HEPiX Network Functions Virtualisation Working Group Report

Editors: Shawn McKee, Marian Babik
Authors:
Attebury, G. (UNL), Babik, M. (CERN), Campanella, M. (GARR), Capone, V. (GEANT), Chen, Hsin-Yen (Academia Sinica), Chown, T. (Jisc), Dart, E. (ESnet), Guok, C. (ESnet), Hoeft, B. (KIT), Ibarra, J. (FIU), Lapacz, R. (PSNC), Martelli, E. (CERN), McKee, S. (Univ. of Michigan), Monga, I. (ESnet), Naegale-Jackson, S. (FAU), Newman, H. (Caltech), Seuster, R. (Univ. of Victoria), Sim, A. (ESnet), Suerink, T. (Nikhef), Zani, S. (INFN), Wu, Wenji (FNAL)
Version 2.0

HEPiX NFV WG

Ahmed Khouder
Andrea Manzi, CERN
Andreas Petzold
Balde Thierno
Ben Mack-Crane
Bruno Hoeft
Dale W. Carder, ESnet
Darcy Hodgson
Dennis van Dok
Duncan Rand, Imperial College London and Jisc
Edoardo Martelli, CERN
Enzo Capone
Eric Yen
Eygene Ryabinkin
Eyle Brinkhuis, SURFnet
Fabio Farina
Fazhi Qi
Felix Lee
Fernando Lopez, PIC
Garhan Attebury, University of Nebraska
Gerben van Malenstein
Giancarlo Viola
Gloria Vuagnin
Harvey Newman, Caltech
Hsin-Yen Chen
Inder Monga
Jan Erik Sundermann
Jin Kim
Joe Metzger
Josep Flix
Julio Ibarra, Florida International University

Kars Ohrenberg
Lars Fischer
Laurent Caillat-Vallet
Marco Marletta, GARR
Marian Babik, CERN
Mario Lassnig
Massimo Carboni, GARR
Matt Zekauskas
Mauro Campanella, GARR
Mian Usman
Michael Steder
Paolo Bolletta, GARR
Shawn McKee, University of Michigan
Stefano Zani, INFN
Rolf Seuster
Sylvain Garrigues
Tao Zhang
Thomas Hartmann
Tim Bell, CERN
Tim Chown, Jisc
Tomoaki Nakamura
Tony Cass, CERN
Tristan Suerink
Warrick Mitchell
Wenji Wu, Fermilab
Xavier Jeannin
Xin Zhao
Yury Gugel

Executive Summary

High Energy Physics (HEP) experiments rely on the networks as one of the critical parts of their infrastructure both within the participating laboratories and sites as well as globally to interconnect the sites, data centres and experiments instrumentation. Network virtualisation and programmable networks are two key enablers that facilitate agile, fast and more economical network infrastructures as well as service development, deployment and provisioning. Adoption of these technologies by the HEP sites and experiments will allow them to design more scalable and robust networks while decreasing the overall cost and improving the effectiveness of the resource usage.

The primary challenge we face is ensuring that WLCG and its constituent collaborations will have the networking capabilities required to most effectively exploit LHC data for the lifetime of the LHC.

Network virtualisation and programmable networks are nowadays quite common in the commercial clouds and telco deployments and have also been deployed by some of the Research and Education (R&E) network providers to manage Wide-Area Networks (WAN), but there are only few HEP sites pursuing new models and technologies to build up their networks and data centers and most of the existing efforts are currently focused on improvements within a single domain or organisation, usually motivated by the organisation-specific factors. Therefore, most of the existing work known in the area is usually site or domain-specific. In addition, there is a significant gap in our understanding of how these new technologies should be adopted, deployed and operated and how the inter-play between LAN and WAN will be organised in the future. While it's still unclear which technologies will become mainstream, it's already clear that software (software-defined) and programmable networks will play a major role in the mid-term.

With the aim to better understand the technologies and their use cases for HEP a *Network Functions Virtualisation Working Group (NFV WG)* has been formed within the [High Energy Physics Information Exchange (HEPiX)](). In the initial phase, the working group was focusing on identifying the work already done, looking at the existing projects and their results as well as better understanding the various approaches and technologies and how they can be helpful for the HEP use cases. This report provides a summary of this phase and in addition attempts to identify key areas that require further work that could be discussed within the community to find synergies and understand where existing efforts should be focused.

This document presents a technical overview of the existing approaches in network virtualisation and programmable networks. It starts by explaining how current paradigm shifts in the computing and clouds are impacting networking and how this will fundamentally change the ways networks are designed in the data centers and sites. Cloud native networking approaches involving new topologies, network disaggregation and virtualisation have been identified as primary drivers that will impact data centre networking, which will in turn impact how data centres will be inter-connected in the future. In the second chapter which is devoted to the WAN, capacity sharing, network provisioning and software-defined approaches are discussed and key projects in the area are highlighted. This is complemented by a brief survey on the mid-term plans of the major R&E providers. The report concludes with proposed projects areas and milestones that should serve as input in discussing the potential future work in this area.

While there are many technology choices that need discussion and exploration, **the most important thing is ensuring the experiments and sites collaborate with the RENs, network engineers and researchers to develop, prototype and implement a useful, agile network infrastructure that is well integrated with the computing and storage frameworks being evolved by the experiments as well as the technology choices being implemented at the sites and RENs.**

Introduction

High Energy Physics (HEP) experiments have greatly benefited from a strong relationship with Research and Education (REN) network providers and, thanks to projects such as LHCOPN/LHCONE and REN contributions, have enjoyed significant capacities on globally distributed high performance networks for some time. RENs have been able to continually expand their capacities to over-provision their networks relative to the experiments needs and were thus able to cope with the recent rapid growth of the traffic between sites, both in terms of achievable peak transfer rates as well as in total amount of data transferred. For some HEP experiments this has led to designs that enable remote direct data access where the network is considered an appliance with almost infinite capacity. Networks are recognized as a critical enabling component for HEP and there is an implicit expectation that HEP will continue to enjoy over-provisioned use of the world's RENs for the foreseeable future.

However, there are reasons to believe that the network situation will change, due to both technological and non-technological reasons, starting in the next few years. Various factors that are in play include the anticipated growth of the non-HEP network usage with other large data volume sciences coming online; introduction of the cloud native and commercial networking and their respective impact on usage policies and securities, as well as technological limitations of the optical interfaces and switching equipment.

As the scale and complexity of the current HEP network grows rapidly, new technologies and platforms are being introduced that greatly extend the capabilities of today's networks. **With many of these technologies becoming available, it's important to understand how we can design, test and develop systems that could enter existing production workflows while at the same time changing something as fundamental as the network that all sites and experiments rely upon**.

This report is surveying specific approaches that could be used to build scalable, high-throughput and reliable network infrastructures within and between data centres. It's aimed at **site managers**, **network engineers** and **HEP experiment infrastructure architects** looking to understand the core concepts of network virtualisation, software-defined networking and how data centres can be inter-connected in the mid-term future. In particular, the report focuses on specific network capabilities related to the cloud-native data centres, those focusing on providing significant resources in a virtualized manner and existing projects and technologies that can enable programmable wide-area networks to interconnect the data centres. In addition, potential use cases for both categories are listed and different trade-offs for the specific approaches are discussed.

This report is broken into three main chapters:
- [Cloud Native DC Network](#) - focuses mainly on the data centre network design, topologies and existing approaches in the network virtualisation. It also discusses network disaggregation and ways how this is integrated in the Linux kernel networking stack. It

concludes with a section on data centre edge, which is an area that falls in between the two chapters and provides a short overview of existing edge services and technologies.

- [Programmable Wide-Area Networks](#) - this chapter is focusing on the existing use cases and novel technologies for the data centre interconnect (DCIs), SD/WAN (SENSE, etc.) as well as applications that are integrating network and transfer technologies (BigDataExpress, NOTED). It also includes plans of the R&E providers wrt. programmable services focusing on the network provisioning and operations (tracing/monitoring) of VPNs (such as LHCONE).
- [Outlook and Areas of Future Work](#) - This is perhaps the most important chapter because we are trying to identify and initiate near-term collaborations between the experiments, the sites and the research and education networks to deliver capabilities needed by HEP for their future infrastructure while enabling the sites and NRENs to most effectively support HEP with the resources they have

Cloud Native DC Networking

Motivation

One of the main drivers for network evolution in the data centres is the changing nature of the applications; if applications wouldn't change, there would be no need to change the underlying networks. With the dawn of *virtualisation*, applications have started to morph at an accelerating pace and moved from mostly static deployments (on bare metal servers) through virtual machines to containers and more recently to clusters of containers sometimes referred to as microservices (or server-less) [MS]. This evolution shown in the Figure below, also highlights one of the main aspects that has changed for networking, the usual life-cycle of the application (develop, deploy, update, re-deploy) has decreased from hours to microseconds. Establishing a full scale cluster of hundreds of containers can be done in less than a second, and upgrades or re-deployments of such a cluster can be done on a rolling basis, which means that the entire cluster can be replaced with new containers, all over again in seconds. This increasingly dynamic environment can easily cause a lot of trouble for the networking infrastructure, which is trying to keep up with this rapid pace.
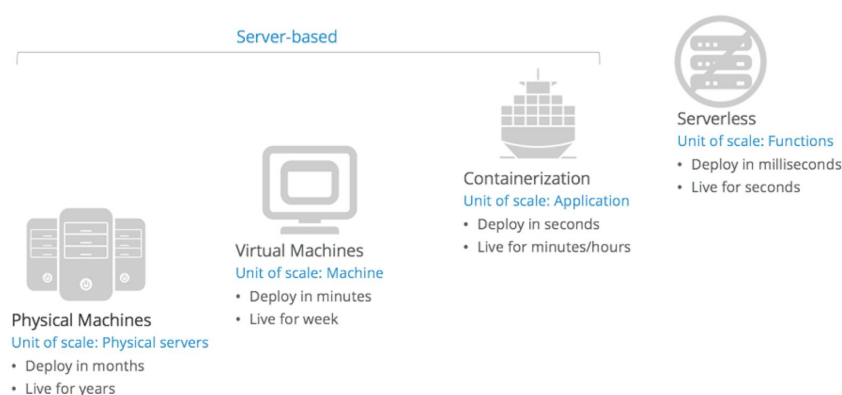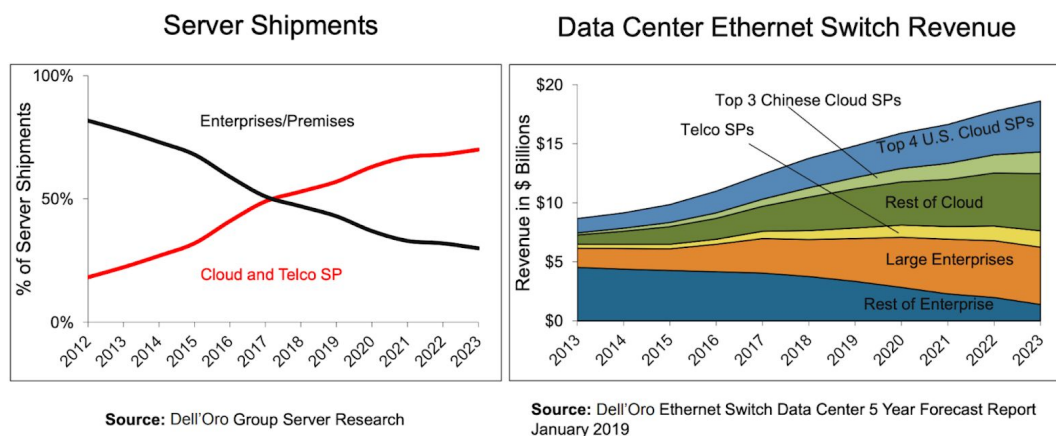

The rise of Linux and the economics of scale has led to the development and operations of clouds. HEP sites have been predominantly statically deployed with allocations usually served by batch systems, operating at job level granularities with high capacity storage hosting the datasets locally. This is slowly changing as experiments are starting to deploy their job payloads in containers and services are moving to container-based or even Kubernetes (K8s) deployments. This creates an interesting shuffle where multiple technologies are starting to overlap and compete. Recently, OSG has announced that they plan to optionally deploy all of its services on container platforms [GDB] and some of the NSF-funded projects such as SLATE, that investigate new infrastructures for sites, are entirely based on Kubernetes (K8s). Physical analysis running on K8s with a full dataset uploaded to the cloud has been demonstrated running on the Google Cloud Platform [Kubecon]. It can be expected that this evolution will continue and accelerate, potentially even picking up the pace and causing major headaches in the networking departments at HEP labs and sites.

Network engineers are facing major challenges while trying to accommodate the new computing models in an environment where they often need to support not only cutting edge container technologies that are now popular, but at the same time legacy systems ("bare metal"), virtual machines and other equipment that needs to be connected to the network or even multiple networks (e.g. experimental equipment, technical networks) with custom designs and protocols. As the situation varies greatly between the sites, this document will focus on answering the most common questions related to cloud native network design, showing possible approaches to consider and discuss their trade-offs.

Fast paced application life-cycle is not the only challenge, virtualisation is progressing into areas that were previously tied to the hardware, such as GPUs or storage systems. With GPU and storage virtualisation, there is a need for improved latency and throughput within the data centre (so called east-west) in order to enable more efficient allocation and use of resources.

Network vendors have already recognised the cloud opportunity and have started to reprofile their revenue expectations from enterprises towards cloud providers [Arista, A2, Cisco]. This will likely have an impact on what the vendors will start pushing and how the network infrastructure and supporting software will evolve in the mid-term.

Source: Dell'Oro Group Server Research

Source: Dell'Oro Ethernet Switch Data Center 5 Year Forecast Report January 2019

As cloud has the potential to become a dominant way to run compute infrastructure, this section aims to explain what impact this has on the network design and what approaches are slowly proliferating into the mainstream. This section is aimed at site managers who are responsible for not just network, but also compute and storage and are looking for a way to understand the current trends in the networking as well as for network engineers looking for information that is usually scattered around in many different sources. As cloud native DC networking is quite a topic, the main focus will be on the introduction of the core concepts, brief explanation of basic trade-offs and challenges, with references pointing to more in-depth coverage. The section is organised as follows:
- Container networking - introduces basic concepts in container networking and explains its complexity and main challenges in the implementation of the DC network design
- New topologies - introduces topologies and design principles used by the existing cloud and telco providers to build up large-scale DC hosting container and VM technologies
- Network Virtualisation - provides in-depth overview of the existing approaches in network virtualisation and discusses the main trade-offs

- Network Disaggregation - highlight an important trend in the network technologies moving to open source hardware and network operating systems based on Linux
- Programmable Network Interfaces - gives an overview of the existing approaches in programming network cards/interfaces, which is the main building block for many of the existing network virtualisation and design implementations

Container Networking

This section briefly introduces the basic concepts of container networking[1]. The reason for the focus on container networking, though current data centres are still mostly organised around bare metal and virtual machines, is that the same core concepts apply, but with containers, the scale and performance is what makes the difference. Another reason is that the intention of this section is not to introduce containers from the application perspective, but rather look at it from the perspective of networking, which is where application and network engineers in the DC meet to make critical design decisions (*and it's very important that they do meet as otherwise things are likely not going to work out*).

Both containers and virtual machines can be framed around the same concepts in Linux, which is the predominant operating system of choice, so this introduction will explain the basic concepts in Linux networking and how it connects to VM/containers world, e.g. how does container interface get an IP, how it communicates with the external world and the other containers on the same host, multi-hosts, etc.

Containers are built around two core Linux kernel constructs, *namespaces* and *cgroups* (control groups). Simply put, *cgroups* ensure proper sharing of resources such as cpu, memory and network for a collection of processes while *namespaces* provide means of isolation. *Namespaces* are a feature of the Linux kernel that partitions kernel resources such that one set of processes sees one set of resources while another set of processes sees a different set of resources [NS], while at the same time resources can be shared across multiple *namespaces*. One such resource is network, which virtualises network stacks, and allows any process to create a number of network interfaces, physical or virtual, which are only seen by the process or its sub-processes, and have their own routing table, firewall, listening sockets, connection tracking, etc. Since the release of the 4.1 linux kernel, there are five additional resources, apart from the network that can be isolated: **UTS** (allow a single system to appear to have different hostnames and domains), **mount** (controlling mount points), **process ID** (this makes it possible to have multiple processes with PID=1, each in its own namespace), **Interprocess communication** (IPC) and finally **user IDs**.

As *network namespace* encompasses the entire network stack all the way up to and including the transport layer, it's perfectly fine to have multiple http daemons running on a single server, all listening on the same port 80. To the network engineers this probably sounds similar to the concept of VRF (virtual routing and forwarding), which can be seen as a subset of the namespace concept and some vendors have actually implemented VRF in the Linux kernel using the network namespaces.

Containers can have four operations modes: no network, host network, single host network and multi host network. No network option is obvious, host network is usually implemented by running a privileged container that can access directly the default network stack. Single and multi-host networks are more complex, see Fig. below/next page.

---

[1] This section partly summarizes demo that was provided in one of the first HEPiX NFG WG meetings (link https://indico.cern.ch/event/715631/)

In the single host network there are two approaches, bridge or macvlan. In the first one a standard Linux bridge is created and connects any number of new containers via virtual ethernet interfaces (veth). Veth pairs are kind of tunnels that can connect interfaces between namespaces or directly to a physical interface (in this case to the bridge). This setting creates a small virtual (L2) switch that allows the different containers within a single host to communicate with each other and connect to the external world via NAT. The other option is macvlan, which is a layer 2 virtual network interface associated with a physical interface. The kernel assigns each macvlan interface a unique MAC address and then uses it to change the MAC address of the outgoing packet and vice versa to match the correct container. This way the container is exposed to the network as an additional host, however this prevents a direct ping between containers on the same host (packet needs to go to the router and back). While macvlan clearly outperforms the bridge solution, it has also some downsides, e.g. two containers on the same host don't have direct connectivity (packet needs to go to the switch and back), assignment of IP addresses needs to be coordinated and finally there is a limitation in the number of MAC addresses for a single NIC.
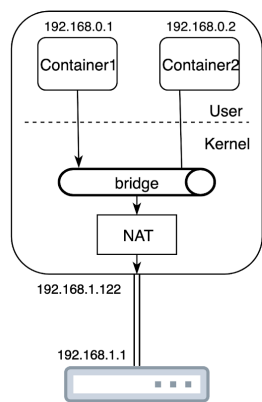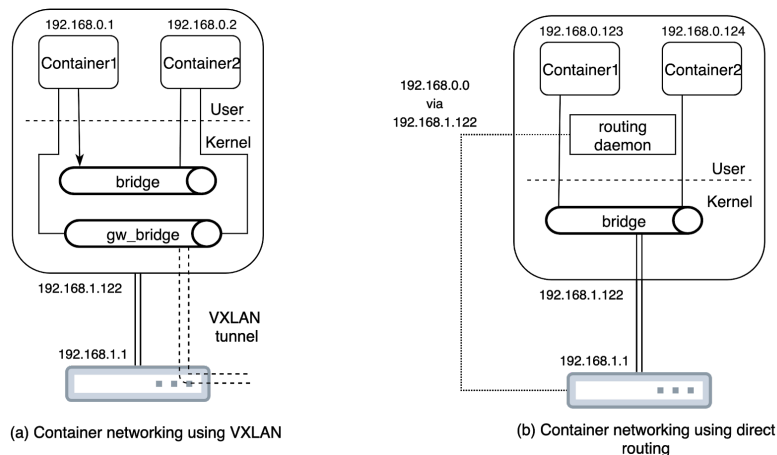

(a) Container networking using NAT
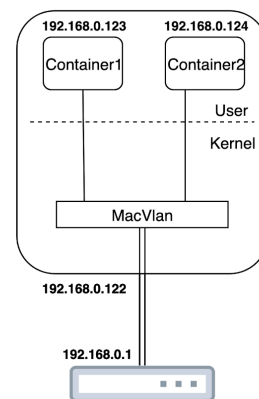
(b) Container networking using MacVlan

In the multi-host networking there are also two options, overlay network or direct routing: Overlay network creates layer 2 network between containers, the most common protocol used is VXLAN where each VTEP (VXLAN tunnel endpoint) is connected to a server/hypervisor IP address and tunnelling is established between the bridges created inside each server, see Fig. below. Coordination of IP address assignment and tunneling management is then usually performed by an orchestrator, e.g. Swarm in case of Docker, Weave, Flannel, etc.. When overlay networking is used, Docker will set up two interfaces, one for communication between containers (bridge as described in the single-host networking) and the other one for communication with the outside world (docker_gwbridge, which has its own subnet). The routing table is then setup to use the appropriate interface for each form of communication.

<div style="display:flex">
(a) Container networking using VXLAN
</div>

<div style="display:flex">
(b) Container networking using direct routing
</div>

Direct routing on the other hand is a layer 3 solution, which uses a single host container network driver, typically bridge, to build multiple containers that are connected via routing (there is no NAT), see Fig. above (part b). A routing daemon runs on the individual servers and announces the container's IP address or the bridge subnet. Examples of such models are for example Tigera's Calico, Cilium, Contiv/FD.io and others (all described in more detail in the next section). The Kubernetes orchestrator can use kube-router that can be deployed to provision direct routing. Finally, this solution can also run with an independent daemon (like FRR), which would then advertise the bridge subnet route whenever it comes up (i.e. without writing additional code to integrate with the orchestrator).

In general, there are trade-offs between the approaches described; overall, direct approaches are likely to outperform the ones based on overlays. This is mainly due to the processing needed to operate the overlay tunnels but also due to the complex setup that is usually needed on the end host (chain of veth pairs, bridge and NIC). On the other hand, direct approaches increase the complexity of DHCP assignment due to the presence of multiple IPAM masters (server vs container) and require L3 networks, which in turn means deployment and operation of separate daemons that integrate with the rest of the networking (usually via BGP).

Kubernetes (K8s), which is the most popular container orchestrator, organises containers into pods (set of containers spun up together, usually sharing the same namespace and cgroup) and provides pluggable networking models via CNI (Container Network Interface). For a functional cluster one has to deploy a pod networking solution (via CNI) which is expected to use no NAT for communication between the pods, all nodes should communicate with all containers without NAT (and vice versa) and the IP that a container sees itself is the same IP that others see it as [CNI]. Any non-trivial containerised application will run multiple pods for different services as well as a layer-4 service proxy that provides load balancing and service discovery. In addition, there are network security policies that need to be implemented in order to secure the pods. All this makes it a non-trivial endeavour considering this is just one of the applications in the DC networking.

Virtual machines networking is quite similar to what was described, the same concepts apply, setup is usually more static as VM turn-around is not as frequent. The usual option is to connect the VMs via veth pairs and configure some form of network virtualisation as described in the next section.

Overall, with so many possible choices and such a wide range of networking functionality available already on the servers, there is no doubt that this is becoming a major challenge. Deploying individual solutions for each functionality can easily end up with lots of moving parts making it extremely difficult to operate and troubleshoot. Coming up with a single solution that would work well for the entire range is non trivial and requires both application and network engineers to come together, which (among other things) makes container networking hard. The next section discusses some of the existing industry approaches and their trade-offs in more detail.

Rethinking Network Design

As was shown in the previous section, the current generation of applications are complex sets of services that run on a simple compute infrastructure with multiple levels of virtualisation that rely on a simple networking model that scales easily and can support a significant amount of east-west traffic (traffic between nodes in the DC).

For these reasons, most (if not all) of the cloud native data centers currently rely on the Clos network topology. The most basic example of such topology is shown in Fig below, it shows a network organised in two layers of switches, one called the spine and the other called leaf (top-of-the-rack switch), this is a so called 1-tier setup, 2-tier setup can be created by adding additional spine level called super-spine.
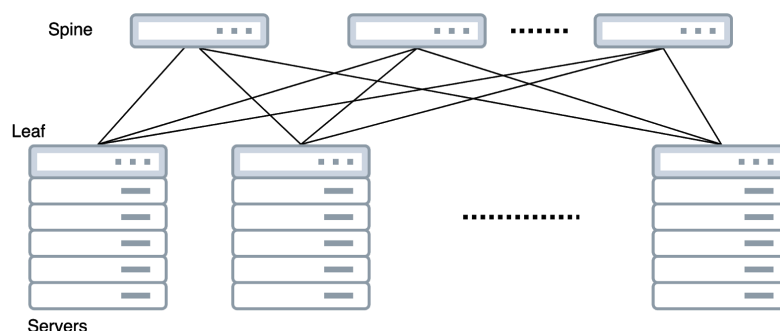
Sample Tier-1 Clos topology, compute is attached to leafs/ToRs, each leaf is connected to multiple spines.
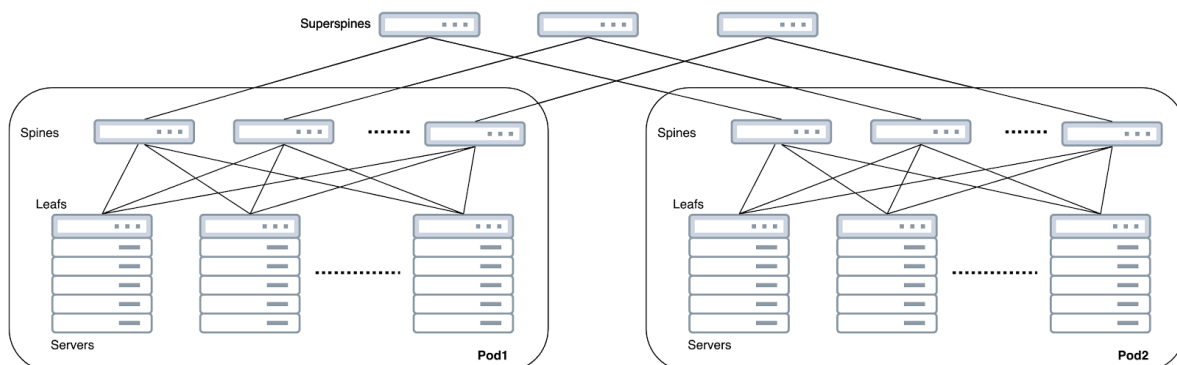
Sample Tier-2 Clos topology, compute is attached to leafs/ToRs, each leaf is connected to multiple spines (altogether forming a pod) and each spine within a pod connects to super-spine.

This topology produces a high-capacity network as there are more than two paths between any two servers and adding more spines increases the available bandwidth between the leaves. It can easily grow in a consistent way as one can just add more leaves and servers and use spines to

scale the bandwidth between the edges. Clos also pushes all of the functionality to the edges of the network, the leaves and/or the servers, while spines are solely used to connect the leaves. This means that the control plane load on the spines doesn't increase (or increases only marginally) as more leaves are added to the network.

Some of the important aspects of this topology are as follows:
- Clos can be used to build very large networks with simple fixed form factor switches - allowing homogenous equipment - that greatly simplifies inventory management and configuration
- The interconnect model is usually based only on routing, bridging is supported only at the leaves (i.e. within a single rack). The rest of the inter-connectivity relies on some form of network virtualisation (as described in the next section)
- The usual oversubscription ratio used in Clos is 1:1 resulting in a non-blocking network (technically it's non-blocking only if the flows can be rearranged from the congested links - but this is easy if routing is the only interconnect model)[2]
- Interconnect link speeds - using higher speeds for the inter-switch links can decrease the number of spine switches reducing the cost of cabling and number of switches to manage. The other possibility is to use more switches at lower speeds, which increases the potential scale of the network and can provide better load balancing
- Failure domains are more fine-grained, e.g., the loss of a link on a spine switch results in a loss of bandwidth that is fractional to the number of spines. In addition, loss of the link only impacts the leaf that lost the link, the rest of the network continues at full bandwidth.

While there can be many different options and configurations of the topology for different requirements, the common goal is to make sure that the equipment runs reliably and cost efficiently. In order to ensure this, it's important to also mention some common principles that in a way overturn the usual principles the DC have been built in the past.

Using standard simple building blocks with Clos, it's possible to build out the network with just one or two types of boxes, which is very different from using many different enterprise specific appliances (or chassis switches) that support a wide range of functionality, many custom configurations, modules, that can all be managed from a management console. However, in order to achieve the simplicity some form of network automation needs to be adopted as it's no longer possible to rely on the vendor-specific configuration tools.

The other aspect is the need to rethink the failure model. Instead of focusing on the reliability of the network with reliable nodes, focus on the reliability of the network itself (and not nodes). This is primarily achieved by relying on routing (L3), which can gracefully degrade, if the router doesn't hear from a neighbor, it will simply try to route around it, which is inherently more stable than L2 failures. With the small factor switches, equipment is relatively cheap and it's easy to replace one box with another (allowing a faulty switch to be debugged offline), so failures, when they occur, can

---

[2] Theoretically, with 64-port switch for both leaf and spine and 1:1 subscription the total number of servers that can be connected is 2038 ($n^2/2$ with n-port switch); total number of switches for full 2-tier topology is n+n/2 (with 64 port switches; 96 switches are needed for full build up). In practice, there are also other considerations, such as cooling and power available in the rack that needs to be taken into account, which usually limits the number of ports that can be used on the leafs.

be relatively quickly fixed. Upgrades are also simpler as it's possible to simply drain traffic gracefully from nodes that are scheduled for maintenance (again using layer 3 routing).

Finally, it's very important to **only** rely on the features that are essentially needed. This, while obvious, can be quite challenging in practice as traditional vendors will come up with many different features or package features together for a better offer. Sometimes this could be because a simpler model is simply not supported (e.g. unnumbered interfaces with OSPF/BGP).

In practice the situation usually gets a lot more complex, since most of the existing HEP sites won't be able to start their network design from scratch. This increases the challenge, as progressive transitions to the new architecture must be found and there will likely be significant resistance to even start it. As was shown in the previous section, postponing it will likely only escalate the networking issues in the DC. Other constraints come from the applications that need multicast or broadcast for discovery, which then makes it hard to eliminate layer 2 altogether.

In summary, a set of basic network topology concepts and principles have been reviewed in this section. There are other references that discuss the design of the cloud-native networking in more depth and can be used as a potential starting point from here [CNDCN, FB, MPLSSDNEra]. The effective use of this topology relies on the ability to deploy a working network virtualisation solution, which is what's surveyed next.

Network Virtualisation

Network virtualisation is a way to organise a single physical network into multiple isolated virtual networks (this is comparable to the server virtualisation concept). In packet networks, each virtual network will assume it owns the interface or link, forwarding tables, policy and packet manipulation tables, packet buffers and link queues. At a higher level, each virtual network assumes it owns the entire address space and all its interfaces. And similar to the compute virtualisation where VM assumes it owns all CPU resources assigned to it, but this in reality can have an oversubscription ratio, one can also have more virtual networks than there are physical interfaces. This is achieved by adding a virtual network identified (VNID) in the packet header, which is then used to select the appropriate forwarding and flow tables.

While there are many network virtualisation solutions for the data centre, most of them rely on a small subset of the existing technologies and concepts that could be mixed together in different ways to create the final solution. Some of those are:
- **Virtual Local Area Network** (VLAN) - provides layer 2 abstraction, it works by applying tags to network frames and handling of these tags in networking systems – creating the appearance and functionality of network traffic that is physically on a single network but acts as if it is split between separate networks. In this way, VLANs can keep network applications separate despite being connected to the same physical network. Number of VLANs on a given network is restricted to 4094, which for usual cloud native DC requires finding ways to reuse VLANs or combine them with other protocols/technologies. In traditional bridging networks it's used together with Spanning Tree Protocol (STP), which prevents the loops and constructs a single tree across a bridged network to forward all packets.

- **Virtual Routing and Forwarding** (VRF) - is a layer 3 abstraction, which provides a separate routing table for each virtual private network (VPN), usually this is done by adding some sort of VRFID to the routing table lookup.
- **Virtual eXtensible Local Area Network** (VXLAN) - a typical example of an overlay model, which is used to build a traffic isolation at layer 2 across layer 3 infrastructure. Usually it's coupled with VRF to provide a full bridging and routing overlay solution.
- There are several other overlay protocols, some notable examples are Network Virtualisation over GRE (NVGRE), pursued mainly by Microsoft, it provides network virtualisation over Generic Routing Encapsulation (GRE) protocol [RFC7673], STT (Stateless Transport Tunneling) and Generic Network Virtualisation Encapsulation (GENEVE), pursued by VMware, Intel and RedHat. While each offers different benefits, their adoption varies and this makes VXLAN usually the most popular choice.
- Multiprotocol Label Switching (MPLS) - grandfather of all virtual networks; operates between layer 2 and layer 3, so it's often referred to as layer 2.5 protocol. It assigns labels to packets and uses the labels to decide on subsequent routing and forwarding. It has been around since the nineteen nineties and one of its core use cases was to establish and manage layer-3 VPNs (L3VPN).

Another important aspect in implementing network virtualisation is the *control plane*, which needs to provide the means by which the virtual networks are provisioned. At a minimum, the control plane needs to provide ways to exchange the mapping between inner and outer addresses for the tunnels to be established as well as a list of virtual networks supported at each endpoint. There are primarily three different directions:

- Ethernet Virtual Private Network (EVPN) is an example of a popular control plane for network virtualisation as it can closely integrate with the network equipment. It uses Border Gateway Protocol (BGP) as its control protocol to exchange reachability for virtual networks and is usually coupled with VXLAN for packet encapsulation, which can have a native routing implemented directly in the network equipment (this is called RIOT, routing in and out of tunnels).
- Another option is to use a centralised controller, this follows the Software Defined Network (SDN) model (more on this later) and can use different control protocols (such as OpenFlow, P4) to directly program the virtual networks on the compute endpoints. In many such cases the network equipment is unaware it is running a network virtualisation solution. This direction is mainly pursued by solutions that prefer to run a deployment without controlling the underlay, some notable examples are VMware NSX, Nuage Networks, Midokura, etc.
- There are also hybrid approaches where a centralised controller is used, but integrates with network equipment by using control protocols such as BGP (by running BGP reflectors or routing daemons that can integrate directly with physical equipment).

Network tunnels have become a popular choice for virtual networking in the cloud native data center due to their scalability, flexibility and simple management. Scalability is the primary benefit as usually only the leafs need to be aware of the network virtualisation and sometimes, for very big cloud providers, the tunneling layer can be pushed all the way to the compute nodes. There are also some notable downsides to tunneling, one such is offloading, which is usually performed directly by the NIC. Adding encapsulation diminishes the usual offloading mechanisms unless

there is a support for VXLAN offloading on the card or encapsulation/decapsulation is left for the upper layers of the network (EVPN). Another potential issue is with the maximum transmission unit (MTU), which needs to be configured correctly on the endpoints to account for the additional headers needed for tunneling or it could cause serious issues in debugging connectivity. Finally, debugging and tracing tools need to be aware of the tunnels, otherwise the information they show is misleading or not relevant.

Network Virtualisation Solutions for the DC

There are a number of existing solutions that offer usually a mixture of all of the above mentioned approaches and are currently being tracked by the [Cloud Native Computing Foundation](#) as shown in Fig below. Both open-source (white) and commercial (grey) approaches exist, but it should be mentioned that some of the commercial approaches actually rely on an open-source software (like Cumulus) while others could use their own protocols, controllers or even customized standards (e.g. commercial providers).

Our focus will be on **open source projects**, which are mature enough to be deployed in production. They can be essentially divided into three large areas, or schools of thought:
- **Network Operating Systems running on hardware/bare metal switches**: Those are usually based on the ideas of network disaggregation and can implement a full range of different networking solutions (typical example of such an approach is Cumulus), their integration with compute orchestrator is however not always straight-forward. One particular approach is to use eBGP as the only routing protocol in the DC and EVPN/VXLAN for the network virtualisation.
- **Software switch approaches** are based on the idea of running a generic software switch directly on the compute node, which is usually configured from a centralised controller (e.g. OVS, Tungsten). Such approaches can usually support some form of integration with the network equipment or can run without any integration as just another application on the network.
- **Linux kernel network stack extensions**, is a set of projects that usually implement or extend some specific area of the Linux kernel networking to provide network and security services for containers (Calico, Contiv, Cilium would fall into this category). Such approaches are usually tightly integrated with the compute orchestrator and usually use some form of BGP-peering to integrate with the rest of the DC networking.

Cumulus

Cumulus approach is based largely on few important advances:

- Rise of the open source network operating systems based on open source software suites, such as Quagga and FRR (backed by Cumulus Networks, BigSwitch Networks[3] and 6Wind). Free Range Routing (FREE) is an open source network routing suite (developed to work with Linux kernel using standard kernel interfaces[4]) that implements most of the popular routing protocols (OSPF, BGP, IS-IS, etc.).
- Ability to run the open source network operating systems on open commodity switches (white boxes) that were in turn made possible by the significant rise in good quality merchant silicon vendors and subsequent rise of original design manufacturers (ODMs).
- With the adoption of Clos topology and layer-3 only design, exterior Border Gateway Protocol (eBGP) is used as the only routing protocol in the DC [rfc7938].
- Rise of the EVPN/VXLAN adoption for the network virtualisation.

A sample configuration following this approach is shown in Fig below and shows sample assignments of ASNs (ASN-per-rack with spines sharing the same ASN) as well as IP addresses assignments for the underlay. The configuration can be simplified by using a number of different BGP extensions, such as unnumbered interfaces, peer groups, routing policies, route maps, etc. more details in [CNDCN]. The integration with the orchestrator is performed by configuring the leafs to peer (via BGP dynamic neighbors) with the BGP daemon running on the compute nodes (such as kuber-router or bird).
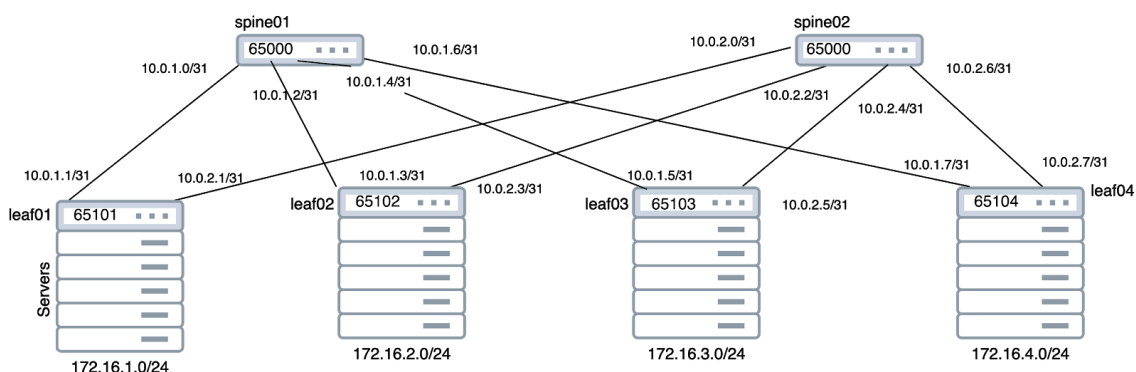
Network virtualisation is then built up with EVPN, which uses BGP as its control protocol and VXLAN for packet encapsulation. There are two options from where the edge of the VXLAN can start, either directly at the compute nodes, which can be done with FRR to start EVPN on the host, or alternatively on the leafs (which can be considered to be the most common deployment model of EVPN in the DCs). In this case it's possible that eBGP session is used for both building the routed underlay and the exchange of the virtual network information.

OpenVSwitch

OpenVSwitch is a prominent open source implementation of the *OpenFlow* switch that was developed by RedHat and VMware. *OpenFlow* is a protocol that was proposed in an influential paper that framed the basics of the Software Defined Networking (SDN) model and was based on

---

[3] BigSwitch Networks was acquired by Arista Networks in February 2020.
[4] This is specific to Cumulus Linux, other network operating systems/ASIC integrations might require specific code to program the forwarding place. It should also be noted that while Cumulus builds on top of open source, it does contain proprietary extensions (such as switchd for example).

the idea that packet processing can be determined by the flow tables, available on most packet switching silicon, and can be programmed remotely, usually from a centralised controller (so called SDN controller).
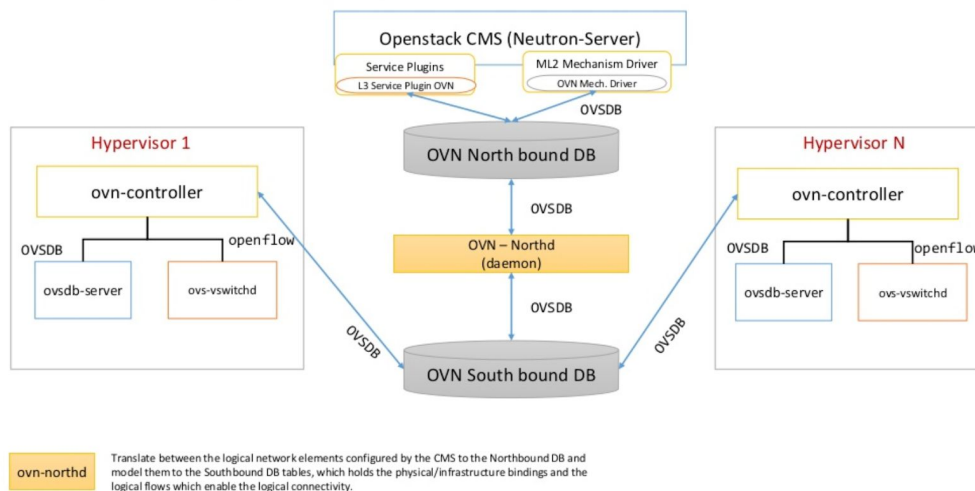
Following this model, OVS implements ovs-switchd, which is a Linux user space application that implements an OpenFlow switch (running directly on the compute nodes) that can be programmed remotely from a centralised controller called Open Virtual Network (OVN). Additional OVS components are the kernel module that handles the datapath packet processing and OVSDB which persists the state of the ovs-switchd. Datapath module has pluggable offloading mechanism for VXLAN, GRE offloading (if NIC supports it) or DPDK. OVSDB has IETF specification [RFC7047] and was implemented as a component in some of the physical switches, thus making it possible to integrate easily with OVN or other controllers (VMware NSX is using this approach). Multiple encapsulation methods are supported including GRE, VXLAN and GENEVE to create virtual networks. OVN as the central controller is used to integrate with the orchestrators (K8s and OpenStack).

Apart from OVN, there are a number of other controllers that can be used to program OpenFlow switches (both physical and software-based), but it's unclear to what extent those are actually used outside of testlabs (in production setups).

In general, flow tables have proven to be very useful for certain functions (policy and security control) and are used in many other approaches. SDN/OpenFlow approaches have been very successful in the SD/WAN area and are quite successfully used in the existing production systems. There are also successful production deployments of SDN/OpenFlow in DCs, most notable is Google's GCP (it should be mentioned that this was achieved using proprietary protocols, tools and equipment).

There are however also a number of obstacles in OpenFlow adoption and its production deployments:
- OpenFlow as such suffers from some fundamental problems; silicon implementation of flow tables is difficult to scale (inexpensively); abstraction models proposed are too loose (or too strict) allowing for many different implementations that don't work together; the proposed

model is trying to encompass a wide range of functionality, which addresses not only what doesn't work well in the classic networks but also what worked well for many years. This probably explains why OpenFlow hasn't seen a broader adoption in the DCs.
- OVS re-implements a significant part of the code already available in the Linux kernel (ARP, IP, etc.) as it doesn't use the lookup tables or packet forwarding logic (using OpenFlow instead). This will be difficult to maintain in the long run and will require commitment going forward.
- Integration with physical equipment is complex due to different versions of OVSDB and different levels of support from the network vendors.

Tungsten Fabric

Tungsten Fabric is an open source project (branched from Juniper's Contrail) that manages and implements virtual networking in cloud environments using OpenStack, OpenShift, Kubernetes and VMWare as orchestrators, including cross-stack networking. As a basis it uses EVPN/L3VPN in combination with BGP [RFC4684] on top of VXLAN or MPLS running over MPLSoUDP/GRE (can also mix them up). It can both integrate with underlay via BGP peering, or directly manage/configure the equipment using netconf (right now only available for Juniper equipment).

Tungsten core components are: *Tungsten Fabric Controller* - set of services that maintain a model of networks and network policies and *Tungsten Fabric vrouter*[5] - software switch running on each compute node and performing forwarding and enforcing the network and security policies. It's implemented with two components (similar to OVS): Linux user space agent called *vrouter agent* and kernel datapath module (*vrouter forwarder*). Virtual networks have VRFs with routes and ACLs in the *vrouter agent* tables which are synchronised from the controller (via XMPP messaging protocol). Datapath module (vrouter forwarder) had pluggable support for different off-loading modes (kernel, DPDK, SR-IOV, and recently smartNICs).

---

[5] For a good comparison between OVS and vrouter see OVS talk by Y. Yang
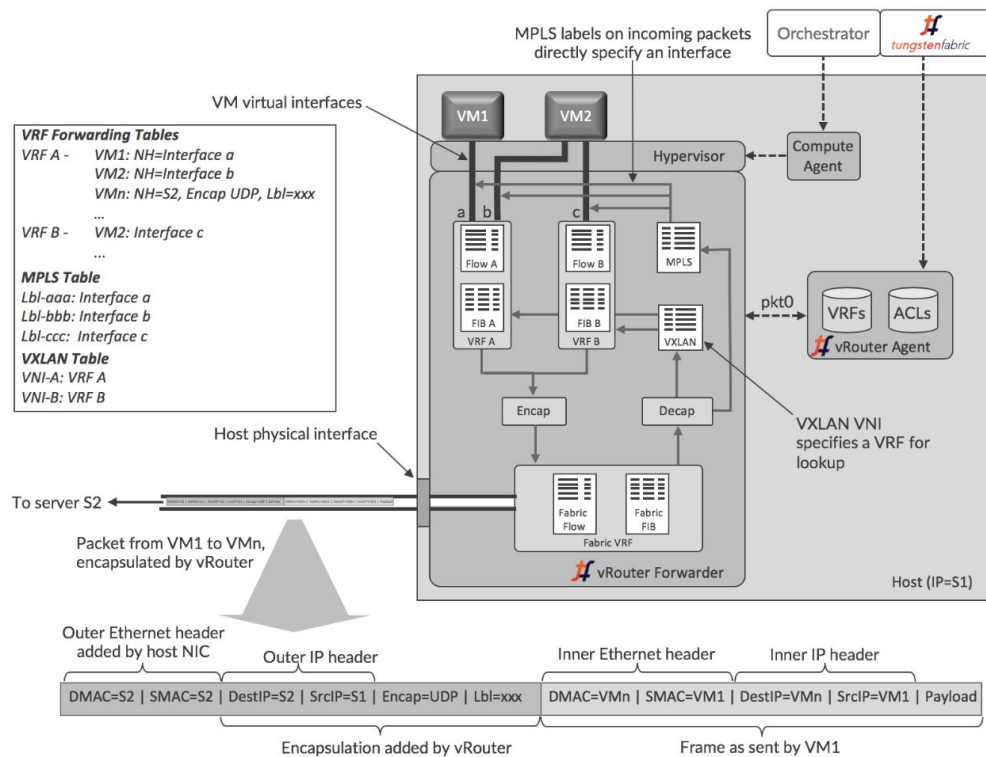
The agent establishes virtual network interfaces that are connected to the VRF forwarding tables as well as forwarding information base (FIB) in each VRF, which is configured with the forwarding entries. Flow tables are used (per VRF) to implement access control list and other security policies. In addition, VXLAN and MPLS tables are kept to map VXLAN VNIs to VRFs and MPLS labels to virtual network interfaces. Vrouter acts as a default gateway for all the virtual interfaces (responding to all ARP queries outside of its subnet with its MAC). A sample packet flow between VMs in the same subnet is shown in Fig below.
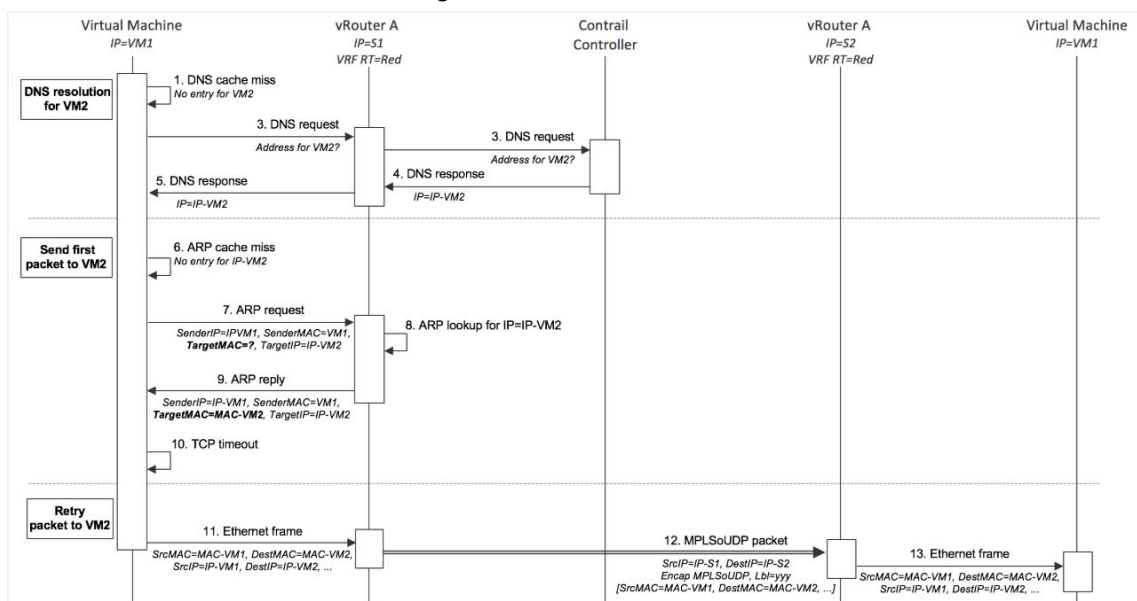
Using the forwarding mechanism described above, Tungsten can easily implement service chains (forcing traffic through specific interfaces/nodes).

Tigera Calico

Calico is a network solution primarily for Kubernetes but also supports Docker, Mesos, OpenStack and Rkt. It's an IP routed fabric that can work both on top layer-2 or layer-3 only networks and uses BGP peering to integrate. Calico is made up of the four basic components: an orchestrator plugin, a data store (etcd), felix - an agent that runs on each compute node, BIRD - BGP client that distributes routing information (also runs on each compute node) and optionally BGP Route Reflector (also BIRD), which can be used for higher scales to aggregate the BGP clients running on the nodes (avoids that each BGP client connect to all the others - full BGP mesh).
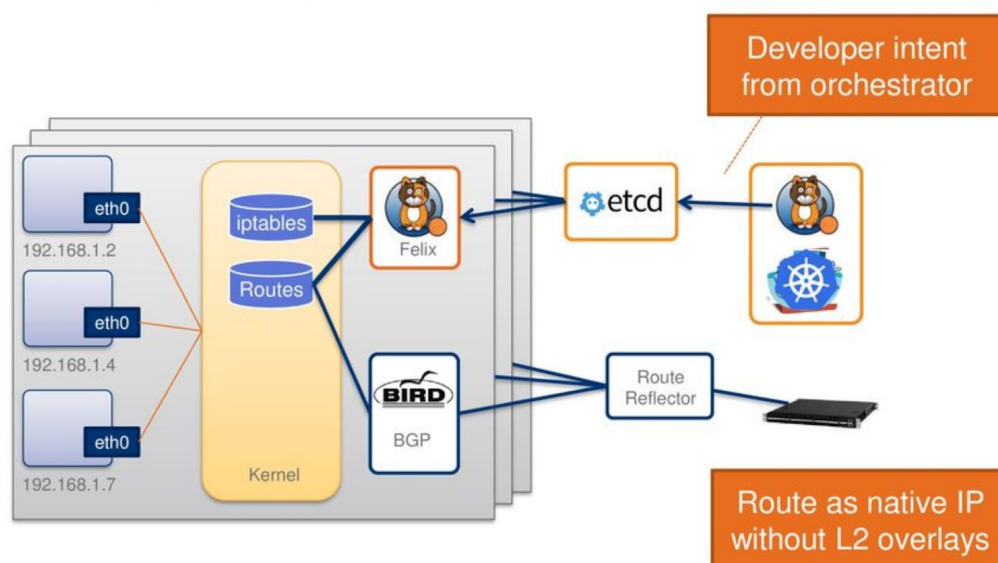
Calico is using the native Linux kernel FIB to insert the routes [Calico] and then uses BIRD to distribute those routes to the other nodes in the infrastructure. It is also using the Linux kernel iptables to enforce security policies and restrict visibility across multiple tenants. By default it's not using any overlay, but can also work over VXLAN.

Cilium

Cilium is somewhat similar to Calico, but focuses mainly on securing the connectivity between the containers, it's using the Berkeley Packet Filter (BPF, best known from tcpdump). Linux network stack supports a set of hooks that can be used to execute BPF programs and Cilium is using those to directly manipulate the packets and enforce security policies. Two networking modes are support, overlay with VXLAN or Geneve and native routing, which uses the routing table of the Linux host.

Cilium supports integration with Kubernetes, Istio, Docker, Mesos and Envoy, so it's primarily a container networking option.

Contiv/VPP

Contiv is a networking solution that directly integrates with Cisco Application Centric Infrastructure, it supports Kubernetes and OpenShift only. Contiv is using VLANs networks with subnet pools and

gateways, directly configured on the switches to interconnect the containers (with integration at the container level networking - docker network/linux bridge). It can also use eBGP peering for layer-3 mode, in which communication between containers on different hosts runs natively using VLANs and communication between container and non-containers is done by BGP peering between the compute nodes and the leafs.

Another project that is related to Contiv is ContiVPP.io, which is a Kubernetes CNI plugin that uses a combination of Contiv, Ligato libraries and FD.io/VPP. Unlike Contiv, it uses FD.io VPP for dataplane, which is an open source implementation of Cisco's Vector Packet Processing (VPP) technology. The main benefit that is claimed is its ability to provide high performance packet-processing that can run directly on the compute nodes. As this is a container-only solution, the software switch is running as a separate container in the Kubernetes (Contiv vSwitch), which runs as an agent in the user space and integrates with Intel's DPDK.

Flannel

Flannel is a simple way to configure a layer 3 network fabric for Kubernetes. It provides a layer 3 IPv4-only network between multiple nodes using several different backends. The recommended choice is the VXLAN (only one vxlan network is created), host-gw (for direct routing, remote gw must be reachable via layer-2), UDP (for debugging purposes) and also AWS, GCE and AliCloud VPC backends (experimental).

Summary

Looking at a possible list of requirements to be taken into account when picking up a solution to build up the DC network and trying to accommodate a cloud native networking, the following list of requirements show areas that can be helpful:
- Native support for multi-stack
  - Connecting and integrating multiple orchestration stacks like K8s, OpenStack, etc.
  - Only very few of the current solutions are multi-stack, but most do support both OpenStack and Kubernetes
- Networking and security across legacy, virtualized and containerized applications
  - Ability to provide networking solution covering not only multiple orchestrators but also multiple types of compute endpoints (bare metal, VMs, containers)
- Multi-tenancy/isolation
  - Support for application/experiment level networking. This is supported by all solutions described.
- Native support for multi-cloud
  - In case extending DC networks to Commercial Clouds is a requirement, or creating federations between DCs, it's necessary to look at how the current solution can support multi-cloud (or hybrid-cloud).
- Network automation
  - Ability to deploy most of the existing solutions relies on some form of automation, both at the compute level and at the network level. This is usually complex if both network and compute is involved as usually compute and network don't share the same automation tools.
- Security and observability

- - ○ Tools to support tracing and debugging are essential and native support from the solutions should expect the core ability to trace and debug or some form of analytical platform that can help in debugging the issues.
  - Across stack policy control, visibility and analytics
    - ○ If applications spanning different stacks (bare metal, VM, containers) are to be deployed then it's important to have a solution that can run policies across different stacks as necessary.

Network Disaggregation

Network disaggregation refers to the breaking apart of the router/bridge into its hardware and software components, allowing each to be purchased separately. This trend has been started and accelerated by the cloud-native networking requirements due the following factors:

- Adoption of the Clos topology requiring standard small-form factor equipment in large quantities, which requires a small set of basic features and functions
- Rise of the merchant packet switching silicon (e.g. Broadcom, Mellanox, Barefoot, Innovium, Marwell, etc.), which is now at the core of most of the network switches (including from traditional vendors) as well as the rise of the manufacturing of the bare metal switches - so called white boxes (Edgecore, Quanta, Agema, Celestica)
- Cost reduction and technical difficulties/limitations in building up large cloud providers networks with traditional approaches and vendors [AWS]

Network disaggregation is important as it significantly impacts the evolution of the network in the same way server disaggregation impacted compute in the past century. It's also important for controlling the cost of the infrastructure, not only capital costs but also operational costs by enabling standard open source tools to be easily used to operate and automate the network equipment. This trend also helps avoid vendor lockin and enables a transition to a more standards-based environment (Open Compute Project, Open Network Installer Environment, etc.).

Building up a large-scale network infrastructures following the Clos topology requires buying a significant amount of equipment, making operators very sensitive to the cost/device ratio and for ways to operate a large number of devices efficiently. This wasn't easy with the classical approaches and vendors and thus opened up the door for open network platforms and operating systems. The basic requirements include the ability to program the devices (to allow for network automation; run own components, such as routing frameworks where new approaches can be tested quickly and/or fix bugs) and the ability to run third-party applications (for monitoring, tracing, etc.).

While there are on-going discussions on the feasibility of the open source network operating systems and ODMs, there has been significant progress made in the past few years and we're well past the early adopters stage [SONiC].

Another important area of network disaggregation are the network operating systems (NOSes). Currently, the two most common elements in the modern network operating systems are Linux (on Intel x86/ARM processors) and some form of packet switching silicon, supporting the core task of

processing the packets. The major exception to this is Juniper's Junos, which is still using FreeBSD,while most of the other vendors have already migrated to Linux.

The current NOSes usually differ in the following two areas:
- Switch Network State location - a common model is to run vendor-specific code in the user space (all DPDK-based NOSes, Cisco NXOS); the hybrid model uses partially user space and kernel space (common in software switches, OVS/vrouter, but also in Arista EOS, MS SONIC, Dell's Openswitch, etc.) and the third one tries to implement all network state directly in the kernel (mainly Cumulus)
- Driver placement and programming - most common placement for the drivers of the switching silicon are in the user space, but kernel space implementations also exist. Then depending on the placement, different interfaces can be used to communicate with kernel and abstract away the silicon driver details: *netlink* (Linux kernel default), *Switch Abstraction Interface (SAI)* - hardware abstraction layer driven by Microsoft and Dell and *switchdev* - hardware abstraction layer driven by Cumulus and Mellanox, (others ?)

The list of existing network operating systems is now quite large and can be found in OCP pages. There are also additional open source network operating systems, such as for example Stratum by ONF, which was recently released as an open source.

Programmable Network Interfaces

Programmable network interfaces (line cards/switches) are currently an area of intense interest for network interface vendors, data center architects and end-users trying to optimize network performance. Ability to program the network interfaces has a number of different use cases:
- Reducing the amount of CPU workload that network virtualisation imposes on the servers and network equipment. The aim is to use dedicated hardware to completely offload packet processing to the network interface card(s) or network switches. Depending on how much functionality is being offloaded this could range from standard network cards (Mellanox cards offers significant amount of offloading features in its drivers - tc/vxlan/mpls/gre offloading) up to so called intelligent NICs (so called smartNICs, e.g. Netronome Agilio Adapters can offload the entire OVS or Tungsten's vrouter switches).
- Security isolation by means of encryption or intrusion detection, which can be performed directly on the smartNIC or on the bare metal switch
- Virtualizing Storage and other cloud resources, such as GPUs or TPUs, through protocols such as NVMe-oF (NVMe over Fabrics), which can help bring storage or graphics to any server over the network while at the same time make it look like a local resource.

Currently three hardware approaches are being followed for NICs:
- FPGA based - good performance, but difficult to program, workload specific optimisation
- ASIC based - best price/performance, easy to program but extensibility usually limited to pre-defined capabilities.
- SOC based - usually higher price, but easily programmable, providing highest flexibility

Recently, new ASIC based NICs with full data path programmability have appeared. The combination of a network programming language and new programmable ASICs is proposed to overcome the limitations of fixed-function switch ASICs. The P4 language has been proposed in 2015 and it is now maintained by ONF. P4 compilers support various chips FPGA, Barefoot Tofino, etc. Broadcom has recently started to support the NPL language for its own products.
A programmable data plane transforms the network ASICs, allowing the programming of new forwarding behaviours in the packet processor itself. Full programming control on processor memory and functions permits a substantial degree of freedom on packet header processing, including information insertion, change and removal enabling in-band Telemetry for detailed monitoring and real time dataplane functionalities.

Network programmability (tutorial) in both NICs and switches has been possible in the past, but significant advances were made recently due to network disaggregation effort, invention of new programming languages and compilers (such as P4) as well as efficient hardware implementations. Different layers of programmability are possible and most of them can be found in the existing offers:

- Application level - OpenVSwitch, Tungsten vRouter in which the entire control and data plane can be offloaded to the smartNIC.
- Packet movement infrastructure (part of data path) - BPF (Berkeley Packet Filter)/eBPF
- Full description of data path - P4 language, also available for network switches such as those from Barefoot, etc (good overview provided in ACM SIGARCH article, Netronome, Mellanox)

Overall, smartNICs provide a powerful blend of hardware capability and software programmability appropriate for implementing new dynamic compute, storage and network architectures. The challenge will be how to incorporate them into future HEP infrastructures to achieve the best impact.

Linux Kernel Networking Models

Network programmability is not limited to hardware, in the past couple of years, there has been a lot of active development in the Linux kernel related to the programmable networking (and corresponding interfaces). As Linux kernel networking stack is rather complex, requiring significant processing resources, various different "fast path" networking approaches were invented. They can be divided into three different areas (with some overlaps):

- *Linux kernel bypass models*, which try to improve networking performance by going around the Linux networking stack, this in turn needs modified device drivers (or additional kernel modules for higher level APIs) as well as handling of the upper protocol layers in the user space, some notable approaches in this area are:
  - Intel Data Plane Development Kit (DPDK) - is an open source toolkit (community established by 6WIND) to accelerate packet processing workloads (on a variety of CPU architectures, x86, Power, ARM); it works by running user space forwarding plane that interacts with a PMD driver (poll mode driver/kernel module) that supports many different NICs. It has many additional features, such as custom accelerators, QoS and extensions for monitoring and metering. It's used as an optional backend

in most of the existing projects (OVS, Tungsten, Contiv). The downside of DPDK is that it needs a dedicated CPU/core fraction to be allocated for the packet processing, thus reducing the overall resources for compute.
- VPP (FD.io) - open source version of Cisco's vector packet processing (VPP), runs in user space (used in Contiv/VPP).
- Dataplane processing with a dedicated kernel modules is also supported by software switches (OVS, Tungsten) and is often the default deployment model (offering easy deployment with reasonable performance)
- There are other projects that could fall into this area, e.g OpenDataPlane, OpenFastPath, netmap, Snabb, pf_ring, etc. but their use in production environments is unclear at the moment
- *Linux kernel fast path models*, which try to process as much data early on the data path as possible, in Linux driver code or on NIC itself:
  - eBPF - Linux socket filtering (best known from tcpdump), i.e. extended Berkeley Packet Filter (BPF), provides a very flexible and efficient way how to program packet processing directly in the Linux kernel (it's a VM-like construct for Linux kernel that allows safe bytecode execution at various hook points like for example traffic control/tc). It's used primarily in Cilium, which claims to have significant performance advantages over Calico (by avoiding extensive use of iptables and using eBPF hooks instead).
  - XDP - eXpress Data Path - uses eBPF programs and performs processing RX packet-pages directly before the driver. It can run as native or offloaded (BPF in NIC, or via DPDK).
- *Linux kernel hardware offloading models*, usually implemented in the drivers (already described in the previous section).

DC Edge

The data center edge is very important for high energy physics and deserves some discussion. HEP data centers typically require significant Wide Area Networking (WAN) bandwidth between centers. High bandwidth capacity is an area in which modern data centre networks differ from the traditional ones, in a typical 2-tier Clos topology with 4 spines/16 leaves and 100GbE connectivity between the leaf and spine results in 400GbE east-west connectivity. Connecting to the external world in the typical HEP site wouldn't require that high capacity, but north-south throughput is still a very important factor as it'll determine how to design the networking to the external world. In Clos topology there are two possible choices, one is to create another set of leaf switches (called border/exit leafs in HA setup, each connected to all spines/or superspines) and instead of connecting them to the servers, connecting them to the edge routers[6]. The other option is to connect the spines to the routers directly, which is usually a rare case in the cloud-native data centres, but it could make sense if expected north-south traffic is higher (or equal) than east-west (it can quickly add up cost as router ports are expensive in the traditional brands, so it would only make sense if there are only few spines or if using a commodity hardware).

---

[6] Number of border leaves would be determined by the oversubscription ratio btw north-south and east-west traffic in the DC
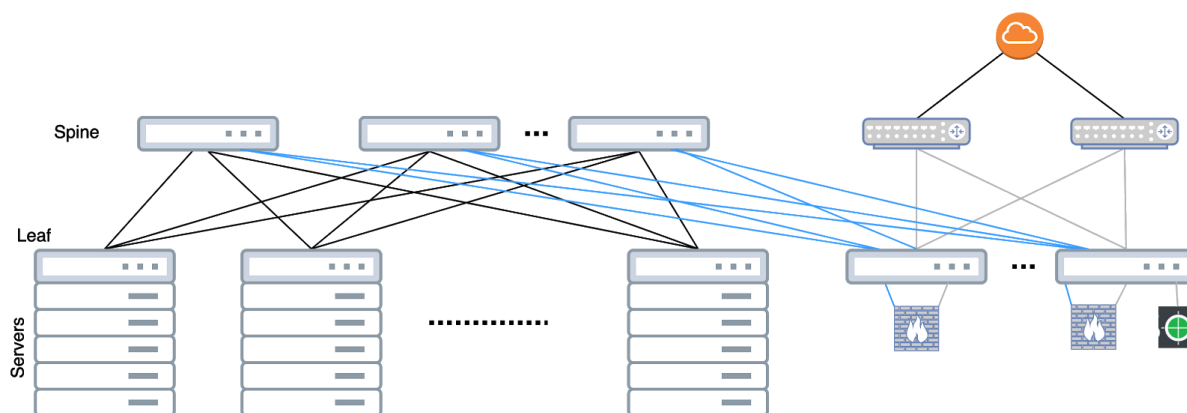
Border leaves would also be a good place where one can instrument additional equipment, such as firewalls, perfSONARs, Cloud gateways as well as strip off any routing data that shouldn't be exposed to the outside world (such as internal AS numbers, etc.). For the Cloud (or SDN) gateways, which essentially serve as a way to extend the DC network to the cloud provider (or to another site), the most common model is called virtual private cloud (VPC). VPC connectivity is usually purely layer-3 (no broadcast/multicast) and uses a virtual private router (or virtual private gateway) that routes between subnets and controls access to the external world (via NAT or by providing public IPs that are routed to the endpoints via the gateway). Connecting to a VPC involves either establishing VPN or a direct routed connection with the cloud provider (recommended). Most of the major cloud providers (Amazon, MS, Google) currently provide mostly equivalent forms of VPC [ESnet-AWS, LHCONECloud].

In a similar way, it's also possible to extend some of the existing network virtualisation solutions to the other centres or to the public clouds (via Data Centre Interconnect - DCI). Tungsten Fabric provides different ways how to extend the DC network by translating the MPLS labels from internal LAN to external WAN and back (expecting MPLS WAN backbone) or by running vrouter switches remotely connecting to a centralised Tungsten Controller [TungstenPrimer].   One of the goals of this working group is to try to document, provision and test compatible ways of matching LAN technologies (driven by the data center) with evolving SD-WAN technologies such that the resulting end-to-end network best meets our HEP needs.

The extensive use of data centers by large content providers created the need for simple, cost-effective, very high capacity connectivity between their sites. The traditional ITU-T Optical Transport Network (OTN) defines equipment does not provide the form size and the easy scalability capabilities to fulfil these requirements at an affordable cost. A more compact and simpler equipment is available now, named Data Centre Interconnect (DCI), which simplifies the optical element to its functionalities, while preserving DWDM capabilities. DCI equipment builds on the high level of integration in optical components as well as technologies that offer large-scale fabrication at low cost. Such equipment allows to provide optical connectivity in the data centre and between the data centers (as a function of distance) at a small fraction of OTN equipment, being the lasers or pluggable optics the larger part of the cost. DCI equipment is also more suited to dynamic provisioning and agile utilization than previous optical equipment.
DCI optics is also frequently coupled in the same chip with advanced, in-line forward error correction algorithms, performed by DSPs, improving substantially the WAN performance and

range at very high transfer rates. Recent advances in optical technologies for data centers were surveyed by Cheng, et al. Additional information and vendor materials are available from Infinera and CIENA.

Finally, another potential use case for the DC edge is the integration of experiment's readout systems and/or other instrumentations into the existing High-Level Trigger (HLT) topology in a similar way as is currently pursued by the telco industry, which is integrating its instrumentation and edge services directly into its core topologies. Fig. below shows a possible topology organisation motivated by Clos on how the readout systems would be integrated with the HLTs. It remains to be seen if this would be feasible in practice, but monitoring the existing approaches in the integration of the edge services and hyper-converged infrastructures can provide valuable input on how to run large-scale experiment readout infrastructures.
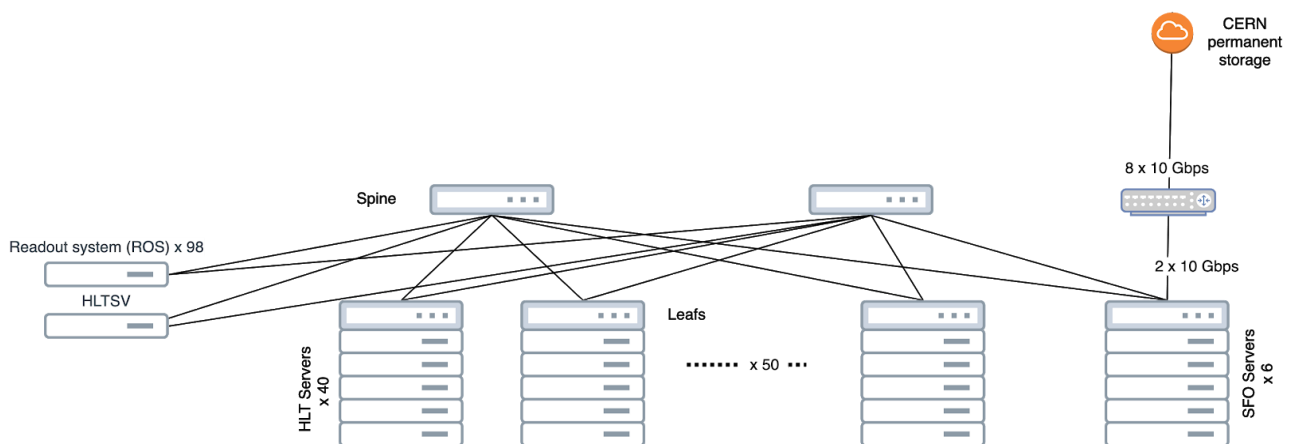
Challenges and Outlook

Cloud native networking and related technologies are providing a way how to design and cost effectively operate large scale DC networks. Still a number of challenges remain, both technological and non-technological that could impact a broader adoption by the HEP community:

● Most of the existing HEP sites won't be able to re-design their DC networking from scratch, which will pose additional challenges in order to find ways to progressively migrate and accommodate the existing constraints.
● Historically, there has been a very clear separation between network and compute, but this no longer applies for the cloud native approaches where complex networking is present at the level of servers/hypervisors. This means that network and compute engineers must work together and build up expertise in the cross-domain areas. Collaboration between the sites will also be very important to bridge the gap and come up with more effective approaches that better fit the existing HEP use cases. Encouraging closer collaboration between network and compute engineers within and across sites will be therefore an important factor in the adoption.
● Automation of the networking is another important area as relying on the open source network operating systems usually requires migrating to standard open source configuration tools. In addition, the usual approaches that work for configuration of compute might not work for network infrastructure due to various reasons [CNDCN].

- Additional work on performance studies and performance tuning of the previously mentioned systems will be needed. This is mainly to evaluate their performance for the HEP specific network patterns (fat flows, frequent remote access, etc.). In addition, comprehensive performance studies comparing different approaches are currently missing (performance tests over LAN/WAN via SR-IOV, OVS, VRouter, DPDK, with/without flows, with/without smartNICs, etc.) [10Gbps benchmarks].
- DCIs (both HW and software-based) are quite novel approaches that will require initial deployments and testbeds to evaluate how they could benefit inter-DC networking (federated/data lakes) and how they could be integrated into the existing R&E infrastructures (commercial DCI software approaches usually rely on MPLS backends, it's unclear if there are alternatives and what developments are needed to make this work over the existing R&E networks).
- DC network extensions towards Cloud via VPC have been demonstrated and different trade-offs have been identified [ESNet-AWS]. Further tests and experiences are needed to better understand if some of the new technologies could help in this quickly evolving area.
- A range of other approaches that were only mentioned briefly such as GPU, virtualised storage (using NVMe-of, e.g. CephNVMEof, MellanoxNVME), hyper-converged architectures and edge services for HEP instrumentation and experiments will require initial testbeds and evaluations.

Overall, this chapter summarises core solutions and technologies that could help our community to rethink the way we design and operate our data centre networks and provide a great opportunity to build large scale data centers (centralised or distributed) that could benefit from economies of scale, simplification of the operational models and potential reduction in the overall operational cost, but apart from technology this will require new policies, priorities and funding to materialise.

Some of the most promising areas of R&D that could lead to a broader adoption of the mentioned technologies are container-based compute platforms such as SLATE edge services, HTCondor K8s backend or native K8s sites that could offer the best opportunity to test and evaluate cloud native networking. Another potential area are batch system deployments using VMs where an isolated region of OpenStack/VMware can be provisioned and tested using the native integration with Tungsten, OVS or other approaches. Provisioning of the storage servers with software switches and/or virtualized storage solutions is another area that has the potential to be easily deployed and tested. Finally, it's also very important that cloud native approaches are considered for any planned extensions of existing centres or new data centers right from the start.

Programmable Wide-Area Networks

Motivation

Paradigm shift in the computing and its impact on the network technologies as described in the previous chapter will make it easier to design and develop bigger data centres that will be inter-connected at very high capacities at lower latencies (by means of DCIs) and could easily off-load to nearby Clouds, HPC centres or other opportunistic resources. A cluster of such centres can then create a federated site that will be exposed behind a single endpoint/interface for the experiments. This transformation has already started within the WLCG data lakes activities (WLCG DOMA WG) and some of the participating sites are already running their storages/compute in a federated setup [ref DOMA pilots].

At the same time, small to medium sites will be able to complement the functionality of the core sites by off-loading some of the Compute activities that can be orchestrated and provisioned directly from a core site by means of Kubernetes or other federated orchestrators (such approaches are being followed by OSG, NSF-funded SLATE and other "lightweight" site projects). This can have a profound impact as design and development of the offline HEP distributed computing model can be radically simplified. Recently, SLATE edge services have been released, which are based on Helm and K8s and HT-Condor team announced that it plans to add the ability to use Kubernetes as one of its backends. This could become a key-enabler for HEP sites that could support different workloads by only running a single container orchestration or edge system (this would be revolutionary and impact many different aspects of running a HEP site apart from networking such as security, operations and management policies).

From a network perspective operating a set of clustered DCs will bring its own challenges and will require close collaboration with the R&E providers. Provisioning of the networks, network telemetry, packet tracing and inspection as well as overall security and network automation will need to improve in order to make it easier for the federations to operate not only their inter-DC activities, but also easily expose their services to the outside.

Historically the National Research and Education Networks (NRENs) have managed to meet HEP networking needs by strategically purchasing capacity when network use exceeded trigger thresholds (over-provisioning). This has been a straightforward method to provide seemingly unlimited capacity for HEP, requiring no new technologies, policies or capabilities. There were occasional "bumps" when regional or local capacities didn't keep up, but overall over-provisioning resulted in excellent networking for HEP.

While over-provisioning has been very good at meeting overall HEP capacity needs, it hasn't addressed localized over-subscription (in time or space) that leads to packet-loss. This is important, since any packet-loss on our long-fat network pipes significantly degrades the throughput TCP can achieve. For example, losing 1 packet out of 22,000 on a Trans-Atlantic 10 Gbps path can reduce the throughput by a factor of around 85, turning a 1.2 GByte/sec transfer into 15 MByte/sec transfer.   Unfortunately there are many ways to lose packets on the network

including locations where the network bandwidth changes, where heavily used network paths converge to a smaller number of paths (congestion), where device buffers are insufficient to handle micro-bursts or where there are misconfigurations, failing hardware or lossy network cabling. Some of the issues are solved by using good network design principles or by purchasing upgraded hardware but in practice these kinds of issues will persist in our infrastructure, simply because sites and NRENs continuously evolve their infrastructure with no global (or even regional) coherence.

While the R&E end-to-end network paths are not under any individual organization's control, there is something that can be done to improve the performance of flows while simultaneously creating a less volatile network environment for other users of the network: **packet pacing**.   Some background: by default, network interfaces try to inject their data buffers into the network at the speed the interface operates at.   A 100 Gbps interface with a buffer of data to transmit, chunks the data into packets corresponding to the Maximum Transmission Unit (MTU) and puts them on the network at a rate corresponding to 100 Gbps until the outgoing buffer is emptied or until TCP limits the sending of packets.   This is OK as long as all the network segments and the receiver are running at 100 Gbps. However, if any part of the path has a speed less than 100 Gbps, packets begin to queue-up until packet loss occurs and TCP responds by halving the sending rate. The resulting ramp-up and back-off of TCP leads to an average transmission rate significantly below the theoretical bottleneck link capacity.   In addition this kind of bursty flow can have an adverse impact on other users of the network as packet trains at the higher rate (micro-bursts) fill various buffers and cause packet-loss to other flows. Wouldn't it be much better to pace the original packets to match the bottleneck rate?   For example, if a 100Gbps server was sending to a 50Gbps server, a logical starting point would be to pace the packets leaving the 100Gbps network interface at 50Gbps. Assuming the bottleneck for the transfer was the 50 Gbps interface on the destination, this pacing would result in a much smoother flow able to hold the 50 Gbps rate. Of course the bottleneck could be elsewhere along the path or even in the destination storage which might not be able to keep up with data coming in at 50 Gbps, but still the impact on the shared network would be less and the achievable rate would be at least as good as the unpaced 100 Gbps case.

This chapter surveys the existing approaches in the Wide-Area Programmable Networks that could address some of the previously mentioned challenges. The focus is mainly on the following set of use cases:
- Traffic engineering - additional capacity is available in the existing networks and can be provisioned on-demand by steering traffic via alternate paths.
- Network provisioning - with DC networking moving towards WAN protocols and network virtualisation, there is an opportunity to leverage this to find alternative ways to organise/manage current HEP networks (L3VPNs/LHCONE)
- Provide QoS transfers - our current networking is usually segmented into private networks providing bandwidth guarantees and public/R&E networks. With other experiments coming online it can be expected that they come up with similar requirements and it would therefore be beneficial to look for synergies and technical approaches that can eliminate additional networks and operations.
- Improve network to storage performance - currently there is often a mismatch between target storage and network performance. In addition, inefficiencies arise when existing data transfer tools run on DTNs (NUMA I/O, scheduling overheads, caches, etc.).

- Capacity sharing between multiple domains is currently based on a first-come first-serve basis. Network as a resource is becoming likely in the future (like compute/storage today), but it depends on the network requirements (and real usage) of the new experiments coming online.
- Effective use of HPCs and Clouds across WAN as well as other opportunistic resources will be an important area. One of the major obstacles in fully utilising current HPCs are existing network limitations (mostly wrt. capabilities not capacity). (SEE euroHPC initiative/PRACE in EU)
- Autonomous networking (cyber-attacks/security relevant) and network telemetry

WAN programmable networks address many of the use cases outlined above and have the potential to change the way HEP sites and experiments connect and interact with the network, this chapter aims to highlight key projects in this area as well as provide an overview of the plans of the major R&E providers in the domain of orchestration, automation and virtualisation of WAN. As WAN programmable networks involve several major technologies the main focus of this chapter is to introduce the core concepts and projects as well as highlight key challenges in their adoption. This chapter is organised as follows:

- Programmable Networks for Data-Intensive Sciences introduces the basic concept of software-defined WAN (SD/WAN), software-defined exchanges (SDX), network orchestrators, provisioning systems and network-aware data transfer systems.
- Research & Education Networks Programmable Services surveys the current plans of the major R&E in the area of orchestration, automation and virtualisation of WAN.

Programmable Networks for Data-Intensive Science

Software-defined WAN (SD-WAN)

What is a Software-Defined WAN ?

The SD-WAN paradigm appears to come from the telecom industry. SD-WAN is defined as a specific application of software-defined networking (SDN) technology applied to WAN connections, such as broadband internet, 4G, LTE, or MPLS. The aim of an SD-WAN is to connect enterprise networks - including branch offices and data centers - over large geographic distances [SDWAN]. For HEP, enterprise networks can translate to a campus network or a research laboratory (facility) network. Branch office refers to a campus where mainly people (researchers/faculty, students, administrators) are located, as well as serve as a site for compute, storage and network resources. Data center is synonymous in use to HEP, as a physical site, in a cloud environment, or hybrid. A key characteristic of the definition is connecting the networks at all of these different sites over large geographic distances.

The term Wide-Area Network (WAN) is broadly used to describe a connection between an auxiliary campus site and a central campus site, where the WAN connection, for example, is a service provided by a metro-ethernet service provider. Likewise, a WAN connection could be between a campus participating in ATLAS or CMS, with a WAN connection to an LHC provider, such as

FermiLab. The difference between WAN and SD-WAN is that SD-WAN decouples the network from the management plane, and detaches the traffic management and monitoring from hardware.

What is the SD-WAN Standard?

The SD-WAN standard describes requirements for an application-aware, over-the-top WAN connectivity service that uses policies to determine how application flows are directed over multiple underlay networks irrespective of the underlay technologies or service providers who deliver them [SDWAN-standard].

A goal of the SD-WAN standard is to distinguish the components of the architecture that describe the SD-WAN Service and the Underlay Connectivity Services over which the SD-WAN operates [SDWANas]. An SD-WAN Service Provider provides the SD-WAN Service to the Subscriber. The SD-WAN Service Provider may provide one or more of the Underlay Connectivity Services, but some or all of them may be provided by the SD-WAN Subscriber or other Service Providers. The following is a description of SD-WAN from the MEF SD-WAN standard [SDWANas]:

An SD-WAN Service provides a virtual overlay network that enables application-aware, policy-driven, and orchestrated connectivity between SD-WAN User Network Interfaces, and provides the logical construct of a L3 Virtual Private Routed Network for the Subscriber that conveys IP packets between Subscriber sites.

An SD-WAN Service operates over one or more Underlay Connectivity Services (IP service, carrier ethernet, MPLS, etc). Since the SD-WAN service can use multiple disparate Underlay Connectivity Services, it can offer more differentiated service delivery capabilities than connectivity services based on a single transport facility.

An SD-WAN service is aware of, and forwards traffic based on, Application Flows. The Service agreement includes specification of Application Flows—IP Packets that match a set of criteria—and Policies that describe rules and constraints on the forwarding of the Application Flows.

A motivation for SD-WAN is its ability to adjust aspects of the service in near real time to meet business needs. This is done by the Subscriber by specifying desired behaviors at the level of familiar business concepts, such as applications and locations and by the Service Provider by monitoring the performance of the service and modifying how packets in each Application Flow are forwarded based on the assignment of policy and the real-time telemetry from the underlying network components.

Network Provisioning (multi-ONE)

This project aims to find technical solutions that can allow a computing site to separate the data traffic generated by the resources allocated for different research collaborations and then use dedicated VPNs similar to LHCONE.

LHCONE is a Virtual Private Network (VPN) that was conceived to connect WLCG Tier1 and Tier2, a relatively small number of trusted data-centres that share common security practices. Its main advantage is that it can be connected directly into the data-centres bypassing slow security border

devices. Today LHCONE has reached a critical number of connected sites, at a point where trust may start to decrease and thus undermining the most valuable characteristic it has.
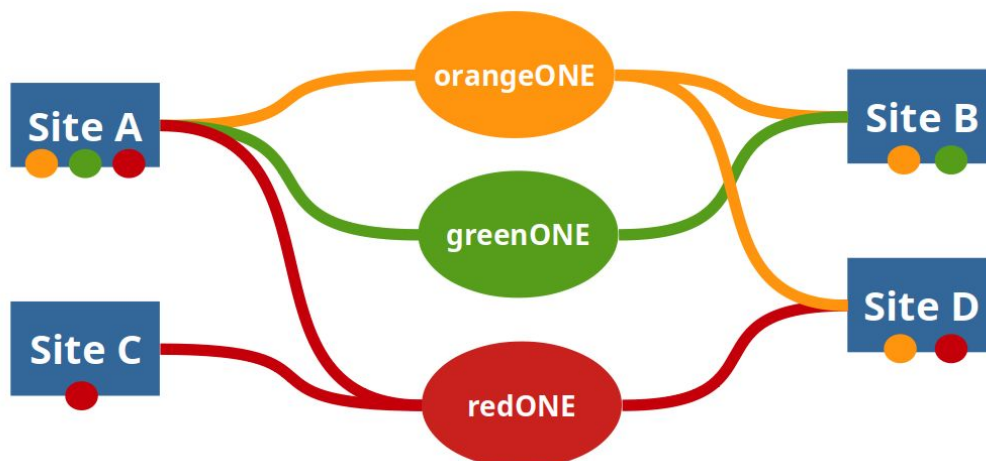
Here is a list of a few possible solutions that could allow the traffic separation. The list is not exhaustive and doesn't exclude other ideas. Individual sites can use different solutions, as long as the objective of routing traffic into the correct VPN is achieved.

- Dynamic, software defined allocation of computing resources: using network virtualization techniques (VXLAN, EVPN, Linux namespaces, etc.), create virtual groups of computing resources that interface with the network to separate the traffic at the sources.
- Paired use of source-destination address: making use of secondary IP addresses, computing resources should make sure to use appropriate source-destination address pairs, to allow simple destination routing.
- Packet tagging: using available fields in the IP header (DSCP, flow label..) set by the applications, make the network policy routes those packets in the correct VPN.

In case of solutions based on mapping by IP addresses, it would be necessary for every site to allocate dedicated IP prefixes to the different collaborations. Since IPv4 is a scarce resource, it is envisaged that the proposed solutions have to work with IPv6.

DUNE and protoDUNE have been proposed as a collaboration that can allow CERN, FNAL and other interested sites to prototype and test these solutions.

Network Service Interface (NSI)

NSI is a web-service based API that operates between a requester software agent and a provider software agent. The full suite of NSI services allows an application or network provider to request and manage circuit service instances. Apart from the Connection Service these include the Topology Service and the Discovery Service. The complete set of NSI services is described in the Network Services Framework v2.0.

This Connection Service document describes the protocol, state machine, architecture and associated processes and environment in which software agents interact to deliver a Connection. A

Connection is a point-to-point network circuit that can transit multiple networks belonging to different providers.

Software defined exchanges (SDX)

Currently there is no single agreed upon definition of what Software-Defined Exchange (SDX) means. The concept of an SDX (from an academic perspective) was first introduced by Gupta et al, motivating the role of Internet Exchange Points (IXPs) as a compelling place to introduce SDN for wide-area traffic delivery by offering direct control over packet- processing rules that match on multiple header fields and perform a variety of actions. To further motivate the SDN community, a report from the NSF workshop on Software Defined Infrastructures and Software Defined Exchanges widened the scope to position SDX in the context of Software Defined Infrastructure (SDI). A taxonomy for SDX was published by Chung et al to describe a classification of SDX and their implementations by organizations and projects. This taxonomy defined Layer-3 SDX, which exchanges BGP routes in Internet Exchange Points; Layer-2 SDX, which exchange multi-domain ethernet circuits in R&E networks; and SDN SDX, which interconnect SDN islands.

For HEP, where R&E networks are used to interconnect research facilities across long geographics distances, having SDXs in the path can diminish, or even eliminate, some of the challenges provisioning end-to-end circuits between research facilities. Network operators can take from days to several weeks to provision an intercontinental end-to-end circuit over multiple R&E networks, because these tasks are often done manually [SDNAmLight]. Adding SDN to the provisioning process can significantly reduce provisioning costs in time and effort of science network services [AdvRes].

HEP can benefit from the advanced networking capabilities of an SDX that is underpinned by SDN even if the R&E networks connected to the exchange point are not SDN enabled. In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications [ONFwp]. Since an SDX is logically centralized, controllers have global visibility of the whole network topology, unlike conventional networking, making it possible to maintain related states for a full path of a flow. Hence, an SDX can dynamically optimize flow-management and resources. Furthermore, per-flow or application-level QoS provisioning becomes easier and feasible for network administrators [QoS].

The U.S. National Science Foundation (NSF) International Research Network Connections (IRNC) program supports three projects to build and operate SDXs that connect the U.S. R&E networks to Europe, Asia-Pacific, and South America. References for these three projects are: StarLight SDX: A Software Defined Networking Exchange for Global Science Research and Education; AtlanticWave-Software Defined Exchange: A Distributed Intercontinental Experimental Software Defined Exchange (SDX); Pacific Wave Expansion Supporting SDX & Experimentation. A detailed description of these projects, their goals and accomplishments can be found in the Appendix.

Orchestrators

SENSE

The Software-defined network for End-to-end Networked Science at Exascale (SENSE) system is a model-based orchestration and resource management system which enables end-to-end multi-domain and multi-resource Layer 2/3 network services instantiation. The SENSE system includes an Orchestrator, a Network Resource Manager, and an End-Site Resource Manager which can be deployed in flexible ways to adapt to a variety of site and infrastructure deployments. SENSE allows the science applications to manage the network as a first-class schedulable resource in a manner similar to instruments, compute, and storage. SENSE operates between the SDN layer controlling the individual networks/end-sites, and science workflow agents/middleware. The SENSE system defines the mechanisms needed to dynamically build end-to-end deterministic and policy-guided Layer 2/3 network services. An intent-based interface allows applications to express service requirements in a high-level domain science specific context, with mechanisms for interactive and full-service lifecycle coordination with workflow automation systems. The SENSE system has two key objectives:

- Facilitate science workflow related provisioning and life-cycle management for a variety of end-to-end network services.
- Provide the intelligence and control to enable more optimal and efficient use of network infrastructures.

The SENSE approach to end-to-end at-scale networking is based on software programmability and intelligent service orchestration. These are enabled by some novel technologies, including a) hierarchical service-resource architecture, b) unified network and end-site resource modeling and computation, c) model based real time control, d) application driven orchestration workflow, and e) future plans for end-to-end network data collection and analytics integration.

There are four main functions of the SENSE system:
- SENSE Orchestrator North Bound Interface - A highly customizable interface for application workflow agents to query regarding possible actions, recommendation, and/or request specific service instantiation. While a standard northbound interface has been defined, this interface is designed to be easily and rapidly customized for individual user requirements. The SENSE system has significant amounts of data and intelligence regarding the underlying networked systems. This information can be customized for user consumption in a highly detailed or abstract manner.
- SENSE Orchestration - This includes the integration of resource model-based descriptions from the underlying network infrastructures, the computation services to process resource models to respond to user requests, and the coordination of provisioning actions.
- SENSE Orchestrator South Bound Interface - Provides for a continuous exchange of topology descriptions which include an ability for the resource owners to tailor the level of abstraction and realtime states in accordance with local policies and service objectifies. This is one of the key innovations of the SENSE system and is based on semantic

> web-based graph models which provides for a high degree of service flexibility and infrastructure owner-controlled customizations.

- Multi-Resource "SDN" Layer - The SENSE architecture does rely on an underlying SDN layer, however it does not require a particular SDN controller or system implementation. The SENSE architecture accepts that there will be a variety of deployed SDN solutions which will cover specific network and administrative regions. The SENSE system provides the mechanisms and infrastructure to leverage these systems and provide guidance for how they can be fully integrated into the SENSE system. One method is for existing SDN systems to implement the SENSE Orchestrator Southbound Interface as their controller Northbound Interface. Existing systems may realize this via native implementation of the SENSE API or via a thin layer on top of their existing API which provides the proper interface. This thin SENSE layer can actually be embedded into the SENSE Orchestrator, such that no changes are required to many existing SDN systems. Adapting to underlying SDN systems for SENSE system integration has been used successfully as part of the SENSE system deployment on ESnet and other R&E infrastructures. Systems based on OpenDaylight (ODL), Network Services Interface (NSI), On-Demand Secure Circuits and Advance Reservation System (OSCARS), and Open Network Operating System (ONOS) have all been integrated into SENSE orchestrator operations. The SENSE development and testing activities have demonstrated that valuable orchestrated services can be provided using these existing SDN systems as they are with no internal modifications. In addition, more advanced functions can be enabled via use of a native SENSE API implementation. To demonstrate these more advanced features, SENSE has developed a "Network Resource Manager" and "End-Site Resource Manager".   These resource managers utilize the SENSE native model-based API to realize advanced features in the areas of multi-resource integration, real time responsiveness, computation to support negotiations, and user interactions.

Additional information regarding the SENSE system is available via the SENSE INDIS Paper and SENSE WebSite.

NOTED

NOTED (Network-Optimized Transfer of Experimental Data) is a project led by CERN which aims to improve the efficient use of WAN networks and accelerate the transfers of large amounts of data between remote data centres.

Engineering the traffic to avoid a chokepoint is not complicated to implement when traffic volumes and end points can be predicted. Unfortunately it's not easy to understand how the traffic could be optimized by just looking at the network, because large flows can end just seconds after they have been identified.

NOTED will implement a Transfer Broker, a framework that can understand when a large data transfer has started and publish such information for the use of network controllers.
The Transfer Broker will query transfer services like Rucio and FTS to understand when a large data transfer has started, the total amount of data that will be exchanged, between which sites, the percentage of transfer completed. It will make this information available over a programmable

interface, to allow network controllers to decide when and where to implement network optimizations.

NOTED will also implement a Network Controller that can query the Transfer Broker and implement network optimizations using SDN.
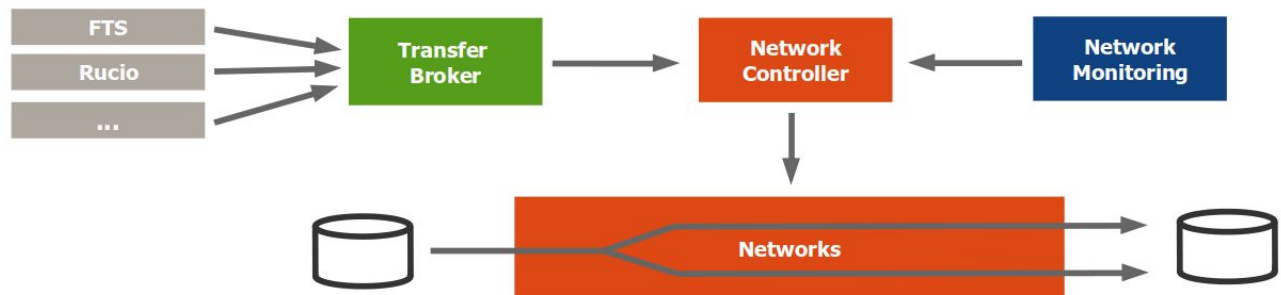
Data Transfer Services

BigData Express

BigData Express (http://bigdataexpress.fnal.gov) is a U.S. Department of Energy (DOE) funded network research project. It seeks to provide a schedulable, predictable, and high-performance data transfer service for big data science.

    A. System architecture and design

BigData Express will typically run in a data center. As illustrated in Figure 1, a typical site will feature a dedicated cluster of high-performance Data Transfer Nodes (DTNs), an SDN-enabled LAN, and a large-scale storage system.

BigData Express optionally requires an on-demand site-to-site WAN connection service to provide the path(s) between source and destination sites. Normally, the WAN service supports guaranteed bandwidth and designated time slot reservations. ESnet and Internet2 currently are capable of providing such a WAN service via OSCARs and AL2S, respectively. This requirement is necessary for BigData Express to establish end-to-end network paths with guaranteed QoS to support real-time and deadline-bound data transfer. Otherwise, BigData Express would provide best-effort data transfer.
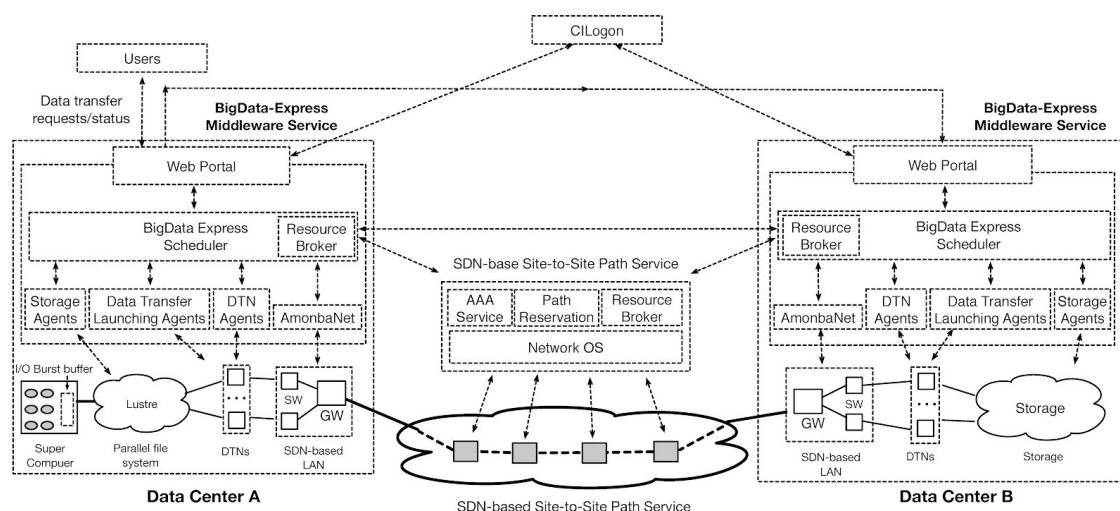
Figure 1 BigData Express Architecture

BigData Express adopts a distributed, peer-to-peer model. A logically centralized BigData Express scheduler coordinates all activities at each BigData Express site. This BigData Express scheduler manages and schedules local resources (DTNs, storage, and the BigData Express LAN) through agents (DTN agents, storage agents, and AmoebaNet). Each type of resource may require one or multiple agents. The scheduler communicates with agents through a MQTT-based message bus. This architecture offers flexibility, robustness, and scalability. BigData Express Schedulers located at different sites negotiate and collaborate to execute data transfer tasks. They execute a

distributed rate-based resource brokering mechanism to coordinate resource allocation across autonomous sites.

Web Portal allows users and applications to access BigData Express services. For a data transfer task, the following information will be conveyed to BigData Express via Web Portal: X.509 certificates of the task submitter, the paths and filenames of the data source, the paths of the data destination, the task deadline, and the QoS requirements. BigData Express uses this information to schedule and broker resources for the data transfer task, then launch data transfers. Web portal also allows users to browse file folders, check the data transfer status, or monitor the system/site status.

DTN agents collect and report the DTN configuration and status. They also assign and configure DTNs for data transfer tasks as requested by the BigData Express scheduler.

AmoebaNet keeps track of the BigData Express LAN topology and traffic status with the aid of SDN controllers. As requested by the BigData Express scheduler, AmoebaNet programs local networks at run-time to provide custom network services.

Storage agents keep track of local storage systems usage, provide information regarding storage resource availability and status to the scheduler, and execute storage assignments.

Data Transfer Launching Agents initiate data transfer jobs as requested by the BigData Express scheduler. Typically, Data Transfer Launching Agents launch 3rd party data transfers between DTNs using X.509 certificates on behalf of users. Data transfer launching agent features an extensible plugin framework that is capable of supporting different data transfer protocols, such as mdtmFTP, GridFTP, and XrootD.

The BigData Express scheduler implements a time-constraint-based scheduling mechanism to schedule resources for data transfer tasks. Each resource is estimated, calculated, and converted into a rate that can be apportioned into data transfer tasks. The scheduler assigns rates for data transfer tasks in the following order of priority: real-time data transfer tasks à deadline-bound data transfer tasks à background data transfer tasks.

B.   A High-performance Data Transfer Engine

mdtmFTP is BigData Express' default data transfer engine. It offers high-performance data transfer capabilities.

mdtmFTP achieves high performance through several key mechanisms. First, mdtmFTP adopts a pipelined I/O centric design. A data transfer task is carried out in a pipelined manner across multiple cores. Dedicated I/O threads are spawned to perform network and disk I/O operations in parallel. Second, mdtmFTP utilizes the MDTM middleware services to make optimal use of the underlying multicore system. Finally, mdtmFTP implements a large virtual file mechanism to address the Lots of Small Files (LOSF) problem. Evaluations have shown that mdtmFTP achieves higher performance than data transfer tools such as GridFTP, FDT, and BBCP.

mdtmFTP supports third-party data transfer. It also supports GSI-based security. Figure 2 illustrates a BigData Express data transfer example. A Data Transfer Launching Agent launches a third-party data transfer between two DTNs using X.509 certificates.
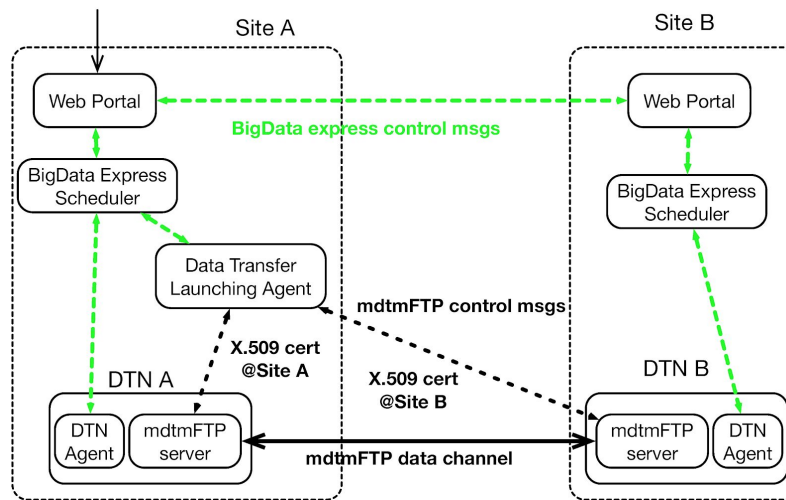
Figure 2 BigData Express launches data transfer jobs

C. On-Demand Provisioning of End-to-End Network Paths with Guaranteed QoS

BigData Express intelligently programs network at run-time to suit data transfer requirements. It dynamically provisions end-to-end network paths with guaranteed QoS between DTNs. An end-to-end network path typically consists of LAN and WAN segments. In BigData Express end-to-end data transfer model, LAN segments are provisioned and guaranteed by AmoebaNet, while WAN segments are provisioned through on-demand WAN path services such as ESnet OSCARS, or Internet2 AL2S to provide paths between the data source and destination sites.

AmoebaNet applies SDN technologies to provide "Application-aware" network service services in the local network environment. It offers several capabilities to support BigData Express operations. To support network programmability, AmoebaNet provides a rich set of network programming primitives to allow BigData Express to program the local area network at run-time. To support QoS guarantees, AmoebaNet provides two classes of services, *priority* and *best-effort*. Priority traffic flows are typically specified with designated rates or bandwidth. AmoebaNet uses QoS queues to differentiate priority and best-effort traffic at each SDN switch. Priority traffic is transmitted first, but metered to enforce rate control. In addition, AmoebaNet supports QoS-based routing and path selection. Finally, AmoebaNet supports fine-grained control of network traffic.

WAN QoS can be provisioned and guaranteed by utilizing ESnet OSCARS, or Internet2 AL2S to reserve bandwidths between Service Termination Points (STPs), where AmoebaNet services end.

Typically, AmoebaNet gateways (GWs) are either logically, or physically connected to WAN STPs. VLAN popping, pushing, and/or swapping operations are performed at AmoebaNet gateways to concatenate WAN and LAN segments.

As illustrated in Figure 3, BigData Express typically performs the following operations to provision an end-to-end network path:

1)   Estimate and calculate the DTN-to-DTN traffic matrix, and the related QoS requirements (e.g. throughput, delay).

2)   Negotiate and broker network resources to determine the end-to-end rate for the path.

3)   Call ESnet OSCARS or Internet2 AL2S circuit service to set up a site-to-site WAN path.

4) Call AmoebaNet at each site to program and configure the LAN paths.

5) Send PING traffic to verify a contiguous end-to-end network path has been successfully established.

A large data transfer job typically involves many DTNs, and a corresponding large number of data flows. To avoid the necessity of establishing many WAN paths between the source and destination sites, multiple LAN segments can be multiplexed/de-multiplexed to/from a single WAN path, which in turn is configured to support the aggregated bandwidth of its component paths. This strategy helps to reduce burden on WAN path services.
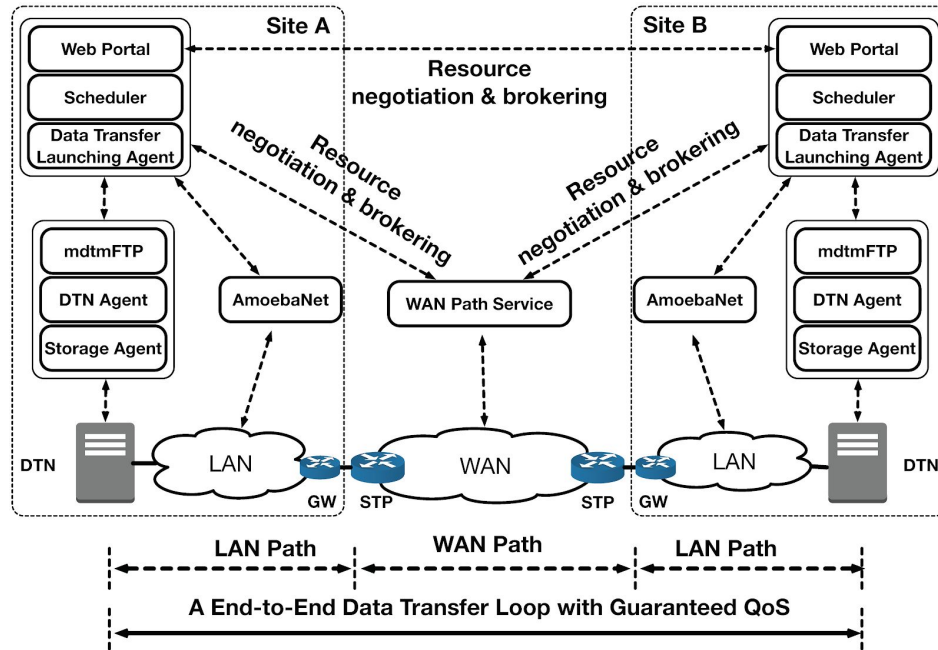
Figure 3 Provisioning of end-to-end path with guaranteed QoS

D. Security

BigData Express runs in secure environments. At each site, BigData Express systems run in trusted security zones protected by security appliances. All DTNs are secured by using X.509 certificates. All BigData Express sits use a common single-point sign-on service (CILogon) to obtain X.509 certificates for secure access to DTNs. In addition, each site publishes its public key so that different sites can establish trust. Communication channels between two sites are secured by HTTPS.

Users are authenticated and authorized to access BigData Express services. From a user's perspective, BigData Express provides two layers of security:

(1) A user must first use his/her username and password to login to a particular BigData Express web portal. Once login is successful, a user can manage data transfer tasks (submission, cancellation, and monitoring), or monitor the system/site status.

(2) Within a logged-in web portal, a user must further login to data transfer source and/or destination site(s) to obtain X.509 certificates for secure access to local DTNs. Once authenticated locally, the user can browse files, and/or launch data transfer tasks. With

CILogon issued X.509 certificates, the BigData Express scheduler will request Data Transfer Launching Agents to launch data transfer tasks on behalf of the user.

The BigData Express software is currently deployed and being evaluated at multiple research institutions, including UMD, StarLight, FNAL, KISTI, KSTAR, SURFnet, and Ciena. The BigData Express research team is deploying BigData Express on various research platforms, including Pacific Research Platform, National Research Platform, and Global Research Platform.

Research & Education Networks Programmable Services

ESnet6

ESnet is the High-Performance Network (HPN) user facility for the DOE Office of Science. As the current ESnet network reaches almost 10 years of operations, the organization is in the middle of a significant upgrade to its network that is being executed in parallel.

ESnet6 Architecture for programmability

The ESnet6 network was conceived with three main objectives in mind:
1. **Capability**: Meeting the bandwidth requirements of the DOE Office of Science research mission. With the current exponential growth that ESnet is experiencing, ESnet staff project that the network will carry approximately 1 exabyte of traffic per month in the FY 2021-2022 timeframe.
2. **Reliability**: Addressing the resiliency requirement of scientific workflows. With increasing reliance on the network to support data movements ranging from bulk data transfers to low-latency remote control applications, the network is a critical resource in the larger application workflow that includes instrument, compute, and storage elements.
3. **Flexibility**: Enabling the evolution of network services. With the increasing need for data management within complex workflows involving distributed resources (e.g., instruments, local computing, remote high performance computing (HPC), storage, and networks), it is necessary for network services to evolve from simply providing best-effort connectivity to higher-level intelligent services.

To achieve these objectives, ESnet6 network was designed with five overarching goals:
1. *No single point of failure*. The primary consideration for this design goal was to ensure that the physical network infrastructure met this objective. The single-point-of-failure protection also extends to the network elements deployed within the ESnet6 network. This entails properties such as redundant routing engines, switching fabrics, and power supplies. In addition, management of compute processes (e.g., Virtual Network Functions (VNF)) can be run in a high-availability (HA) fashion, utilizing local compute clusters with software to manage process migration.
2. *Agility to add and/or move bandwidth capacity*. Having the capability to dynamically add and/or move bandwidth capacity within the network greatly increases the ability to address short-term capacity over-subscription and long-term growth.
3. *Fine-grained traffic engineering*. The ability to perform fine-grained traffic engineering at scale within the wide-area network is a crucial building block for many advanced network services. In addition to supporting data transfers that require predictable service guarantees (e.g., deadline scheduling, real-time remote control, etc.), fine-grained traffic engineering can be used to run the network more efficiently by implementing network-wide, optimized load-balancing and global (vs. local) resiliency strategies.
4. *Comprehensive automated management*. As the complexity of the network increases, manual management at scale becomes exponentially more difficult and untenable.

Automated functions are needed to administer the network beyond human-scale limitations, as well as provide a basis for more intelligent proactive management. The use of abstraction and modeling can facilitate intent-based management to simplify administration by allowing engineers to dictate what should be done, instead of how it is done.

5. *Highly programmable and flexible services*. As scientific workflows become more complex, there is an increasing dependency on the network to provide more intelligent services beyond best-effort connectivity. In a traditional Internet Service Provider (ISP) environment, the creation of new protocols or software features needed to support advanced services is typically gated by what commercial vendors are willing to develop. Therefore, it is astute to augment the network with flexible programmable platforms, such as P4 switches or FPGAs, to enable the development of new services that cannot be run on more traditional ASIC-based network equipment.

Apart from a fundamental change in the network architecture from the ESnet5 to ESnet6, there is an intentional focus on comprehensive orchestration and automation for ESnet6. As part of this effort, ESnet has formalized a software architecture which dictates how both in-house developed, and purchased software will interoperate and integrate within this software framework. The ESnet6 software architecture covers five main areas necessary to the day-to-day operations of the ESnet network, these include;

i. Assurance (e.g., fault management, root cause analysis, incident management),
ii. Provisioning (e.g., workflow orchestration, automation provisioning),
iii. Security (e.g., audit logging, black hole routing),
iv. Analytics (e.g., time series data, streaming telemetry, traffic planning), and
v. a unified data model for network intent, discovery, and topology.

"High-Touch" services

ESnet6 is implementing a new class of programmable services called "High-Touch". These services will motivate the deployment of programmable data plane hardware like P4 switches, NPUs, and/or FPGAs into the network, along with associated compute and storage platforms. While many ideas can be implemented using this platform, from NFV-like services to in-network caching, the first service ESnet is working on is Precision Network Telemetry described below.

Precision Network Telemetry

Precision Network Telemetry is a key new capability being developed as part of the High-Touch services portfolio that will be present in ESnet6. As a network operator, ESnet will be able to select flows based on a variety of criteria, such as L3VPN, L2VPN, and IP 5 tuples within these VPNs to select anything from a single data transfer, to groups of transfers between a range of sites.   For any of these flows, it will be possible to monitor every packet header in the flow.

With the capability to analyze each packet, along with the augmentation of precision timestamps of packet arrival times, obtaining per flow detailed statistics becomes a reality. This in turn opens a myriad of possibilities into understanding how data transfers are performing (e.g., real-time auditing of packet loss per flow, monitoring throughput vs goodput), discerning usage patterns and trends

(e.g., mapping data movement across the entire data lake), and developing models to characterize distributed workflows (e.g., correlating per flow information of data movements with job submissions). Translating this deep visibility into a useful tool for HEP workflows will take iterative dialogue and joint work, especially to understand what user experience needs to be improved vs. what can be improved.

It is critical to understand that developing relevant High-Touch services will require a level of engagement and collaboration between HEP and ESnet. It is important to have representatives who understand every detail of HEP technologies like data lakes, XrootD, Rucio and compute schedulers, along with ESnet counterparts who understand what level of detail ESnet's high precision telemetry can expose, in order to design a solution that works and has real benefit to HEP.

In-network caching/computing

In-network caching and in-transit computing services: In-network services such as temporary data caching and computing could potentially have a big impact on how the remote data is being accessed and processed while in-transit. For example, the amount of data volume moving through the network gets increased with newer scientific experiments and simulations, and network bandwidth requirements also gets higher proportionally to deliver the data within a certain time frame. We observe that a significant portion of the popular dataset is transferred multiple times to different users as well as to the same user due to various reasons including lack of accommodating storage volume. Sharing data among users and multiple jobs will reduce the redundant data transfers and save network traffic volume consequently. Sharing data per site may be able to be accomplished by the large storage on the site. However, sharing data among geographically distributed users can only be accommodated with some kind of content-delivery network. We experiment how in-network caching mechanism helps network traffic performance and application performance, how much data can be shared within the network, how much network traffic volume can be reduced consequently, and how much efficiency the temporary in-network cache gives to the application performance. We expect significant network traffic savings by sharing data and overall performance improvement in applications because the access to the shared data has less latency of the network cache. In our preliminary study with a ~14TB dataset for multiple analysis jobs, about 60% of the dataset is cached in the end, and about 19TB of network traffic is saved with the shared data among the analysis jobs. Shared data caching mechanism is expected to reduce the redundant data transfers and saved network traffic consequently. There will be further studies on the capabilities of in-network data caching mechanism and in-transit computing to provide better services for the HEP community.

FABRIC

FABRIC is a unique research infrastructure to enable cutting-edge and exploratory research at-scale in networking, cybersecurity, distributed computing and storage systems, machine learning, and science applications. It is an everywhere programmable nationwide instrument composed of novel extensible network elements equipped with large amounts of compute and storage, interconnected by high speed, dedicated optical links. It will connect a number of

high-performance computing facilities and specialized testbeds (5G+/IoT, NSF Clouds) to create a rich fabric for a wide variety of experimental activities. It will create the opportunities to explore innovative solutions not previously possible and will provide a platform on which to educate and train the next generation of researchers on future advanced distributed systems designs.

This National Science Foundation (NSF) funded project creates a nation-wide high-speed (100-1000 Gigabits per-second) network interconnecting major research centers and national computing facilities that will allow researchers and scientists at these facilities to develop and experiment with new distributed application, compute and network architectures not possible today. FABRIC nodes can store and process information "in the network" in ways not possible in the current Internet, enabling research into completely new networking protocols, architectures and applications that address pressing problems with performance, security and adaptability in the Internet. Reaching deep into university campuses, FABRIC will connect university researchers and their local compute clusters and scientific instruments to the larger FABRIC infrastructure. The figure below shows the planned FABRIC deployment with initial nodes being available by the end of 2020. The FABRIC network will include a set of core nodes deployed at ESnet6 points of presence, and multiple edge nodes deployed at a variety of regional network, campus, and scientific facility locations.

The FABRIC infrastructure is not intended to be an isolated testbed, but will serve as a programmable interconnect fabric to a variety of experimental and production resources, allowing the community to deploy novel applications that range from a one-time wide-scale deployments with highly experimental software to a near-production-ready persistent experimental service serving a wide range of users from campuses, industry and labs. In addition to the network embedded compute, storage, and data plane programmability, the testbed will include Layer 3/2 peering with the Research and Education (R&E) infrastructure such that experiment topologies can access production resources and data sets.   FABRIC will also support hybrid infrastructures interconnecting FABRIC's physical network infrastructure with cloud-based virtual end systems or clusters. This will include dedicated Cloud Connect Services including Amazon AWS Direct Connect, Google Dedicated Interconnect, and Microsoft ExpressRoute between FABRIC core

nodes and commercial cloud providers. This will allow the virtual networks, virtual machines and application stacks in these cloud providers to "peer" with the FABRIC network to form hybrid systems to support the development of customized application workflow functions.

The High Energy Physics (HEP) experiment community could leverage the FABRIC infrastructure to enhance, validate and test pre-production workflows at scale associated with LHC data distribution, computation, and analysis. The network embedded compute, storage, and data plane programmability can be combined with access to NSF supercomputer centers (like TACC), R&E production networks (like ESnet) and Public Cloud infrastructures and data repositories to facilitate research and development into new scientific methods and paradigms.

The FABRIC team is led by UNC-Chapel Hill RENCI (Renaissance Computing Institute) and includes the University of Kentucky, the Department of Energy's Energy Sciences Network (ESnet), Clemson University and the Illinois Institute of Technology. Contributors from the University of Kentucky and ESnet will be instrumental in designing and deploying the platform's hardware and developing new software. Clemson and Illinois Institute of Technology researchers will work with a wide variety of user communities—including those focused on security, distributed architectures, scientific applications and data transfer protocols—to ensure FABRIC can serve their needs. In addition, researchers from many other universities will help test the platform and integrate their computing infrastructure and scientific instruments into FABRIC.

GEANT Higher-level Services

GÉANT community survey on orchestration, automation and virtualization

In 2019 the GN4-3 project executed a survey to learn more about the current status and plans of NRENs (National Research and Education Networks) in the GÉANT community for adopting and implementing orchestration, automation and virtualization (OAV) principles [D6.2]. Main goals of the initiative were as follows:

- Learn about the strategy and actions of each NREN related to network and service orchestration, automation and virtualisation (OAV).
- Explore if there are common use cases, ideas, needs and issues in the community in the areas of automation, orchestration and virtualisation.
- Recognise possible areas of collaboration both amongst NRENs and between NRENs and GÉANT.
- Determine and recommend possible future work of the project that could be of benefit to as many partners as possible for identified use cases.

The survey participants have shown a high level of interest in OAV. The benefits of adopting OAV in NRENs are very similar to those seen by commercial communication service providers or content providers and includes:

- Faster service delivery / reduced delivery time.
- Reduced human error and manual work.
- Lower service delivery costs.
- Better reporting.

- Increased efficiency.
- Ensured and increased configuration uniformity and consistency.

There is a wide variety of components and systems being used by NRENs. These are mostly OSS, with rather fewer BSS, and very little OSS-BSS integration (whether proprietary, in-house or open source). Further, there is no obvious common single direction or best practice for OAV, and currently little use of standard data models and APIs that facilitate OAV.

A wide variety of use cases were reported. Connectivity services were of very broad interest to most if not all NRENs, with configuration integrity ranking highest of the application areas, followed by problem troubleshooting and security operations centre enhancements. While most NRENs do not yet have inter-domain OAV, NRENs were mostly interested in the following cross-domain use-cases: NREN and a campus (42%), NREN-GÉANT network (35%) and then NREN-cloud (23%). Interestingly, some 80% of NRENs would be willing to allow changes to their configurations resulting from requests initiated in other domains, assuming the agreed policies, procedures, authentication and authorisation are in place.

GÉANT Testbeds Service

The [GÉANT Testbeds Service](#) (GTS) was initially developed to allow researchers to set up individual testbed environments in an automated fashion. However, the word 'testbed' is misleading as GTS is a general provisioning system that allows users or network engineers to automatically set up any virtual network for any purpose.
GTS is based on the concept of a Generalized Virtualization Model (GVM) which allows networks to be sliced in an abstracted way where abstracted virtualized objects, called resources, can be defined, instantiated and arranged to create application-specific insulated networks. GVM defines this set of service objects (drawn from the NSI standards) to create virtual objects or resources and describe their data flows through their life-cycles. Everything in GVM is considered a resource, i.e. a resource could refer to computational facilities, transport circuits, custom switching, forwarding capabilities or integrated storage and all resources are building blocks for dynamically provisioned networks.

In GVM the purpose of a resource, its function or behavior is not constrained by the GVM architecture. GVM simply describes how types or classes of resources are defined and how they can be reserved and managed through their life-cycles and how their data flow relationships can be structured and controlled. Since a resource in GVM can represent anything, GVM is technology agnostic; in other words, it does not promote a certain underlying physical infrastructure or hardware technology to be a resource or service capability.

This virtualisation of service objects in GTS allows the automation of resource provisioning: Users can construct network environments using these virtual building blocks via a web-page and the GTS software-based resource manager then coordinates and orchestrates the setup of the virtual network environments.

Available resource building blocks in GTS are currently Virtual Machines (VMs), Virtual Circuits (VCs), Virtual SDN switches (which are mapped to completely virtualized hardware switch

instances (VSIs) and Bare Metal Servers (BMSs). Different resource classes can be defined at the virtualization layer representing certain types of physical resources and software components. Software modules referred to as Resource Control Agents (RCAs) are responsible for instantiating these abstract resource classes in the physical infrastructure.

GTS therefore allows the integration of any type of resource for automatic provisioning and integration into the system such as science instruments (e.g. radio telescopes, gene sequencers, etc.), licensed mobile spectrum or sensors, as long as there is an RCA available that maps the virtual service object to the underlying infrastructure. The concept of abstract service objects extends to fine grained functional capabilities such as traffic handlers (rate shapers, policers, load balancers, filters, IDS functions, etc.) in the emerging field of network function virtualization (NFV).

GTS and its underlying GVM architecture were developed in parallel to the ETSI NFV industry specification group's virtualisation model [ETSI] and both models show many overlapping components: The ETSI-NFV orchestrator can be compared to the GUI of GTS, and the ETSI Service, VNF and Infrastructure description corresponds to the DSL code used in GTS to describe a virtual network's desired topology. ETSI-NVF Virtual Computing and Computing Hardware components resemble RCA-VM and RCA-BMS components in GVM; similarly the Virtual Network component matches the RCA-VC for Virtual Circuits.

AmLight Express and Protect (AmLight-ExP)

AmLight-ExP is a project with support from the U.S. National Science Foundation (NSF) International Research Network Connections (IRNC) program, in collaboration with project partners Florida International University, AURA, LSST, RNP, ANSP, Clara, REUNA, FLR, Telecom Italia Sparkle, Angola Cables, and Internet2.
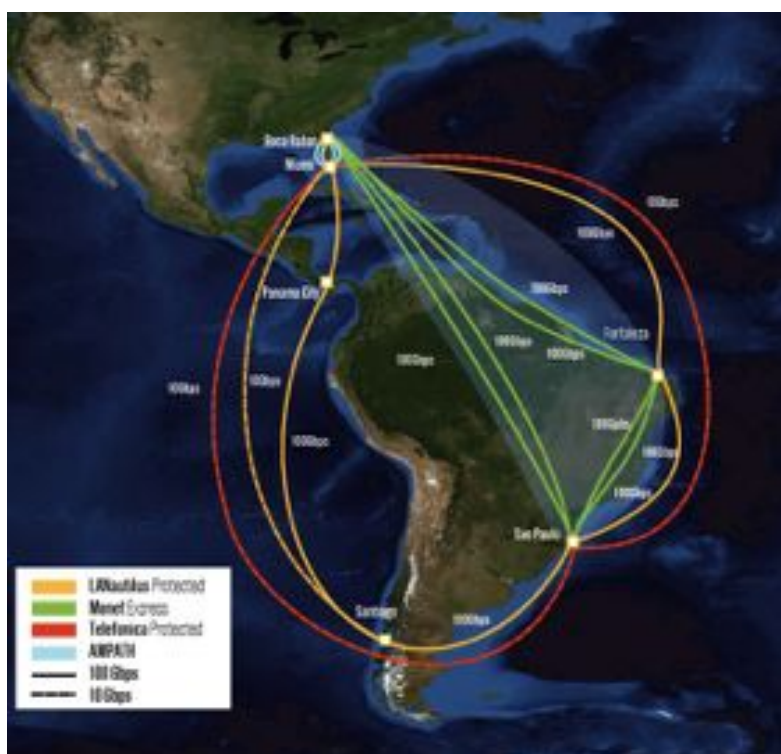
This figure represents the topology of the AmLight-ExP network infrastructure. The Express Ring from Boca Raton to Fortaleza to Sao Paulo consists of six (green lines) x 100G links (four managed by RNP and two managed by FIU/ANSP/LSST). The 100G Protect Ring from Miami to Fortaleza, Fortaleza to Sao Paulo, Sao Paulo to Santiago, Santiago to Panama, and Panama to Miami has one x 100G link (solid orange). The 10G ring from Miami to Sao Paulo and

50

Sao Paulo to Miami for protection has one x 10G link (red dashed). There is an additional 10G link Miami to Santiago for protection (orange dashed). The 100G and 10G rings are diverse, operating on multiple submarine cables. Current total upstream capacity presently is at 630Gbps. The total aggregated capacity of all the segments is 1,230 Gbps / 1.23 Tbps. All the links are configured to support Jumbo frames, OpenFlow flow entries, IPv4/6, and multicast.

Support for Network Virtualization and Programmability on AmLight-ExP
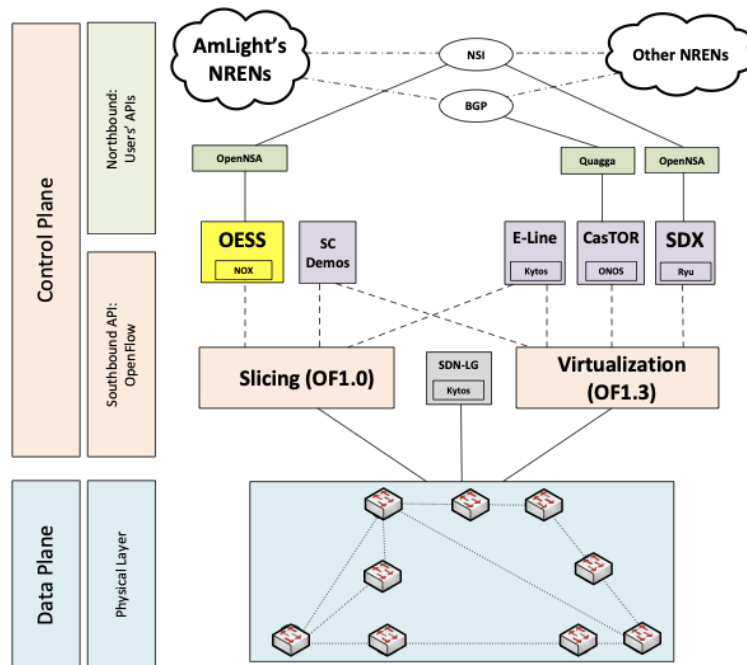
AmLight has been supporting SDN in production since 2014. Researchers may use slicing or virtualization to prototype network-aware applications. Researchers may implement testbeds with real network devices. They can validate their research in a production environment, and at scale. In the figure above, the boxes in lavender to the right of the yellow OESS box represent testbeds. The green boxes above are applications that use the northbound API of the SDN applications below in the figure. For example,the production SDN controller OESS uses OpenFlow to connect to the DataPlane and provides a REST API to be used by OpenNSA. For interdomain connections, BGP for Layer 3 and NSI for Layer 2 are supported.

Challenges and Outlook

Programmable WAN is still an area of intensive research and development and while the existing projects have well defined scope and good match to the HEP use cases, there are still a number of challenges that remain:

- One of the core challenges for some time is the fact that it appears to be difficult to bring the existing projects from testbed/prototype stage into production. Within LHCONE R&D efforts, a number of projects were successfully demo-ed in the past, but it has proven to be very challenging to deploy them in the production infrastructure. What appears to be missing are network infrastructures where prototypes can be tested at scale and then easily deployed/migrated to production.
- While the situation differs between R&Es, there is currently a significant lack of telemetry and monitoring data that can be programmatically accessed via APIs. This makes it difficult to develop systems that would improve efficiency in using the existing capacities (avoiding spikes in traffic when multiple experiments transfer large amounts of data at the same time).
- The lack of available telemetry, tracing and insight into how the current network operate is to some extent currently balanced out by deployment and operation of a complex network of end-to-end testing infrastructure (perfSONAR), but this is only applicable to a limited set of use cases (mostly to debugging/tracing of end-to-end network performance issues).
- From the perspective of data transfer systems, the concept of a transfer broker that could provide planned/anticipated transfers across multiple experiments is currently missing. It should be relatively easy to accomplish this provided that experiments would agree on ways to publish their planned/anticipated or even real-time transfers.
- As another alternative, automated methods for traffic engineering that would automatically adapt to the existing workloads have been proposed by different projects (both in SDX and orchestrators). Such systems promise to keep the existing status quo where the state of the underlying network and its operations are transparent to the experiments. It remains to be seen if such approaches will be feasible in a large scale federated environment (such as LHCONE).
- There is a wide range of functionalities and services planned and it can be expected that significant variations will exist between different R&Es. It's less clear how synergies will be found and how software defined networks will be offered across R&E boundaries.
- As was mentioned in the previous chapter, DCIs and other inter-DC connectivity options are available, but it's unclear how they will be supported and work across R&Es.
- One of the major challenges that remains is to understand how we can effectively propose and run cross domain projects involving sites, experiments and R&Es (there are existing projects that have faced similar challenges such as EU OCRE, HNSciCloud; but cross continental projects still remain a major challenge).

Proposed Areas of Future Work

A primary goal of this document is to seed a collaboration between the experiments, the sites and the research and education networks to deliver capabilities needed by HEP for their future infrastructure while enabling the sites and NRENs to most effectively support HEP with the resources they have.

In this section we will outline possible areas of future work that can help tie together activities within and among the experiments and sites with network engineers, NRENs and researchers. It is critical that we identify projects that are useful to the experiments, deployable by sites, and that involve a range of participants spanning the sites, the experiments and the (N)RENs. Without the involvement of each, we risk creating something unusable, irrelevant or incompatible.

This document completes the HEPiX NFV/SDN working group Phase I effort, which targeted surveying the various technologies and projects relevant in this area for WLCG and suggesting an outline of a possible Phase II.   The Phase I results were presented at the Fall 2019 HEPiX in mid-October and this report will be broadly distributed thereafter for comment. We will use the next LHCOPN/LHCONE meeting at CERN in mid-January 2020 as a decision point for a Phase II program of work.

Below is a list of areas we have discussed that we believe are relevant to at least some of our stakeholders for a second phase of this working group:
- Data lakes, federated sites, cross site deployment technologies, specifically projects focusing on exploring inter-DC connectivity technologies such as DCI (software or hardware) exploring the possibilities for automated cross-site deployment and operations as well as the possibility to co-locate caches (such as XCache) at the network hubs
- Container-based edge services (such as SLATE), container native and/or federated prototypes (Kubernetes only sites), HTCondor Kubernetes backend prototype testing. In particular, projects focusing on deployment strategies of the container-based system in HEP and adoption of previously listed industry standards and approaches.
- Federated sites, cross site deployment of network virtualisation and cloud-native networking. Two potential areas would be orchestration of a separated VM region running on cloud-native and inter-connecting to other sites via DCIs, the other would be federated storage using DCIs and cloud-native.
- Storage/GPU virtualisation for container/VM based sites - integration of block storage and GPUs in the data centres using NVMe-oF (NVMe over Fabrics) or similar technologies DC Edge hyper-converged infrastructures and edge service architectures for the HLT/readout systems
- SD/WAN integrations at the DC edge (multi-ONE and other SD/WAN approaches) - transfer broker prototypes (that would include anticipated and existing transfers across experiments; or existing transfers per domain/experiment; also tie to WLCG DOMA). Additional projects in this area might involve programmatic access to the network topologies (LHCONE), routing tables and their visualisation.

- SDX use cases in network provisioning, domain failure limitation, automated/intelligent brokering of the experiments traffic in the background, APIs (for multi-ONE/LHCONE ?)
- Clouds - extending DC site networking to Clouds (via VPC) and exploring transit gateways and similar approaches [AWS]
- GEANT connect, ESNet6 telemetry/higher level services, FABRIC and GEANT Testbed Service (GTS) as testbeds for data management R&D. Packet tracing mechanisms across different R&Es, monitoring and network telemetry APIs.

However, given the diversity and length of this list, individual items are either too generic, too specific or not sufficiently relevant to all stakeholders to create a broad collaboration around.

To make progress in the near-term, we would like to suggest a few specific areas where the sites, experiments and NRENs might profitably collaborate. Below are three examples we believe deserve discussion to see if we can identify interest in near-term collaboration with targeted deliverables. *These are not meant to be exclusive, merely suggestions based upon the working groups interactions and discussions amongst its members.*

1. **Making our network use visible** - Understanding the HEP traffic flows in detail is critical for understanding how our complex systems are actually using the network.   The potential work here is to identify how we might label our traffic to indicate which experiment and task it is a part of. This is especially important for sites which support many experiments simultaneously where any worker node or storage system may quickly change between different users. With a standardized way of marking traffic, any NREN or end-site could quickly provide detailed visibility into HEP traffic to and from their site. The technical work would encompass how to mark traffic at the network level, defining a standard set of markings and providing the tools to the experiments to make it easy for them to participate. We note that the increasing use of VMs and containers might make marking traffic easier where those technologies are in use.

2. **Shaping data flows** - It remains a challenge for HEP storage endpoints to utilize the network efficiently and fully.   An area of potential interest to the experiments is traffic shaping (see packet pacing discussion above in the Programmable WAN Motivation section).   WIth traffic shaping, network packets are emitted by the network interface in bursts corresponding to the wire speed of the interface.   Packets sent from a 10 Gbps network interface can create a micro-burst which can overflow buffers along the path or at the destination, causing packet loss. The impact on TCP, especially for high-bandwidth transfers on long network paths can be significant.   If instead, flows are shaped to better match the end-to-end usable throughput, the result is much smoother flows at higher bandwidth. A significant extra benefit is that these smooth flows are much friendlier to other users of the network by not bursting and causing buffer overflows.

3. **Network orchestration to enable multi-site infrastructures** - Within our data centers, technologies like OpenStack and Kubernetes are being leveraged to create very dynamic infrastructures to meet a range of needs.   Critical for these technologies is a level of automation for the required networking using both software defined networking and network function virtualization.   As we look toward high luminosity LHC, the experiments are trying to find tools, technologies and improved workflows that may help bridge the anticipated gap between the resources we can afford and what will actually be required to extract new physics from massive data we expect to produce. The ways in which we may organize our computing and storage resources will need to evolve.   Architectures like Data Lakes,

federated or distributed Kubernetes and multi-site resource orchestration will certainly benefit (or require) some level of WAN network orchestration to be effective.   To support this type of resource organization evolution, we need to begin to prototype and understand what services and interactions are required from the network. We would suggest a sequence of limited scope proof-of-principle activities in this area would be beneficial for all our stakeholders.

Conclusions

The primary challenge we face is ensuring that WLCG and its constituent collaborations will have the networking capabilities required to most effectively exploit LHC data for the lifetime of the LHC. To deliver on this challenge, automation is a must. The dynamism and agility of our evolving applications, tools, middleware and infrastructure require automation of at least part of our networks, which is a significant challenge in itself. While there are many technology choices that need discussion and exploration, **the most important thing is ensuring the experiments and sites collaborate with the RENs, network engineers and researchers to develop, prototype and implement a useful, agile network infrastructure that is well integrated with the computing and storage frameworks being evolved by the experiments as well as the technology choices being implemented at the sites and RENs.**

Terminology

Switch/bridge - is networking hardware that connects devices on a computer network by using packet switching to receive, and forward data to the destination device. In the context of this report switch/bridge are used interchangeably and refer to a multiport network bridge that uses media access control addresses to forward data at the data link layer (*layer 2*) of the OSI model.

Router - a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. When multiple routers are used in interconnected networks, the routers can exchange information about destination addresses using a *routing protocol.* In the context of this report we usually refer to routers/routing at layer 3, where packets are sent to a specific next-hop IP address, based on destination IP address.

VPN - Virtual Private Network - extends a private network across a public network, and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network.

VLAN - Virtual Local Area Network, provides layer 2 abstraction, in traditional bridging network it's used together with Spanning Tree Protocol (STP), which constructs a single tree across a bridged network to forward all packets. With VLANs, there is usually a separate tree per VLAN.

VRF - Virtual Routing and Forwarding is a layer 3 abstraction, which provides a separate routing table for each virtual private network (VPN), usually this is done by adding some sort of VRFID to the routing table lookup.

VXLAN - Virtual eXtensible Local Area Network (VXLAN) - a typical example of an overlay model, which is used to build a traffic isolation at layer 2 across layer 3 infrastructure. Usually it's coupled with VRF to provide a full bridging and routing overlay solution.

NVGRE - Network Virtualisation over GRE (GRE) protocol [RFC7673]

STT - Stateless Transport Tunneling

DWDM - Dense Wavelength Division Multiplexing is an optical technology used to increase bandwidth over existing fiber optic backbones.

GENEVE - Generic Network Virtualisation Encapsulation (GENEVE)

GRE - Generic Routing Encapsulation protocol

FRR - Free Range Routing (FRR) is an open source framework implementing number of common routing protocols (OSPF, IS-IS, BGP) and deployed as part of open source network operating systems

SDN - Software Defined Networking - technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management.

AS - Autonomous System - is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators on behalf of a single administrative entity or domain that presents a common, clearly defined routing policy to the internet.

BGP - Border Gateway Protocol - is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet.

eBGP - exterior Border Gateway Protocol - protocol used to transport information to other BGP enabled systems in *different* autonomous systems (AS)

iBGP - interior Border Gateway Protocol - protocol used between the routers in the *same* autonomous system (AS).

OSPF - Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS).

IS-IS - Intermediate System to Intermediate System (IS-IS, also written ISIS) is a routing protocol designed to move information efficiently within a computer network, a group of physically connected computers or similar devices. It accomplishes this by determining the best route for data through a Packet switching network.

MPLS - Multiprotocol Label Switching (MPLS) is routing and forwarding protocol; it operates between layer 2 and layer 3, so it's often referred to as layer 2.5 protocol. It assigns labels to packets and uses the labels to decide on subsequent routing and forwarding. It has been around since 90-ties and one of it's core use cases was to establish and manage layer-3 VPNs (L3VPN).

Acknowledgments

Appendix

StarLight SDX

The NSF IRNC-funded StarLight International/National Software Defined Networking Exchange (SDX) focuses on research, development, and deployment of services, architectures, and technologies designed to provide scientists, engineers, and educators with highly advanced, diverse, reliable, persistent, and secure networking services, enabling them to optimally access resources in North America, South America, Asia, South Asia, Australia, New Zealand, Europe, Africa, and other sites around the world. The StarLight SDX ensures continued innovation and development of advanced networking services and technologies. The goal of the StarLight SDX initiative is to expand upon existing and emerging experimental and prototype SDN/SDX/SDI networking architectures and technologies through the development of advanced international communication services for data-intensive science, in part, by leveraging existing designs and developments as well as by collaborating with science-domain research communities. This NSF IRNC SDX initiative is creating and implementing innovative methods for transporting large capacity data over long distance WANs, including many thousands of miles.

The StarLight SDX initiative specifically focuses on three main activities: (1) designing and implementing an SDN Development Environment – the StarAX (StarLight Advanced eXchange) Innovation Engine – that enables facilities around the world to deploy SDXs that are compatible and interoperable with others; (2) deploying instances of this system at the StarLight facility; and, (3) coordinating and collaborating with others around the world to deploy such systems at their exchange points. This project is developing an SDN Innovation Platform (software closely integrated with select hardware) to provide multiple network functions and features quickly and reliably.

The StarLight SDX initiative is creating methods for programmable networking using SDN and related virtualization techniques to enable higher levels of abstraction for network control and management functions at all layers and across all underlying technologies. Virtualization-based architecture leads to new capabilities for network services and infrastructure, allowing for a wide range of services and capabilities and options to closely integrate networks with other resources, such as compute resources, storage, instruments and sensors (e.g., through SDI architecture).

These new fabrics are comprised of multiple components, implemented within a emerging flexible architecture. The architecture is based on the concept of distributed Science DMZs. Components include high level APIs, protocols, data commons, compute systems, file systems, storage systems, and specialized network systems and services. Key enablers of these emerging services and capabilities are SDXs, which leverage the programmability of virtualized network resources made possible by SDN techniques. SDN has been widely implemented in large scale data centers and on private WANs interconnecting multiple data centers. The increased deployment of SDNs has motivated the development of creating SDXs, not only to extend SDN capabilities across multiple domains but also to provide exchanges with many more capabilities, especially for data intensive science. Multiple demonstrations at the SC International Conferences for High Performance Computing, Networking, Storage and Analysis have showcased a wide range of

capabilities of the StarLight SDX, including capabilities for dynamic provisioning, SD-WAN provisioning, dynamically implementing WAN L2 paths, and supporting large scale file transfers and high capacity data streams through Data Transfer Nodes (DTNs).

SDXs have a significant potential for highly granulated policy implementation; enabling slice exchanges that include more than network resources, migrating away from restrictive protocols such as BGP to provide a richer set of peering options; reduce routing complexity and allow more path flexibility, enabling highly distributed and granulated domains; enhancing capabilities for specialized access to resource descriptions, brokerages, and clearinghouses; providing enhanced flowvisor capabilities;
including: (1) managing large-scale network capacity, such as end-to-end 100 Gbps flows over WANs, as high-volume individual data streams; (2) highly granulated programmability for network services and resources, including ultra-high-capacity flows; (3) direct edge access to and control over these capabilities, e.g., by individual processes and applications; (4) direct management of individual workflow data streams; (5) individual stream attribute management; (6) topology exchange services; (7) controller federation, including support for East-West control channels across multiple domains (a major challenge today for SDN/OpenFlow networks); and demonstrating and showcasing these services and capabilities at major national and international conferences and workshops.

SDN technology not only specifies a separate control plane and data plane, it creates non-traditional data-plane traffic. For example, a national OpenFlow backbone project coined the term OpenFlowvlan (OFvlan), which means an OpenFlow switch can add a VLAN tag to the traffic header and send it to the next hop's network equipment, and it might not carry features that traditionally tagged VLAN frames support. To allow different types of SDN traffic to traverse data planes at different SDXs and traverse non-SDN networks without problems, this capability is a high-priority challenge.

This IRNC SDX initiative has established a project to make networks more application aware, for example, through the use of Jupyter, a scientific workflow manager that is being integrated with extensions for managing network services and resources. Today, science workflow management and network orchestration are accomplished with completely separate software stacks. The StarLight SDX has been used to investigate how science workflow management and network orchestration can be integrated using a combined software stack vs two separate software stacks. One project established a technique using Jupyter to integrate both within a common package that can eventually be integrated into an SDX service. This project is demonstrating the utility of using Jupyter for integrating scientific workflows with network orchestration techniques required for data intensive science.

Optimal management of large scale data intensive science workflows is often dependent on the orchestration of infrastructure resources, including network paths and attributes. This project leverages the utility of using Jupyter for integrating science workflow management and programmable network
orchestration, using SDN techniques. A major advantage of this approach is that it allows scientists to avoid learning the complexities of network provisioning and orchestration by providing a layer of abstraction above network orchestration processes. By using a well known scientific workflow

management tool, Jupyter, integrated with network orchestration processes, data intensive science can more easily provision and configure network resources without a deep understanding of network management and control processes.

Another project is investigating the potential for integrating Jupyter high level processes with an underlying set of network programming processes, including P4 (Programming Protocol-Independent Packet Processors). P4 is an emerging language networking programming language, a domain specific language for network protocols. This SDX project is exploring how P4 can be used to support large scale data intensive science workflows on high capacity high performance WANs and LANs, by introducing into the network a high degree of differentiation not previously possible. It has long been recognized that managing large scale data intensive science workflows on high performance LANs and WANs requires special programmable networking techniques, because most networks are designs for millions of small
flows. The StarLight SDX supports an international P4 testbed for computer science research in partnership with the GENI initiative.

This IRNC SDX initiative has established a collaborative partnership that is exploring specialized software stacks useful for creating a multi-institution, hyperconverged science DMZ, using Kubernetes as a core orchestrating resource. This consortium has been demonstrating the utility of the services provided by a prototype model to support data intensive science. The JupyterHub Kubernetes Spawner enables JupyterHub to spawn single-user notebook servers on a Kubernetes cluster. Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. Kubernetes enables the deployment of a JupyterHub setup that can scale across multiple nodes, for example, supporting over 50 simultaneous users. It scale to more nodes and reduces to less easily. JupyterHub can be run from inside Kubernetes. Consequently, many JupyterHub deployments can be run with Kubernetes only, eliminating a need for using scripts such as Ansible, Puppet and Bash. It has utilities for integrated monitoring and failover for the hub process. It allows for spawning multiple hubs in the same kubernetes cluster, with support for namespaces. Kubernetes also has comprehensive resource and security parameter controls.

Other components being developed to support these services are Data Transfer Nodes (DTNs). The PetaTrans 100 Gbps Data Transfer Node (DTN) research project is directed at improving large scale WAN services for high performance, long duration, large capacity single data flows. iCAIR is designing, developing, and experimenting with multiple designs and configurations for 100 Gbps Data Transfer Nodes (DTNs) over 100 Gbps Wide Area Networks (WANs), especially trans-oceanic WANs, PetaTrans
– high performance transport for petascale science, including demonstrations at major conferences and workshops. The PetaTrans 100 Gbps Data Transfer Node (DTN) research project is directed at improving large scale WAN services for high performance, long duration, large capacity single data flows. iCAIR is designing, developing, and experimenting with multiple designs and configurations for 100 Gbps Data Transfer Nodes (DTNs) over 100 Gbps Wide Area Networks (WANs), especially trans-oceanic WANs, PetaTrans – high performance transport for petascale science, including demonstrations at major conferences and workshops. These DTNs are being designed specifically to optimize capabilities for supporting E2E (e.g., edge servers with 100 Gbps

NICs) large scale, high capacity, high performance, reliable, high quality, sustained individual data streams for science research.

To provide multiple domain capabilities, this IRNC SDX consortium is developing and demonstrating services, architecture, and technologies for SDXs that can directly control dynamic 100 Gbps paths. These SDX capabilities are also based on advanced switching capabilities , including through experiments. Several of the SC19 IRNC SDX planned demonstrations include a fundamentally new concept of —consistent network operations, where stable load balanced high throughput workflows crossing optimally chosen network paths, up to preset high water marks to accommodate other traffic, are provided by autonomous site-resident services, dynamically interacting with network-resident services, in response to requests from science programs principal data distribution and management systems. This is empowered by end-to-end SDN control at Layer 2 (switching) and Layer 1 (optical) extending to autoconfigured DTNs, combined with advanced transfer applications such as mdtmFTP and Big Data Express.

The StarLight consortium and its research partners have also been working on a DTN-as-a-Service project so that these capabilities can be integrated with SDXs. The DTN-as-a-Service Software Stack provides users with an advanced science workflow environment for high-speed network data transfer using DTNs. The DTN-as- a-Service Software Stack consists of multiple modules implement core functions and the modules are controlled by a science workflow controller which allows users to orchestrate high-speed network data transfer with predefined parameters. The system configuration module is responsible for managing storage devices for the data transfer. Users can define a transfer setup to configure RAIDs or individual drives, choose a file system and choose the desired files or create test files for the transfer. The system optimization module provides autonomous system tuning for DTNs. It checks system parameters for a high-speed network data transfer and informs users on the current settings. Users can optimize DTNs autonomously using for the high-speed network data transfer. Transfer tools are managed by the transfer tools module, which executes high-speed network data transfer based on the defined transfer setup. The module manages predefined transfer tools and executes with the transfer setup which includes a number of parallel flows and storage devices to use. The network provisioning module orchestrates network paths between the DTNs using OpenNSA, this module also provides interfaces for other types of network provision.The monitoring module allows users to watch transfer activities in real-time and evaluate each transfer executed. Users can generate graphs from each transfer and evaluate them using the monitoring module. It also allows users to observe the activity in Grafana dashboards. Below is an overview of the software stack.
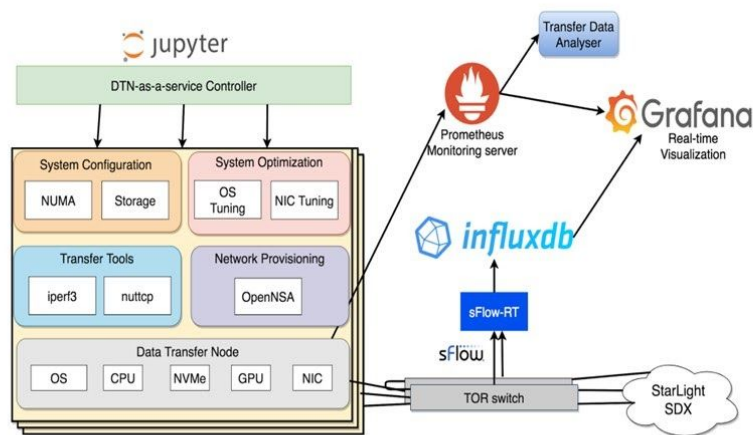
This DTN service provides a) a network and system test point (software stack, architecture, connectivity and performance, including over WANs) for large data transfer projects b) a platform for large flow projects c) additional capacity for large flow projects d) access to experimental scalable DTN technology, including over WANs.

In preparation for SC19, a project has been initiated to evaluate the standards, technologies, and switches that are emerging for support for 400 Gbps Ethernet NICs. Plans are being developed to determine how to showcase a) 400 Gbps LAN capabilities at SC19, including in the StarLight booth, b) 400 WAN capabilities and c) the interface between the 400 Gbps WAN and the 400 Gbps LAN.

AtlanticWave-SDX

The AtlanticWave-SDX (AW-SDX) is being built as a distributed, multi-domain, wide-area SDX platform that controls many network switches across the U.S. and South America. Its primary user audience is network administrators and domain scientists. Because of its distributed nature, the AW-SDX architecture is split into multiple layers: (1) Users, Orchestrators and Applications; (2) the AW-SDX Controller; and (3) the Local Controllers (LC). Figure below provides a representation of the current AW-SDX architecture.
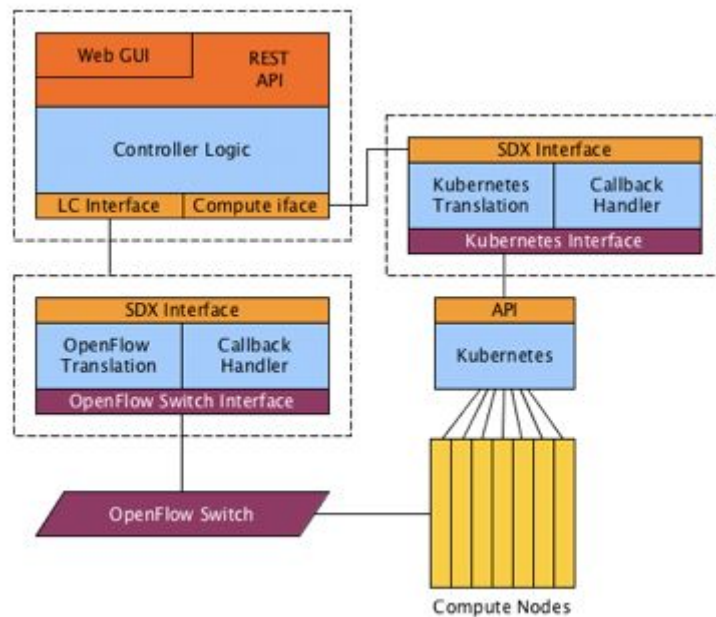
The SDX Controller provides northbound interfaces for external requests from orchestrators, users and applications. Orchestrators that are currently supported are SENSE [14] and NSI [15]. Users consist of domain scientists and network operators. They are supported through different web interfaces with meaningful input templates for each external request, as well as through a REST API. Applications, such as scientific workflow management systems, that consume end-to-end services composed by an SDX controller, are also supported through a REST API. Authentication and Authorization of external requests are the responsibility of the AW-SDX Controller. The AW-SDX Controller breaks down policies from the northbound interfaces to rules for the Local Controllers. A Local Controller-to-SDX API (LC/SDX API) enables the AW-SDX Controller to maintain a full topology of the network.

Local Controllers (LC) operate at each local exchange point site. Each site as a LC and local switches that collectively define the Local Data Plane. LC translate rules into a switch's lower-level southbound interface. LC are responsible for relaying status information from a resource to the SDX Controller. Southbound interfaces currently supported are OpenFlow 1.3, and OpenFlow 1.3 with Corsa extensions. Southbound interfaces are designed to add new interfaces easily, such a P4, and compute and storage. AtlanticWave-SDX will be deploying SDX Controllers at the following SDXs: SoX in Atlanta, AMPATH in Miami, SouthernLight in Sao Paulo, Brazil, and AndesLight in Santiago, Chile.

Pacific Wave Expansion Supporting SDX & Experimentation

References:
1. MEF Publishes Industry's First SD-WAN Standard, https://newswire.telecomramblings.com/2019/08/mef-publishes-industrys-first-sd-wan-standard/
2. SD-WAN Service Attributes and Services, July 2019. https://www.mef.net/mef-3-0-sd-wan
3. What is Software-Defined WAN (or SD-WAN or SDWAN)? https://www.sdxcentral.com/networking/sd-wan/definitions/software-defined-sdn-wan/

4. Gupta, Arpit, et al. "Sdx: A software defined internet exchange." ACM SIGCOMM Computer Communication Review. Vol. 44. No. 4. ACM, 2014.

5. Feamster, N., Ricci, R., "Report of the NSF Workshop on Software Defined Infrastructures and Software Defined Exchanges." February 4-5, 2016.

6. Chung, Joaquín, et al. "Atlanticwave-sdx: An international sdx to support science data applications." Software Defined Networking (SDN) for Scientific Networking Workshop, SC'15. 2015.

7. J. Ibarra, J. Bezerra, H. Morgan, L. Fernandez Lopez, M. Stanton, I. Machado, E. Grizendi, D. Cox, Benefits brought by the use of OpenFlow/SDN on the AmLight intercontinental research and education network, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM, 2015, pp. 942–947, http://dx.doi.org/10.1109/INM.2015.7140415.

8. J. Chung, E.-S. Jung, R. Kettimuthu, N.S. Rao, I.T. Foster, R. Clark, H. Owen, Advance reservation access control using software-defined networking and tokens, Future Gener. Comput. Syst. 79 (Part 1) (2018) 225–234.

9. Software-Defined Networking: The New Norm for Networks, 2012. Technical Report, Open Networking Foundation (ONF) (April).

10. Karakus, Murat, and Arjan Durresi. "Quality of service (qos) in software defined networking (sdn): A survey." Journal of Network and Computer Applications 80 (2017): 200-218.

11. StarLight SDX: A Software Defined Networking Exchange for Global Science Research and Education. https://www.nsf.gov/awardsearch/showAward?AWD_ID=1450871&HistoricalAwards=false

12. AtlanticWave-Software Defined Exchange: A Distributed Intercontinental Experimental Software Defined Exchange (SDX). https://www.nsf.gov/awardsearch/showAward?AWD_ID=1451024&HistoricalAwards=false

13. Pacific Wave Expansion Supporting SDX & Experimentation. https://www.nsf.gov/awardsearch/showAward?AWD_ID=1451050&HistoricalAwards=false

14. I. Monga, C. Guok, J. MacAuley, A. Sim, H. Newman, J. Balcas, P. Demar, L. Winkler, T. Lehman, X. Yang, SDN for end-to-end networked science at the exascale (SENSE), in: The 5th Innovating the Network for Data-Intensive Science, INDIS Workshop, Dallas, TX, USA, 2018.

15. Open Grid Forum, "Network Service Interface." https://redmine.ogf.org/projects/nsi-wg.

16. SDN for End-to-end Networked Science at the Exascale (SENSE), http://sense.es.net