FABRIC Network Service Model

Paul Ruth, Ilya Baldin, Komal Thareja
RENCI - UNC Chapel Hill
Chapel Hill, NC, USA
{pruth,ibaldin,kthare10}@renci.org

Tom Lehman
Virnao
tlehman@virnao.com

Xi Yang, Ezra Kissel
Energy Sciences Network
Berkeley, CA, USA
{xiyang,kissel}@es.net

***Abstract*—The FABRIC research infrastructure enables cutting-edge experimental networking research at-scale. Central to the FABRIC philosophy is providing researchers access to everywhere-programmable infrastructure including compute, storage, and networking connected with dedicated optical links deployed across more than 30 geographically distributed sites.**

At a high-level, each FABRIC site can be understood as a small cloud providing access to advanced computational hardware. One challenge to developing a novel faculty, such as FABRIC, is modeling the set of advanced networking services that are available to the user. The network services required by FABRIC users have several factors that contribute to this challenge including the need to experiment with layer 2 and layer 3 protocols spanning wide- and local-area networks, as well as dedicated connections to external 'edge' facilities such as other testbeds (Chameleon Cloud, CloudLab, and the PAWR testbeds), supercomputing centers, campus infrastructure, and other large instruments.

This paper presents the FABRIC network services that enable users to experiment with complex network topologies. The work presented includes both the design and implementation of the control framework that instantiates network services on behalf of the user, as well as the efforts toward the FABlib library and JupyterHub environment that abstract the network services, simplifying the design and deployment of advanced networking topologies on FABRIC.

***Index Terms*—Computer networks, Wide area networks, Cloud computing, Application programming interfaces**

I. Introduction

FABRIC [1] provides users with the ability to deploy networking experiments at-scale targeting cybersecurity, distributed computing and storage systems, machine learning, and science applications. Each experiment is composed of one or more *slices* [4] containing resources acquired from FABRIC's geographically distributed infrastructure. Many of FABRIC's resources are similar to those available in traditional public and private clouds with the addition of deeply programmable networking hardware (P4, OpenFlow, Smart NICs, etc.). The major contribution of FABRIC is that its resources are located in the core of the wide-area network and can be connected using user-specified networking services with many possible configurations not available to researchesrs using the existing Internet.

FABRIC is supported in part by a Mid-Scale RI-1 NSF award under Grant No. 1935966.

ISBN 978-3-903176-48-5

One of the main FABRIC innovations is the design of the network service model that defines multiple types of on-demand network services that are available to experimenters. These services can be best effort or QoS-assured on-demand Layer 2 channels between desired interfaces in the topology, they can attach desired interfaces of the topology to shared IPv4 or IPv6 dataplanes that are shared among multiple experiments and are private to FABRIC, they can create on-demand peering with the public Internet, and can create on-demand port mirroring arrangements to enable the collection of measurements from inside the slice or about other slices. FABRIC also allows on-demand connections from slices to external partner facilities, referred as Facility Ports. Multiple services and service instances can be combined in a single slice, as desired by the experimenter, mixing L2 and L3 services to achieve desired results. This rich set of capabilities allows FABRIC experimenters to create unique experiment topologies that combine FABRIC resources, resources of FABRIC facility partners, testbeds, scientific instruments and other resources reachable via on-demand Layer 2 networks or via public Internet. Due in large part to these capabilities, we refer to FABRIC as the 'testbed-of-testbeds'.

II. Network Services

FABRIC has multiple options for Layer 2 Network Services. A single slice may include one or more FABRIC provided Layer 2 services between its VM slivers. Experiments cannot bridge together two or more Layer 2 services. That is, FABRIC Layer 2 services remain isolated from one another. When needed, interconnects between FABRIC Layer 2 Services can be accomplished using routing.

All FABRIC network services are implemented using Cisco NSO (Network Service Orchestrator) and exposed to the user via FABRIC Control Framework, which presents FABRIC Information Model Abstractions (FIM) to higher layers, like FABlib. The implementation relies on Cisco IOS MPLS-SR (Multi-Protocol Label Switching-Segment Routing) protocol stack. Unlike more traditional VLAN-based network service implementations, this means that very little state is maintained in the network to support each service, substantially cutting down on service setup time. Services do present a VLAN-like service to the end user, however this is only done on the last hop between the FABRIC Cisco router and the interface of a network card at one of the sites. VLAN translation is performed automatically by MPLS SR, thus freeing the user or

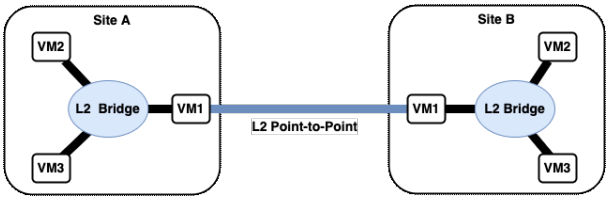

Fig. 1. FABRIC L2 Point-to-Point (L2PTP) network service connecting two L2 Bridge network: A pair of nodes are directly connected using a L2PTP connection and act as routers for local L2 Bridge network services on each site.

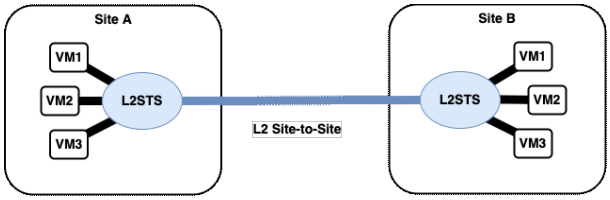

Fig. 2. FABRIC L2 Site-to-Site (L2STS) network service: Several nodes from two different FABRIC sites are connected with a L2STS network service. All nodes can communicate directly over the single L2 broadcast network.

the FABRIC Control Framework from the need to coordinate last hop VLAN assignments, simplifying the implementation.

In the following sections we describe the available FABRIC network services, starting from the simpler on-demand L2 services - Layer 2 Point-to-Point (L2PTP) and Layer 2 Site-to-Site (L2STS), Layer 2 Bridge (L2Bridge) followed by Layer 3 IPv4 and IPv6 services (FABNetv4, FABNetv6), Layer 3 VPN, on-demand peering, Port Mirroring and Facility Ports.

A. Layer 2 Services

The Layer 2 Point-to-Point Connection Service (L2PTP) is based on an industry standard "Virtual Private Wire Service (VPWS)". A service instance connects a *service termination point* on one dataplane network element to a *service termination point* on another dataplane network element There may be one or more intermediate or transit dataplane network elements whose function is to switch MPLS labels. A "service termination point" is defined by a port and an associated *VLAN Profile*. A VLAN Profile typically is defined by a single VLAN, but could also be a range of VLANs. This service is a completely transparent ethernet frame transport service. There is no MAC learning, MAC forwarding, or ethernet protocol processing. L2PTP can have associated QoS parameters (burst size and bitrate) as well as an ERO (Explicit Route Object) that allows experimenter to dictate the path through the network this service must take.

Figure 1 shows an example of how a slice may use the Layer 2 Point-to-Point Connection Service. In the figure, `VM1` and `VM2` are directly connected with a layer 2 circuit that spans a pair of geographically distributed sites. As depicted in the figure, L2PTP links are commonly used to connect a pair of nodes (i.e. VMs or dedicated networking hardware) that are responsible for processing and forwarding traffic on behalf of several other nodes hosted at that site.

Layer 2 Bridge (L2Bridge) service always serves a single site, unlike L2PTP. This service is a local layer 2 broadcast domain configured on a single network dataplane element. Unlike other services, the technology utilized to create this is based on VLANs. A single broadcast domain (a 'bridge') is created within a single network element connecting the desired interfaces together. Because the service is local to a single network element, dedicated network interfaces of VM slivers do not experience contention with other slices and can utilize the full line rate (25Gbps or 100Gbps, depending on the interface type). In Figure 1, each site has an L2Bridge that is used to connect a set of local VMs. One VM is also connect to an L2PTP link and has the responsibility to forward traffic on behalf of the other VMs.

The Layer 2 Site-to-Site Connection Service (L2STS) is based on an industry standard "Ethernet Virtual Private Network (EVPN)" technology. A service instance connects multiple service termination points on one dataplane network element to multiple service termination points on another dataplane network element. There may be one or more intermediate or transit dataplane network elements whose function is to switch MPLS labels. This service is not a transparent Ethernet frame transport service. This services includes MAC learning and MAC forwarding which is distributed using the network internal BGP messages. Unlike L2PTP, L2STS does not allow quality of service parameters or ERO to be specified. Figure 2 show an L2STS network service. All VMs are directly connected the to L2STS service and can communicate directly with all other nodes.

B. Layer 3 Services (FABNetv4 and FABNetv6)

The FABRIC Layer 3 Services provide a preconfigured routed service between desired interfaces of a slice topology. Two service types are offered - one based on IPv4 addresses from RFC1918 [5] private address space (FABNetv4), the other is based on IPv6 service based on FABRIC's own allocation of publicly routable IPv6 addresses (FABNetv6). Importantly there is just a single instance of each service present in FABRIC and sliver interfaces simply attach and detach from these instances based on experimenter requirements. At each site each slice is allocated a block of addresses (IPv4 or IPv6) by the Control Framework, that is used for interfaces belonging to this slice at that site. Routing between them is automatically accomplished by the FABRIC routing stack. Dynamic ACLs (Access Control Lists) can be introduced to prevent slices from communicating with one another. Both FABNetv4 and FABNetv6 services are best effort.

Figure 3 shows the FABNet service. Users are provided with an L2 network for each site that includes an IP subnet allocation and a FABNet router IP. The FABNet service is responsible for forwarding L2 traffic on behalf of the user.

C. Layer 3 Virtual Private Network (VPN) Service (L3VPN)

The Layer 3 Virtual Private Network (VPN) service shown in Figure 4 is based on an industry standard "Virtual Private

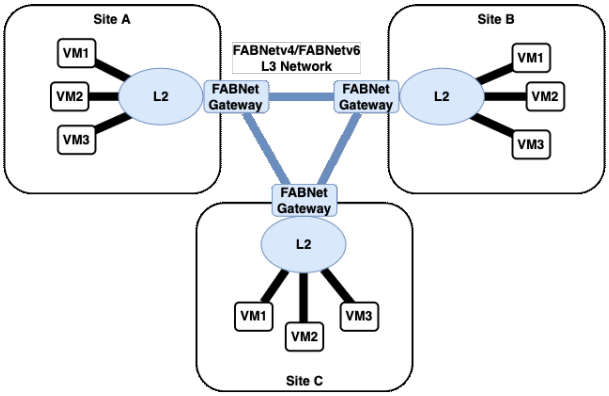

Fig. 3. FABRIC Layer 3 Routed Services (FABNetv4 and FABNetv6). Separate FABNet networks are created at each of three sites. The local FABNet networks are assigned a subnet and gateway. Nodes on each site are configured to use the assigned FABNet gateway to route traffic to any other FABNet network.

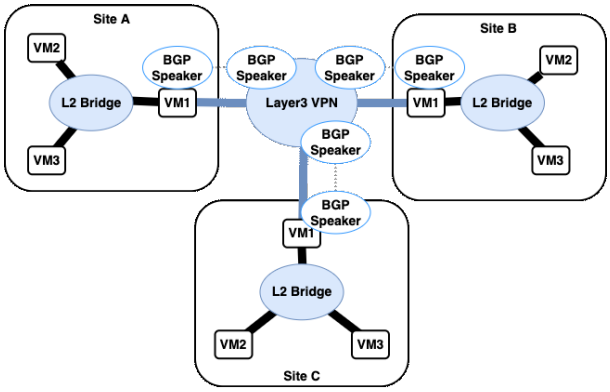

Fig. 4. FABRIC Layer 3 VPN network service connecting L2 Bridges at three sites. L3VPN network services can be used for wide area connections. Unlike L2PTP and FABNet network services, L3VPN requires the use of BGP.

Network (VPN)" technology. This VPN is provisioned over a standard IP routed service. The purpose of the service is to attach external entities to a slice using publicly routable network. This could include experimenter's laptop, a facility on campus or a scientific instrument.

A service instance connects one or more service termination points on a dataplane network element to a Layer 3 VPN. The slice must run a BGP Speaker, with private ASN, at each of the one or more service termination points and peer with the FABRIC Network L3VPN BGP Speaker. The L3VPN will only route *slice private* IPv4 or IPv6 address space. This address space must be provided by the slice owner. The service will provide routing services based on Slice specific private address space, between distributed slice resources (slivers) across the FABRIC Infrastructure. Generally this service is expected to be best-effort.

D. Port Mirror

Port Mirroring service allows experimenter to request that all traffic from a specific port/interface on the dataplane switch in a given FABRIC site is mirrored onto another port that belongs to the experimenter slice. This allows experimenters to collect traffic from their own or even other experimenters' slices. This service requires special authorization to be instantiated to protect the privacy of individual experiments and only a small number of experimenters are expected to be allowed to use it.

E. Facility Port

Facility Port service is a special instance of an L2PTP service that connects a designated interface in the experimenter's slice to a pre-defined interface outside the FABRIC footprint. These facility ports are predefined and pre-configured within FABRIC Information Model aggregate advertisements and the experimenters can choose which of these ports they want to connect to their topology. Special authorization is required to instantiate a Facility Port service to protect the security of the facilities.

In order to enable a Facility Port, FABRIC negotiates with facility's networking team to create a predefined Layer 2 peering point - an interface on their switch, which is connected to an interface on FABRIC dataplane switch at some site. Since this service is similar to L2PTP, an experimenter may choose to connect an interface of a VM in any site in FABRIC to this facility port, not just the site immediately adjacent the facility in FABRIC topology. These services can have quality-of-service parameters associated with them.

III. FABlib

The FABRIC network service model provides a powerful set of abstractions that enable wide array of Internet-scale networking experiments. Executing a networking experiment in a complex network topology is challenging. The FABlib library simplifies the development and deployment of complex networking experiments using the FABRIC network service model.

FABlib is a Python library included as part of the `fabrictestbed-extensions` PyPi package. FABlib is designed to simplify the use of the FABRIC API and it's network service model. Python applications can query and use FABRIC services through FABlib. These capabilities include creating, deleting, and modify slices supporting experiments, as well as querying for available FABRIC resources. The remainder of this section describes the primary FABlib abstractions for the initial FABRIC network services and provides examples of their use.

A. Slices

The *slice* is the primary abstraction used to describe an experiment topology on FABRIC. A slice is a collection of logically-related resources representing a single execution of an experiment or a portion of an experiment. The first step in deploying an experiment using FABlib is to create a slice to contain any resources needed for the experiment. Initially, a newly created slice is empty and not instantiated. The user must add nodes, components, and network services as described in Sections III-A0a and III-B. After the user has

```
1  #Create Slice
2  my_slice = fablib.new_slice(name='MySlice')
3  # Node1
4  node1 = my_slice.add_node(name='Node1', site='STAR', image='default_ubuntu_20',
5                            cores=2, ram=8, disk=100)
6  [iface1] = node1.add_component(model='NIC_Basic', name='nic1').get_interfaces()
7  # Node2
8  node2 = my_slice.add_node(name='Node2', site='STAR', image='default_ubuntu_20',
9                            cores=2, ram=8, disk=100)
10 [iface2a, iface2b] = node2.add_component(model='NIC_ConnectX_6', name='nic1').get_interfaces()
11 # Network(s)
12 net1=my_slice.add_l2network(name='net1', interfaces=[iface1, iface2a])
13 #Submit the request
14 my_slice.submit()
```

Listing 1: Add an L2 Bridge to a slice. The network requires a list of interfaces associated with previously created components. The type of network service created is determined by the locations of the interfaces in the list.

added all required resources to the new slice, the slice is instantiated by *submitting* the slice request to the FABRIC orchestrator which manages the deployment of the slice on behalf of the user.

Listing 1 shows how to deploy a simple FABRIC experiment. Line 4 shows how to create the slice and Line 14 shows submitting the slice to be instantiated.

After a slice is created, the application can add compute resources that include networking, compute, storage, and other specialized accelerator components. At the time this writing, virtual machines are the only compute nodes that are available. Bare metal nodes will, eventually, be available although space and power constraints will limit their availability.

a) Adding Nodes: Adding a new virtual machine to a slice is performed using the *add_node* method in Listing 1. A user is free to choose the name of the node, the virtual machine image, the FABRIC site, as well as the amount of basic resources capacities. Line 4 shows how the user can specify name, FABRIC site, and capacities (compute cores, memory,and local disk space) of each virtual machine.

The only required argument is the `name`. Unspecified capacities will be set to the minimum possible value (currently `cores=2`, `ram=8`, and `disk=10`). If the site is not specified, a random site will be chosen.

b) Adding Components: One of the primary features of FABRIC is enabling experimentation using modern computer hardware in the core the wide-area internets. Many of the advanced hardware components available in FABRIC are PCI devices that are directly attached to virtual machines using PCI passthrough. These devices include high-bandwidth SR-IOV NICs (up to 100 Gbps), Smart NICs, FPGAs, GPUs, and NVMe storage devices. Experiments are given full control of any PCI devices that are added to a slice as *components* of a virtual machine. The devices are accessible via the Linux PCI subsystem and the user can install any device driver and driver configuration required by the experiment.

Components are added to the a node using the *add_component* method in Listing 1 line 10. The example shows a Mellanox ConnectX-6 NIC with dual 100 Gbps ports. FABlib components that include network ports, such as the Mellanox ConnectX-6, have a list of *interfaces*, one for each port. A component's interfaces can be connected with appropriate network services to meet the requirements of the experiment. The example in Listing 1 shows how to access the first interface on each of two nodes, one with a `NIC_Basic` component and the other with a `NIC_ConnectX_6`. However, the `NIC_ConnectX_6` component has an additional unused interface that could be connected to an addition network, if needed.

Currently, other components include Mellanox ConnectX-5 NICs with dual 25 Gbps ports, Tesla T4 GPUs, RTX-6000 GPUs, and 1 GB NVMe drives. Additional component types will be added as FABRIC continues to be deployed. The planned Xilinx FPGA components will include network interfaces and will enable FPGA based networking experiments including hardware P4 support.

B. Network Services

Dedicated at-scale networks topologies are critical to experimenting with novel internet architectures. The network services described in this paper are used to connect the nodes and components that are part of the experiments designed by FABRIC users. FABlib provides a simple abstraction for incorporating these network services in a user's experiment.

FABRIC experiments are able to target network stack layers as low as layer 2. The FABlib abstraction supports deploying layer 2 and layer 3 network services.

1) L2 Network Services: FABRIC's L2 network services provide Ethernet connectivity between networking components of nodes in a slice. Although, there are several underlying FABRIC L2 network services (L2PTP, L2STS, and L2Bridge), from the experimenter's perspective, the differences between these services are the locations of the connected components and the, possible, attributes of the service, such as QoS bandwidth guarantees. The FABlib library attempts to streamline the complexity of the lower-level network service from the user by automatically choosing the appropriate service based on the location of the connected components and any optional service configuration. All FABlib L2 networks are created using the method call in Listing 1 line 12. The

```
#Create Slice
my_slice = fablib.new_slice(name='MySlice')
#Node1
node1 = my_slice.add_node(name='Node1', site='STAR', image='default_ubuntu_20',
                          cores=2, ram=8, disk=100)
[iface1] = node1.add_component(model='NIC_Basic', name='nic1').get_interfaces()
#Node2
node2 = my_slice.add_node(name='Node2', site='UTAH', image='default_ubuntu_20',
                          cores=2, ram=8, disk=100)
[iface2a, iface2b] = node2.add_component(model='NIC_ConnectX_6', name='nic1').get_interfaces()
#Network(s)
net1=my_slice.add_l2network(name='net1', interfaces=[iface1, iface2a])
#Submit the request
my_slice.submit()
```

Listing 2: Add an L2 STS/PTP to a slice. The network requires a list of interfaces associated with previously created components. The type of network service created is determined by the locations of the interfaces in the list.

type of network service created is determined by the locations of the interfaces in the list.

a) Local Ethernet (L2Bridge): The FABRIC L2 Bridge service is used when an experiment requires a local unmanaged Ethernet network switch. Listing 1 line 12 show the creation of a L2 Bridge service using the `slice.add_l2network()` method. This method will create a L2 Bridge network service when all interfaces included in the network are exclusively from a single FABRIC site. There is no limit on the number of interfaces that can be added to an L2 Bridge network service. The QoS of an L2 Bridge is limited by the components connected to the network. All dedicated PCI devices have their full bandwidth available (i.e. ConnectX-6 ports are 100 Gbps and ConnectX-5 ports are 25 Gbps). Basic NICs are implemented using 100 Gbps SR-IOV VFs but share bandwidth with the othe Basic NICs mapped to that physical port.

b) Wide-Area Links (L2PTP/L2STS): Wide-area layer 2 networking links are generally point-to-point links connecting a pair of geographically distributed switches or routers. FABRIC network services enable users to create wide-area *point-to-point* (L2PTP) links between a pair of component interfaces in their experiment topologies or wide-area *site-to-site* (L2STS) Ethernet networks. FABRIC *site-to-site* network services extend the capabilities of *point-to-point* links to allow any number of component interfaces participate the a single wide-area unmanaged Ethernet switch that, logically, spans two FABRIC sites. Although *Site-to-site* networks provide an easy way to connect many nodes across a pair of FABRIC sites, there are several practical limitations that prevent *site-to-site* networks from being used in many experiment topologies. Most notably, *Site-to-site* networks cannot have QoS guarantees and have connectivity restrictions when connecting Basic NICs (SR-IOV VFs) that are placed on a common host.

The FABlib library simplifies the deployment of L2P2P and L2STS network services by automatically choosing the appropriate wide-area L2 network service depending on the list of interfaces that are added to the network. L2PTP networks are created when there are be exactly two interfaces in the list and the interfaces are dedicated network devices (i.e. SR-IOV devices called `NIC_Basic` cannot be attached to an L2PTP network service). L2STS network services are created when the list includes more than two interfaces or any SR-IOV interfaces. An exception is raised if the list includes interfaces from more the two sites. Listing 2 shows how to use FABlib to create L2P2P and L2STS network services. The only difference between creating L2Bridge networks and wide-area networks is the location of the nodes that are added to the network.

2) L3 Networks (FABNetv4/FABNetv6): FABRIC provides a pair of layer 3 IP networking services across every FABRIC site (FABnetv4 and FABnetv6). These services enable a user's experiment to join FABRIC's private internet that connects all experiments across the testbed using FABRIC's high-performance network links.

The FABlib library provides functionality that enables a user's experiment to join FABnetv4 or FABnetv6. Listing 3 shows how to use FABlib to join FABnetv4. The user must create a local L3 network service at each site where the experiment requires L3 connectivity. All component interfaces added to an L3 network must be from the same site. The L3 network at each site will connect the interfaces with an isolated L2 network and will issue a subnet and gateway that is routable on the FABnet internet. Lines 11 and 12 show how to create the network and add interfaces. Lines 21-33 show how to configure the nodes with the assigned subnet and gateway. Routes must be added to the nodes to select which traffic to send across the FABNet network. Care must be taken to only add the desired routes to the FABnet gateway. Setting the default gateway to the FABnet network will cut off managment access to the node.

FABNetv6 will connect a user's experiment using IPv6 addressess from FABRIC's autonomous system. The FABnetv6 internet will be peered with the public IPv6 Internet. FABnetv4 uses RFC1918 addresses that will not peer with the public Internet. However, there are plans to extend FABnetv4 to include NAT access to the public internet and, possibly, allow a limited number of publicly routable addresses on some FABIRC sites.

```
my_slice = fablib.new_slice(name='MySlice')

node1 = my_slice.add_node(name='Node1', site='STAR', image='default_ubuntu_20',
                          cores=2, ram=8, disk=100)
iface1 = node1.add_component(model='NIC_Basic', name='nic1').get_interfaces()[0]

node2 = my_slice.add_node(name='Node2', site='UTAH', image='default_ubuntu_20',
                          cores=2, ram=8, disk=100)
iface2 = node2.add_component(model='NIC_Basic', name='nic1').get_interfaces()[0]

net1=my_slice.add_l3network(name='net1', interfaces=[iface1], type='IPv4')
net2=my_slice.add_l3network(name='net2', interfaces=[iface2], type='IPv4')

my_slice.submit().

# Wait for slice to become ready ...

net1 = my_slice.get_network(name='net1')
net2 = my_slice.get_network(name='net2')

net1_available_ips = net1.get_available_ips()
net2_available_ips = net2.get_available_ips()

node1 = my_slice.get_node(name='Node1')
iface1 = node1.get_interface(network_name='net1')
iface1.ip_addr_add(addr=net1.get_available_ips().pop(0), subnet=net1.get_subnet())

node2 = my_slice.get_node(name='Node2')
iface2 = node2.get_interface(network_name='net2')
iface2.ip_addr_add(addr=net2.get_available_ips().pop(0), subnet=net2.get_subnet())

node1.ip_route_add(subnet=net2.get_subnet(), gateway=net1.get_gateway())
node2.ip_route_add(subnet=net1.get_subnet(), gateway=net2.get_gateway())
```

Listing 3: Add an L3 network to a slice. The network requires a list of interfaces associated with previously created components and the type of L3 network.

C. Examples

Jupyter [3] notebooks have become a powerful and prolific way to learn about and interact with many online tools including cloud services. The FABlib library was designed to work well within Jupyter notebooks. Together, FABlib and Jupyter notebooks are the recommended way to use FABRIC. FABRIC provides a JupyterHub service that creates a private Jupyter environment for each user to develop and deploy experiments. Each user's Jupyter environment is seeded with a set of example notebooks. These notebooks are the best way to get started with FABRIC and to share advanced FABRIC techniques.

The examples in this paper are included in FABRIC's default Jupyter environment. In addition, the portability of Jupyter notebooks allowed FABRIC to be used in other Jupyter environments. The FABRIC examples are available in the Chameleon Cloud [2] Jupyter environment and can be loaded by invoking a Chameleon artifact [6].

IV. Conclusion

The FABRIC network service model is a powerful way for uses to design an deploy experiment with typologies using complex network services. FABRIC users can access the network services using the FABlib library on many available Jupyter environments.

References

[1] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S. Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing*, 23(6):38–47, 2019.

[2] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. Lessons Learned from the Chameleon Testbed. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 219–233, 2020.

[3] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[4] Rick McGeer, Mark Berman, Chip Elliott, and Robert Ricci. *The GENI Book*. Springer, 2016.

[5] Robert Moskowitz, Daniel Karrenberg, Yakov Rekhter, Eliot Lear, and Geert Jan de Groot. Address Allocation for Private Internets. RFC 1918, February 1996.

[6] Paul Ruth. Fabric testbed: Jupyterhub experiment examples, https://doi.org/10.5281/zenodo.6434975, April 2022.