online] [Cited: 11 26, 2017.] https://www.eugdpr.org.

[8] **PCI Security Standards Council .** PCI SECURITY. *PCI SECURITY.* [Online] [Cited: 11 29, 2017.] https://www.pcisecuritystandards.org/pci_security/.

[9] **John Liebman et al.** *United States Export Controls.* s.l. : Aspen Publishers, 2011.

[10] **Wassenaar Arrangemen Secretariat.** The Wassenaar Arrangement (WA). [Online] [Cited: 11 26, 2017.] http://www.wassenaar.org.

[11] **Paul Baran.** *On Distributed Communication Networks.* 1962 : The Rand Corporation,.

[12] **Leah A Lievrouw, Sonia Livingstone.** Handbook of New Media: Student Edition. s.l. : SAGE, 2005.

[14] **IEEE.** 802.1Q-Rev - 802.1Q Revision - Bridges and Bridged Networks. [Online] [Cited: 12 12, 2017.] http://www.ieee802.org/1/pages/802.1Q-rev.html..

[15] —. 802.1D MAC Bridges. [Online] [Cited: 12 13, 2017.] http://www.ieee802.org/1/pages/802.1D-2003.html..

[16] **NIST.** NIST Cloud Computing Program - NCCP. [Online] [Cited: 12 12, 2017.] https://www.nist.gov/programsprojects/.

[17] **Jaydip Sen .** Security and Privacy Issues in Cloud Computing. *Security and Privacy Issues in Cloud Computing.* [Online] TATA Consulting. [Cited: 12 02, 2017.] https://pdfs.semanticscholar.org/4dc3/70d253020947a8e66b701e12dd0233161229.pdf.

[18] **Maverick Research .** *The Edge Will Eat the Cloud. .* s.l. : Gartner Inc., 2017.

[19] **Christine Currie.** 4 REASONS TO PICK A CARRIER NEUTRAL DATA CENTER. *4 REASONS TO PICK A CARRIER NEUTRAL DATA CENTER.* [Online] 365DataCenters. http://www.365datacenters.com/blog/4-reasons-pick-carrier-neutral-data-center/.

[20] **White, Russ and Denise, Donohue.** *The Art of Network Architecture.* s.l. : Cisco Press, 2016.

[21] **Giaccone, Paolo.** *Switching technologies for datacenters.* Torino : Politecnico di Torino, 2017.

[22] **Liu, Yang and et , al.** *Data Center Networks Topologies, Architectures and Fault-Tolerance.* s.l. : SpringerBriefs in Computer Science, 2013.

[23] **Stanford University.** *Introduction to Clos Networks.* . s.l. : Stanford University Press, 2006.

[24] *A scalable, commodity data center network architecture.* **Al-Fares, M., Loukissas, A., Vahdat, A.:.** 2008. ACM SIGCOMM 2008 Conference on Data Communication,.

[25] *a scalable fault-tolerant layer 2 data center network fabric.* **Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: Portland:.** 4, 2009, ACM SIGCOMM Comput. Commun. Rev., Vol. 39, pp. 39-50.

[26] *VL2: a scalable and flexible data center network.* **Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S.:.** 4, 2009, SIGCOMM Comput. Commun. Rev., Vol. 39, pp. 51-62.

[27] *DCell: a scalable and fault-tolerant network structure for data centers.* **Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.:.** 4, 2008, ACM SIGCOMM Comput. Commun. Rev., Vol. 38, pp. 75-86.

[28] *BCube: a high performance, server-centric network architecture for modular data centers.* **Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.:.** 4, 2009, ACM SIGCOMM Comput. Commun. Rev., Vol. 39, pp. 63-74.

[29] *c-Through: part-time optics in data centers.* **Wang, G., Andersen, D., Kaminsky, M., Papagiannaki, K., Ng, T., Kozuch, M., Ryan, M.:.** 2010, ACM SIGCOMM Computer Communication Review,, Vol. 40, pp. 327-338.

[30] *Helios: a hybrid electrical/ optical switch architecture for modular data centers.* **Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H., Subramanya, V., Fainman, Y., Papen, G., Vahdat, A.:.** 2010, ACM SIGCOMM Computer Communication Review,, Vol. 40, pp. 339-350.

[31] *an optical switching architecture for data center networks with unprecedented flexibility.* **Chen, K., Singla, A., Singh, A., Ramachandran, K., Xu, L., Zhang, Y., Wen, X., Chen, Y.:.** 2012. 9th USENIX Conference on Networked Systems Design and Implementation,.

[32] *Ficonn: using backup port for server interconnection in data centers.* **Li, D., Guo, C., Wu, H., Tan, K., Zhang, Y., Lu, S.:.** 2009. IEEE INFOCOM.

[33] *MDCube: a high performance network structure for modular data center interconnection.* **Wu, H., Lu, G., Li, D., Guo, C., Zhang, Y.:.** Rome : s.n., 2009. 5th International Conference on Emerging Networking Experiments and Technologies,.

[34] *Symbiotic routing in future data centers.* **5th International Conference on Emerging Networking Experiments and Technologies.** 4, 2010, ACM SIGCOMM Comput. Commun. Rev., Vol. 40, pp. 51-62.

[35] **Clos, Charles.** *A Study of Non-Blocking Switching Networks.* s.l. : Bell Laboratories, 1952.

[36] **Krattiger, Lukas , Kapadia, Shyam and Jansen,, David .** *Building Data Centers with VXLAN BGP EVPN - A Cisco NX-OS Perspective.* s.l. : Cisco Press, 2017.

[37] **Cisco Systems.** *Cisco Data Center Spine-and-Leaf Architecture: Design Overview.* s.l. : Cisco Systems, 2017.

[38] **Big Switch Networks.** Core and Pod Data Center Design. *Core and Pod Data Center Design.* [Online] [Cited: 12 02, 2017.] http://go.bigswitch.com/rs/974-WXR-561/images/Core-and-Pod%20Overview.pdf.

[39] **Cisco Systems.** *Data Center Top-of-Rack Architecture Design.* s.l. : Cisco Systems, 2014.

[40] **Amazon AWS .** (AWS re: Invent 2016: Tuesday Night Live with James Hamilton). [Online] [Cited: 12 12, 2017.] https://www.youtube.com/watch?v=AyOAjFNPAbA..

[41] **J. Bagga et al.** Open networking advances with Wedge and FBOSS. s.l. . [Online] [Cited: 12 12, 2017.] https://code.facebook.com/posts/145488969140934/open-networking-advances-with-wedge-and-fboss/.

[42] **Seb Boving et al.** *Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter.* s.l. : Google Inc, 2017.

[43] **Greenberg, Albert et al.** Microsoft Showcases Software Defined Networking Innovation at SIGCOMM. [Online] [Cited: 12 12, 2017.] https://azure.microsoft.com/en-gb/blog/microsoftshowcases-.

[44] **Fang, Luyuan.** Hierarchical SDN to Scale the DC/Cloud to Tens of Millions of Endpoints at Low Cost. [Online] [Cited: 12 12, 2017.] https://www.google.ie/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ve

d=0ahUKEwiaxbmwyILYAhUFCcAKHdF0Di4QFggpMAA&url=https%3A%2F%2Fwww.ietf.org%2Fproceedings%2F91%2Fslides%2Fslides-91-sdnrg-7.pdf&usg=AOvVaw1knh_WG4kQ4K4v0Qpu77qD.

[45] **Lajos, Sárecz. .** *Oracle Cloud Platform: Modern Cloud Infrastructure.* s.l. : Oracle, 2017.

[46] **IETF.** RFC 7365 - Framework for DC Network Virtualization. [Online] [Cited: 09 25, 2017.] https://tools.ietf.org/html/rfc7365.

[47] **Hassan, Syed Farrukh, Rajendra , Chayapathi and Paresh , Shah.** *Network Functions Virtualization (NFV) with a Touch of SDN.* s.l. : Pearson Education, 2016.

[48] **Cisco Systems.** Configuring VSS. [Online] Cisco. [Cited: 09 25, 2017.] https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/vss.pdf.

[49] —. Virtual Switching System (VSS) Q&A. *Virtual Switching System (VSS) Q&A.* [Online] [Cited: 11 27, 2017.] Virtual Switching System (VSS) Q&A.

[50] **IEEE P802.3ad Link Aggregation Task Force.** IEEE P802.3ad Link Aggregation Task Force - Public Area. [Online] [Cited: 11 26, 2017.] http://www.ieee802.org/3/ad/public/index.html.

[51] **Cisco Systems.** Configuring Cisco vPC. [Online] Cisco. [Cited: 09 25, 2017.] https://www.cisco.com/c/en/us/products/collateral/switches/nexus-5000-series-switches/design_guide_c07-625857.html.

[52] —. Cisco FabricPath. [Online] [Cited: 11 26, 2017.] https://www.cisco.com/c/en/us/solutions/data-center-virtualization/fabricpath/index.html.

[53] **IETF Trill WG.** Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS. [Online] [Cited: 11 25, 2017.] https://tools.ietf.org/html/rfc6326.

[54] **IETF.** Virtual eXtensible Local Area Network (VXLAN): A Framework. [Online] [Cited: 11 25, 2017.] https://tools.ietf.org/html/rfc7348.

[55] **IETF - LISP Working Group .** GPE-VPN: Programmable LISP-based Virtual Private Networks - draft-maino-gpe-vpn-00. *GPE-VPN: Programmable LISP-based Virtual Private Networks - draft-maino-gpe-vpn-00.* [Online] [Cited: 11 28, 2017.] https://tools.ietf.org/html/draft-maino-gpe-vpn-00.

[56] **IETF.** Multiprotocol Extensions for BGP-4 - RFC 4760. *Multiprotocol Extensions for BGP-4.* [Online] [Cited: 12 11, 2017.] https://tools.ietf.org/html/rfc4760.

[57] **IETF MPLS Working Group.** Mpls Status Pages. *Mpls Status Pages.* [Online] [Cited: 11 28, 2017.] https://tools.ietf.org/wg/mpls/.

[58] **Cisco Systems.** Overlay Transport Virtualization. *Overlay Transport Virtualization.* [Online] [Cited: 11 28, 2017.] https://www.cisco.com/c/en/us/solutions/data-center-virtualization/overlay-transport-virtualization-otv/index.html.

[59] —. Technical Overview of Virtual Device Contexts. [Online] [Cited: 11 26, 2017.] https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-10-slot-switch/White_Paper_Tech_Overview_Virtual_Device_Contexts.html.

[60] —. *Virtual Route Forwarding Design Guide.* s.l. : Cisco Press, 2008.

[61] **Paul Goransson, Chuck Black, Timothy Culver.** *Software Defined Networks: A Comprehensive Approach. .* s.l. : Morgan Kaufmann. 2016.

[62] **Laboratory, Cambridge Computer.** DCAN - Devolved Control of ATM Networks. [Online] [Cited: 11 26, 2017.] https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/dcan/.

[63] *Open signaling for ATM, internet and mobile networks (OPENSIG'98).* **Andrew T. Campbell, Irene Katzela, Kazuho Mika, John Vicente.** 1, s.l. : ACM SIGCOMM Computer Communication Review, 1999, ACM SIGCOMM Computer Communication Review, Vol. 29 , pp. 97-108 .

[64] *A survey of active network research.* **Tennehouse D., Smith J., Sincoskie W.,Wetherall D., Minden G.** 1, 1997, IEEE Commun Mag., Vol. 35, pp. 80-86.

[65] **University of Washington.** A comparison of IP switching technologies from 3Com, Cascade, and IBM. [Online] 1997. [Cited: 11 26, 2017.] http:// www.cs.washington.edu/ education/ courses/ csep561/ 97sp/ paper1/ paper11. txt..

[66] **Johnson, Johna Till.** MPLS explained. [Online] [Cited: 11 26, 2017.] https://www.networkworld.com/article/2297171/network-security/network-security-mpls-explained.html.

[67] **IETF.** Forwarding and Control Element Separation (ForCES) Protocol Extensions - RFC 7391. [Online] [Cited: 11 26, 2017.] https://tools.ietf.org/html/rfc7391.

[68] *A clean slate 4D approach to network control and management.* **Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, Hui Zhang.** 5, 2005, ACM SIGCOMM Computer Communication Review, Vol. 35, pp. 41-54.

[69] **Stanford University - Clean State Project Team,.** Ethane: A Security Management Architecture. [Online] [Cited: 11 26, 2017.] http://yuba.stanford.edu/ethane/.

[70] **Rigney C., Willens S., Rubens A., Simpson W.** *Remote Authentication Dial In User Service (RADIUS).* s.l. : Internet Engineering Task Force;, 2000.

[71] **Durham D., Boyle J., Cohen R., Herzog S., Rajan R., Sastry A.** *The COPS (Common Open Policy Service) Protocol.* s.l. : Internet Engineering Task Force;, 2000.

[72] **G., Ferro.** Automation and orchestration. Network Computing,. [Online] 2011. [Cited: 11 26, 2017.] http:// www.networkcomputing.com/ private-cloud-tech-center/ automation-and-orchestration/ 231600896..

[73] **Inside, Technology.** The phenomenon of Ipsilon. [Online] [Cited: 09 25, 2017.] https://technologyinside.com/2007/02/08/networks-part-2-the-flowering-and-dying-of-ipsilon/.

[74] **Azodolmolky, Siamak.** *Software Defined Networking with OpenFlow.* s.l. : Packt Publishing, 2017.

[75] **Brand Salisbury.** TCAMs and OpenFlow – What Every SDN Practitioner Must Know. *SDXCentral.* [Online] [Cited: 12 12, 2017.] https://www.sdxcentral.com/articles/contributed/sdn-openflow-tcam-need-to-know/2012/07/.

[76] **Big Switch Networks.** Big Switch Big Cloud Fabric. [Online] Big Switch Networks. [Cited: 09 25, 2017.] http://www.bigswitch.com/products/SDN-Controller.

[77] **Project Floodlight.** http://www.projectfloodlight.org/floodlight/. [Online] [Cited: 11 25, 2017.] http://www.projectfloodlight.org/floodlight/.

[78] **Wikipedia.** Fundamental theorem of software engineering. [Online] [Cited: 11 25, 2017.] https://en.wikipedia.org/wiki/Fundamental_theorem_of_software_engineering.

[79] **Jensen, David.** *Building Data Centers with VXLAN BGP EVPN: A Cisco NX-OS Perspective (Networking Technology).* s.l. : Pearson Education, 2017.

[80] **Pfaff, B. and B. Davie.** RFC 7047 - The Open vSwitch Database Management Protocol. [Online] [Cited: 11 26, 2017.] https://tools.ietf.org/html/rfc7047.

[81] **Jabber open-source community.** XMPP.org. [Online] [Cited: 11 26, 2017.] https://xmpp.org.

[82] **IETF L2VPN Workgroup .** A Network Virtualization Overlay Solution using EVPN. [Online] IETF. [Cited: 11 25, 2017.] https://tools.ietf.org/html/draft-ietf-bess-evpn-overlay-01.

[83] **Juniper Networks.** Contrail Data Sheet. *Juniper Networks.* [Online] [Cited: 12 02, 2017.] https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000521-en.pdf.

[84] —. Learn About VXLAN in Virtualized Data Center Networks. *Learn About VXLAN in Virtualized Data Center Networks.* [Online] [Cited: 12 02, 2017.] https://www.juniper.net/documentation/en_US/learn-about/LA_VXLANinDCs.pdf.

[85] **Portolani, Lucien Avramov. Maurizio.** *The Policy Driven Data Center with ACI: Architecture, Concepts, and Methodology.* s.l. : Cisco Press, 2016.

[86] **Cisco Systems.** isco Application Centric Infrastructure Release 2.3 Design Guide White Paper. *isco Application Centric Infrastructure Release 2.3 Design Guide White Paper.* [Online] [Cited: 12 08, 2017.] https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-737909.html.

[87] **Rajendra Chayapathi, Syed F. Hassan, Paresh Shah.** The Journey to Network Functions Virtualization (NFV) Era. *InformIT.* [Online] [Cited: 12 08, 2017.] www.informit.com/articles/article.aspx?p=2755705&seqNum=2.

[88] **IETF.** Service Function Chaining (SFC) Use Cases. [Online] [Cited: 11 25, 2017.] https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06.

[89] **Open Networking Foundation.** Network Functions Virtualization group. [Online] [Cited: 11 28, 2017.] https://www.opennetworking.org/news-and-events/blog/our-work-with-the-new-network-functions-virtualisation-group/?option=com_wordpress&Itemid=72.

[90] **ETSI.** *NFV White PAper.* 2012.

[91] —. Leading operators create ETSI standards group for network functions virtualization. [Online] [Cited: 11 28, 2017.] http://www.etsi.org/index.php/news-events/news/644-2013-01-isg-nfv-created.

[92] **IETF.** Service Function Chaining (SFC) Architecture. *Service Function Chaining (SFC) Architecture.* [Online] [Cited: 12 08, 2017.] https://tools.ietf.org/html/rfc7665.

[93] —. Service Function Chaining (SFC) Architecture. [Online] [Cited: 11 25, 2017.] https://tools.ietf.org/html/rfc7665.

[94] **IETF .** VNF Pool Orchestration For Automated Resiliency in Service Chains - draft-bernini-nfvrg-vnf-orchestration-02. [Online] [Cited: 12 08, 2017.] https://tools.ietf.org/html/draft-bernini-nfvrg-vnf-orchestration-02.

[95] **SDXCentral.** Which is Better – SDN or NFV? *Which is Better – SDN or NFV?* [Online] [Cited: 12 08, 2017.] https://www.sdxcentral.com/nfv/definitions/which-is-better-sdn-or-nfv/.

[96] **ETSI.** Network Functions Virtualisation. *Network Functions Virtualisation.* [Online] [Cited: 12 10, 2017.] http://www.etsi.org/technologies-clusters/technologies/nfv.

[97] **Cisco Systems.** *Cisco Secure Agile Exchange - Solution Overview.* s.l. : Cisco Live 2017, 2017.

[98] **Anderson, Gunnar.** Secure Agile Exchange Revealed. [Online] [Cited: 11 25, 2017.] https://blogs.cisco.com/cloud/secure-agile-exchange-revealed.

[99] **Gill, Bob.** *Eight Trends Will Shape the Colocation Market in 2016.* s.l. : Gartner Inc., 2016.

[100] **Dell EMC Networking Solutions Engineering.** *Architecting a software-defined datacenter with Big Cloud Fabric and Dell EMC ScaleIO.* s.l. : Dell EMC, 2017.

[101] **Big Switch .** Big Switch Demo Lab. [Online] [Cited: 11 29, 2017.] https://labs.bigswitch.com/users/login.

[102] **Big Switch Networks.** Next-Generation Data Center Networking at the Speed of Virtualization. *Next-Generation Data Center Networking at the Speed of Virtualization.* [Online] Big Switch Networks. [Cited: 12 10, 2017.] https://www.bigswitch.com/bringing-physical-network-automation-and-visibility-to-vmware-virtualization-environments.

[103] **Deepti Chandra.** *This Week: Data Center Deployment with EVPN/VXLAN.* s.l. : Juniper Networks Books, 2017.

[104] **Hanks, Douglas Richard.** *Juniper QFX 5100 Series.* s.l. : O'Reilly, 2015.

[105] **Juniper Networks.** *Solution Guide - Infrastructure as a Service: Contrail and OVSDB.* s.l. : Juniper Networks.

[106] —. Clos IP Fabrics with QFX5100 Switches. *Clos IP Fabrics with QFX5100 Switches.* [Online] [Cited: 12 12, 2017.] https://www.juniper.net/assets/fr/fr/local/pdf/whitepapers/2000565-en.pdf.

[107] —. *Using Contrail with OVSDB in Top-of-Rack Switches.* s.l. : Juniper Networks, 2017.

[108] —. Contrail Service Chaining. *Contrail Manual.* [Online] [Cited: 11 25, 2017.] https://www.juniper.net/documentation/en_US/contrail3.1/topics/task/configuration/service-chaining-vnc.html.

[109] —. Installing and Provisioning VMware vCenter with Contrail. *Juniper Networks.* [Online] [Cited: 12 02, 2017.] https://www.juniper.net/documentation/en_US/contrail3.0/topics/task/configuration/vcenter-integration-vnc.html.

[110] **Cisco Systems.** *Cisco Application Centric Infrastructure Release 2.0 Design Guide.* s.l. : Cisco Press, 2017.

[111] **Tom Edsall .** ACI Aniwhere! [Online] Cisco Systems. [Cited: 11 25, 2017.] https://blogs.cisco.com/news/aci-anywhere.

[112] **IETF .** OpFlex Control Protocol. *OpFlex Control Protocol.* [Online] [Cited: 12 02, 2017.] https://tools.ietf.org/html/draft-smith-opflex-00.

[113] **Cisco Systems.** Multi Protocol BGP . *Multi Protocol BGP .* [Online] [Cited: 12 11, 2017.] https://www.cisco.com/networkers/nw00/pres/3200/3200_c1_Mod5_rev1.pdf.

[114] **IETF Network Working Group .** Layer 2 (L2) LISP Encapsulation Format - draft-smith-lisp-layer2-01. *Layer 2 (L2) LISP Encapsulation Format - draft-smith-lisp-layer2-01.* [Online] [Cited: 11 29, 2017.] https://tools.ietf.org/html/draft-smith-lisp-layer2-01.

[115] **Sridharan M, et al.** *NVGRE: network virtualization using generic routing encapsulation.* s.l. : Internet Draft, IETF, 2012.

[116] **Cisco Systems.** Service Insertion with Cisco Application Centric Infrastructure. *Service Insertion with Cisco Application Centric Infrastructure.* [Online] [Cited: 12 12, 2017.] https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-732493.html.

[117] —. Cisco Application Centric Infrastructure and VMware Integration . *Cisco Application Centric Infrastructure and VMware Integration .* [Online] [Cited: 12 02, 2017.] https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731961.pdf.

[118] **Danilo Ciscato, Mark Fabbi, Andrew Lerner.** *Magic Quadrant for Data Center Networking.* s.l. : Gartner , 2017.

[119] **Vahdat, Amin.** ONS Keynote Vahdat 2017. [Online] http://events17.linuxfoundation.org/sites/events/files/slides/ONS%20Keynote%20Vahdat%202017.pdf.

[120] **Fang, et al.** Hierarchical SDN for the Hyper-Scale, Hyper-Elastic Data Center and Cloud (Article 7). *SOSR 15 , Proceedings of the 1st ACM SIGCOMM Symposium on SDN Research.* Santa Clara, CA : ACM, 2015.

[121] **Yonghong Fu, Jun Bi, Senior Member, IEEE, Ze Chen, Kai Gao, Baobao Zhang, Guangxu Chen, and.** A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale Software-Defined Networks. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT.* 2015, Vol. 12, 2.

[122] **TOCA: A Tenant-Oriented Control Architecture for Multi-domain Cloud Networks. *The 18th Asia-Pacific Network Operations and Management Symposium (APNOMS).* 2016.**

[123] Facebook Code. Facebook Open/R - Open Routing for moder networks. *Facebook Code.* [Online] [Cited: 01 02, 2018.] https://code.facebook.com/posts/291641674683314/open-r-open-routing-for-modern-networks/.

[124] *ACM SIGCOMM 2013 - B4: experience with a globally-deployed software-defined WAN .* Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh,. s.l. : SIGCOMM13, 2013.

[125] Vahdat, Amin. Networking Challenges for the Next Decade. [Online] Google, 2017. http://events17.linuxfoundation.org/sites/events/files/slides/ONS%20Keynote%20Vahdat%202017.pdf.

[126] *2016 ACM SIGCOMM - Evolve or Die: High-Availability Design Principles Drawn from Googles Network .* Ramesh Govindan, Ina Minei. Mahesh Kallahalla, Bikash Koley. Amin Vahdat. Florianopolis, Brazil : ACM, 2016. 978-1-4503-4193-6 .

[127] Fang, Luyuan. Hierarchical SDN to Scale the DC/Cloud to Tens of Millions of Endpoints at Low Cost. *IETF 91, SDNRG.* 2014.

[128] Code, FB. Disaggregate: Networking recap. *FB Code.* [Online] Facebook, 2017. [Cited: 01 02, 2018.] https://code.facebook.com/posts/1887543398133443/disaggregate-networking-recap/.

[129] McKeown, Nick. Programming the Forwarding Plane. *Stanford University Forum.* [Online] Stanford University. [Cited: 01 20, 2018.] https://forum.stanford.edu/events/2016/slides/plenary/Nick.pdf.

[130] *SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs.* Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, Minlan Yu. Los Angeles : Proceedings of SIG- COMM '17, 2017.

[131] Making Stateful Layer-4 Load Balancing Fast and Cheap . *Making Stateful Layer-4 Load Balancing Fast and Cheap .* [Online] http://conferences.sigcomm.org/sigcomm/2017/files/program/ts-1-2-silkroad.pptx.

[132] IDC. *Software-Defined WAN (SD-WAN) Survey,.* s.l. : IDC, 2017.

[133] McGillicuddy, Shamus. Establishing a Path to the network-ready SD-WAN. [Online] EMA, 2018. [Cited: 02 02, 2018.] https://d3v6gwebjc7bm7.cloudfront.net/event/15/66/20/0/rt/1/documents/resourceList1516642391548/emafuturewanfinal1516642428742.pdf.

[134] Andreessen, Marc. Why Software Is Eating The World. *The Wall Street Journal.* [Online] The Wall Street Journal, August 20, 2011. [Cited: 03 27, 2018.] https://www.wsj.com/articles/SB10001424053111903480904576512250915629460.

[135] A Koshibe, A Baid, I Seskar. Towards Distributed Hierarchical SDN Control Plane. *First International Conference in Modern Networking Technologies (MoNe Tec).* 2014.

[136] *A Purpose-built global network: Google's move to SDN - A discussion with Amin Vahdat,David Clark, and Jennifer Rexford.* ACM Queue. 3, s.l. : Communications of the ACM, 2016, Vol. 59.

[137] Bin Zhang, Pengfei Zhang, Y. Zhao, Yongkun Wang, Xuan Luo, Yaohui Jin,. Co-Scaler: Cooperative scaling of software-defined NFV service function chain. *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN),*. 2016.

[138] Vahdat, Amin. Google's Networking Lead Talks SDN Challenges for the Next Decade. *Linux.com.* [Online] [Cited: 01 20, 2018.] https://www.linux.com/blog/event/open-networking-summit/2017/5/networking-challenges-next-decade.

[139] Pat Bosshart†, Dan Daly*, Glen Gibb†, Martin Izzard†, Nick McKeown‡, Jennifer Rexford, Cole Schlesinger**, Dan Talayco†, Amin Vahdat¶, George Varghese§, David Walker**. P4: Programming Protocol-Independent Packet Processors. *ACM SIGCOMM Computer Communication Review.* July, 2014, Vol. 4, 3.**