Application-Aware End-to-End Virtualization Using a Named-Object Based Network Architecture*

Francesco Bronzino
LISTIC, Université Savoie Mont Mont Blanc
fbronzino@univ-smb.fr

Sumit Maheshwari, Ivan Seskar, Dipankar Raychaudhuri
WINLAB, Rutgers University
{sumitm, seskar, ray}@winlab.rutgers.edu

***Abstract*—Network virtualization enables applications and users to access network resources in isolation on top of a shared infrastructure. Through their mechanisms, virtual networks support performance and security guarantees that would otherwise not be achievable if relying solely on existing network protocols. Unfortunately, due to the large variety of network protocols and architectures that populate today's Internet, existing virtualization techniques have become disparate and different depending on the segment of the network they are deployed on. Thus, as no consolidation protocol exists, no solution is available to modern services and applications to coordinate the resources they employ to function. Due to the inefficiencies associated with overlay implementations they have to revert to, it becomes challenging for these services to meet strict application requirements, *e.g.*, low latency.**

In this paper, we revisit years of identity based communications to propose an approach to integrate network virtualization techniques around a single framework: the named-object abstraction. The presented approach uses unique virtual network identifiers to describe and implement custom topologies and routing metrics and achieve desired QoS requirements. This technique enables the support of an integrated network virtualization with end-to-end application aware routing on top of the network infrastructure. A proof-of-concept implementation running on the ORBIT testbed confirms that the named-object architecture can achieve low VN processing and control overhead. The proposed solution achieves an average latency performance improvement of 60% in comparison to a baseline implementation without compute and network cross layer optimizations.

***Index Terms*—Virtual networks, Named-object based networking**

I. Introduction

The continuous evolution and improvement of the internet's network infrastructure has over the years stimulated the growth of diverse applications and services. Thanks to increasing expectations, these applications nowadays demand stringent performance requirements of different kinds. For example, emerging applications such as augmented and virtual reality, as well as remote control of autonomous systems, require extreme low response time to function, in the order of a few milliseconds or less, posing orchestration challenges for the infrastructure resources on top of which they are deployed. As a possible solution to maximize performance, these applications often rely on compute and storage components distributed at different locations of the network infrastructure, *e.g.*, edge clouds. Network level solutions, *e.g.*, Extensible Internet components [1], seek to enable more built-in flexibility and control in the deployment of services on top of the shared network infrastructure and simplify the deployment of distributed applications and services.

Deploying applications and services on the networking infrastructure requires managing a distributed collection of resources spanning across different networking domains, technologies, and protocols. Meeting the desired requirements depends not only on these components themselves, but also on their interaction with the network. Ultimately, these services can only perform as well as the networks they traverse. Network virtualization is a common technique used to implement resource isolation over a shared infrastructure, implementing performance and privacy guarantees for the applications that are deployed on top. For example, virtualization techniques can enable resource slicing and seamless service integration.

Ideally, the optimal solution for applications would be to take advantage of virtualization technologies to deploy a private network with isolation and performance guarantees across all of the used infrastructure. Unfortunately, no single virtualization technology can work across the entire end-to-end path. The realization of an integrated end-to-end virtualization solution to sustain applications requirements would demand a concerted effort in monitoring as well as resources provisioning based on the application demand across domains and technologies. However, due to the lack of dedicated coordination mechanisms, existing virtualization techniques are unable to provide this level of support and connect the disparate compute, storage, and networking resources.

Two key factors limit the ability to deploy a holistic virtual network: First, the absence of shared mechanisms for inter-network communication makes very challenging for service providers to obtain information about applications' requirements, unless a direct deal is defined between the application provider and all network providers traversed. This can induce to inefficiencies, especially in those scenarios where mechanisms to support advanced packet delivery are supported (*e.g.*, 5G celllular networks). Second, the inherent absence of resource discovery mechanisms across networks makes it impossible for distributed applications with deployments spanning across multiple domains to take advantage of advanced delivery techniques (*e.g.*, load balancing across points of presence), with the exception of basic overlay solution. Unfortunately, overlays can result in suboptimal performance

*Invited paper

ISBN 978-3-903176-43-0

due to the lack of visibility of hosting infrastructure.

In this paper we argue that a more concerted effort to orchestrate the entire end-to-end path hosting modern applications is, indeed, possible. Ultimately, as presented by Balakrishnan *et al.* [1], the network compute infrastructure has become sufficiently mature to host application specific processing on high-capacity networks. But limiting solutions to overlays might limit the impact of such efforts. This is especially true in contexts where advanced delivery modes can be exploited to optimize application performance. For example, 5G architectures support by design advanced transmission mechanisms that can greatly impact application performance, but a lack of visibility into the applications requirements might limit their applicability. Towards better integrating application oriented network processing across all layers of the network architecture, we argue that any solution should achieve two fundamental goals:

1) **Offer a unified identifier for networks components to identify applications.** Unique identifiers (or tags) are commonly used to identify flows in virtual network architectures. Such approach could be extended across network domains and technologies to offer a unified virtualization language. Indeed, such integration should not break compatibility with legacy components, maintaining the ability for applications to work on top of the Internet's best-effort approach when enhanced support is not available.
2) **Offer the ability for applications to specify their intent and requirements.** Applications should be capable of expressing rules and requirements to the network infrastructure, enabling virtualization technologies to implement advanced delivery services without relying on ad-hoc deals between providers. Obviously, no unintended information regarding the application should be exposed to uninterested third parties.

Towards the goal of implementing an integrated virtualization architecture, we present in this paper a solution based on named-objects, a network abstraction that exploits name-address separation to implement advanced network services [2]. Name-address separation has inherent benefits in handling mobility and dynamism that when augmented with the application state information can be extended to achieve considerable flexibility in creating a variety of new service abstractions, thus supporting dynamic resource assignment and slicing, supporting QoS, and enabling heterogeneous virtual functions. We extend the named-object abstraction to support an application-aware end-to-end virtualization that offers the logical simplicity of L2 virtualization, and flexibility of the L3 one, with an ability to view the application state at the network to provide custom resource allocation, dynamic topologies, and advanced routing of data capabilities. Our proposed architecture lays its foundation on three virtualization technologies: (1) the Virtual Base Station (vBTS) to slice wireless access resources across multiple logical entities; (2) the Named-Object based Virtual Network (NOVN) for L3 inter-domain connectivity; finally,(3) a cross-layer optimized routing mechanism called Application Specific Routing (ASR).

The rest of the paper presents our previous work on enabling virtualization through named-based abstractions across multiple layers of the network architecture and how we combine it to implement and end-to-end virutalization architecture. We first summarize existing virtual network techniques and introduce the named-object abstraction in Section II. Section III describes how we virtualize the wireless access with vBTS. Section IV details the L3 network virtualization and Section V-A describes how NOVN is extended to support ASR routing. Section V-B covers how the components are integrated into end-to-end virtualization architecture. Finally, the evaluation and results are presented in the Section VI, and Section VII concludes the paper with a note on future work.

II. Background

In this section, we categorize network virtualization technologies based on the layer they belong to §II-A. We then illustrate the named-object abstraction and introduce how it can support end-to-end application-aware virtualization §II-B.

A. Virtual Network Technologies

Virtual networks research has been carried out for more than three decades. Table I summarizes the key features and limitations of some popular virtualization technologies from both industry as well as academia. We group network virtualization techniques depending on whether they are naturally supported within a layer of the protocol stack or whether they are implemented as overlays.

Layers 2 and 3. Native virtualization within the network stack normally relies on the use of unique identifiers, or tags, to identify virtual network membership. In particular, tag-based approaches place unique identifiers at different layers of the network stack to uniquely identify packet flows. Example of this are VLANs [3], [10], and MPLS [11]. Cloud networks have been one of the main adopters of Layer 2 virtual networks, with VN techniques being used to abstract the distribution of physical and logical resources within data centers (*e.g.*, applications, databases, and more), allowing for flexible management techniques. The core limitation of these solutions is their limited scope of applicability: employed tags are limited in size and have validity only within a single network. For this reason they can solely be used to support single domain solutions.

Layer 7. Layer 7 overlay based VNs, *e.g.*, VINI [12] or point to point connectivity between remote cloud locations [13], represent a flexible way for deploying experimental networks and protocols on top of the existing infrastructure. Through encapsulation of network packets on top of UDP packets and tunneling across participating nodes, they allow for the quickest solution to implement experimental protocols on top of the existing infrastructure. With this solution, flexibility and simplicity come at the cost of additional overhead. Moreover, residing at the application layer they lack the visibility of underlying network layer performance parameters, limiting

TABLE I
VIRTUAL NETWORK COMPARISON

VN Tech.	Objective	Technique	Operation	Scalability	Resource	Advantage	Areas of Improvement
VLAN -1989[3]	Switch-independent Host grouping	Frames bear VLAN ID in MAC header	L2	Single broadcast domain	Static (port-based) /dynamic (protocol -based) conf.	Simple, reconfigurable	Small domain, overhead and speed; maximum 4094 VLANs
VPN-1996[4]	Secure connection in public n/w	Sata encryption & secure forwarding	L2/L3	Multi-domain support	L2TP=PPTP+L2F sets up tunnels across the network	Secure across geo-locations	Complex to configure, migrate and inter-operate
IEEE 802.1ad -2005[5]	Switched or stacked VLANs	Multiple VLAN (QinQ) tags in Ethernet frame	L2	Multi-domain using cross connect	Similar to VLAN	Multiple VLANs in service provider n/w	Scaling infeasible (learning MAC of all users); tags overhead
NFV-2012[6]	Decouple network & hardware	Chaining virtual functions for services	Seperate control & data plane	Scalable if inter-operable functions	Commodity hardware + virtualization	Low cost, power, time to market; high efficiency	IWF to inter-connect with PNFs; increased overhead
Renovate -2017[7]	Failure handling	Analytics for faster network recovery	Booting failed VNs based on priority	C++ Implementation	Tested on 65 nodes	Alternate method	Optimization unavailable
RT-VNE -2017[8]	Resource allocation	Online resource allocations for VNEs	Divide time slots, fulfill/ defer requests	C++ Implementation	Tested up to 100 nodes	Coordinated node/link mapping phases	VN acceptance ratio
COVE -2018[9]	Resource allocation	Virtual network embedding	Hybrid VNE (central control +trusted nodes)	Used VNE tool (Embed) – 100 nodes, 500 links.	Mid-size ISP	Topology aware resource allocation	Average revenue and acceptance ratio

their utility in scenarios that might benefit from custom metrics and deeper cross layer optimization [14], [15].

B. The Named-Object Abstraction

In this paper, we argue that to implement an end-to-end virtualization architecture, a shared common abstraction should be provided to all involved layers. Towards this, we adopt and present the named-object abstraction as a candidate solution. Named-objects are an abstraction meant to represent any network entity that could be abstracted as an addressable network element. This can cover any possible abstraction: from the original host based abstraction of a virtual link bridging two interfaces, to recently introduced ones such as contents, to any potential future abstraction, *e.g.*, context and services. While name based approaches have already been considered in the past, they were mostly focused on either solving specific issues such as mobility [16] or security [17] or to shift the communication focus to new entities such as content [18]. Named-objects aim to bring a more comprehensive solution that can enable powerful abstractions and services to underpin the Internet architecture.

Figure 1 outlines the general approach behind the named-object abstraction definition: abstracting resources through separation of names and addresses. Separating names —*i.e.*, identities— from addresses has been advocated by the research community for quite some time [17], [16], [19], [20] and has inherent benefits in handling mobility and dynamism for one-to-one communications. If properly employed, names can also provide additional advantages to facilitate the creation of new service abstractions that can be used to support advanced applications.

The named-object approach achieves its goal through three steps: First, *"what"* (or *"who"*) will take part in the communication is identified through a unique name, or Globally Unique Identifier (GUID), that is understandable by all parties involved (*e.g.*, end points, routing elements, etc.). When forwarding

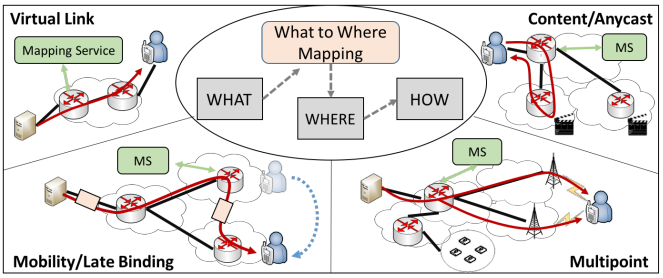

Fig. 1. The Named-Object abstraction applied to different use cases.

is required, names are then resolved to *"where"* they are located. While this could be applied at different locations of the network and in the network stack (*e.g.*, having the separation at the end points), previous proposals [16], [21], [22] demonstrated that the use of a globally accessible Name Resolution Service (NRS) is a suitable approach for this goal, scaling to globally support the size of the namespace while supporting the dynamicity of hybrid routing schemes, *i.e.*, less than 100ms for 95th percentile of lookup operations. Finally, if the semantical value of such element is known, it can be indicated through the use of a service identifier properly located in a packet header, giving an indication on *"how"* such packet should be treated. For example, this can express specific flow requirements or the type of communication in place (*e.g.*, content retrieval).

III. VIRTUAL BASE STATION

Layer 2 network virtualization enables the sharing of physical network access technologies, including both wired and wireless mediums. Solutions for wired networks have existed for more than 30 years [3] but, with the vast adoption of mobile devices, new solutions became necessary to support wireless

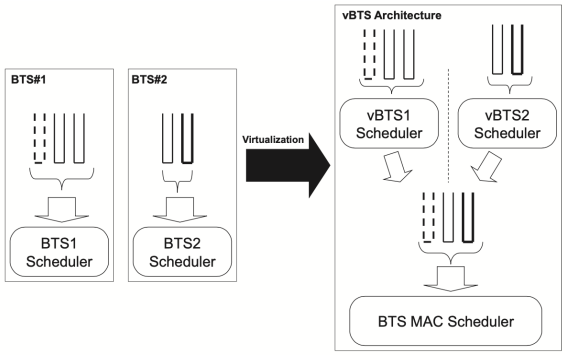

Fig. 2. **Service flow abstractions on a virtualized base station**

technologies. Sharing a wireless access network requires providing abstractions to share both the networking infrastructure as well as physical radio resources. Multiple solutions across different wireless technologies [23], [24], [25], [26] have been proposed with the goal of supporting multiple co-existing independent and customizable logical (virtual) wireless networks on top of a shared infrastructure, including specific designs for WiFi [27], WiMax [28], and LTE [29] networks.

Integrating a virtualized wireless access network requires to: provide mechanisms to map network flows to each virtualized base station (§III-A), specify service classes for QoS purposes (§III-B), and finally enforce the QoS requirements at the radio level (§III-C). In this paper, we focus our discussion on our previous solution aimed at the virtualization of a WiMax base station [28], which we call Virtual Base Station (vBTS), but similar design challenges and choices can be applied across most access technologies. In the rest of the section, we detail the vBTS design and map it to the named-object abstraction (§III-D).

A. L2 Datapath To BTS.

Layer 2 frames have to be forwarded from and to the physical base station. To ensure that the datapath can work in a wide range of application environments, the frame transport mechanisms have to satisfy three core requirements: (1) Protocol independence, *i.e.*, should be independent of any layer-3 mechanism; (2) Slice traffic separation, *i.e.*, some mechanism by which traffic can be separated across multiple virtual base stations; Finally, (3) transparency, *i.e.*, the implemented datapath mechanism should be transparent to the slices. In our previous work [28] we meet these requirements by implementing an L2 over L3 tunnel. First, protocol independence is achieved because of tunneling, and the grouping of slice traffic is achieved by separating packets within different IP tunnels. This mechanism also allows geographic decoupling of the virtual base station substrate and the ASN-GW allowing them to be housed on different networks as long as the required tunnels are established.

B. Service Flow Abstraction.

Virtualizing a base station requires the ability to map different service flow types which are defined in the physical base station to the ones that are exposed to each of the virtual ones. Specifically, solving the core issue of provisioning service classes on the physical BTS and the mechanisms for mapping service classes within the vBTS to the service classes in the physical BTS. Conventional cellular technologies, *e.g.*, WiMAX [30], rely on a connection oriented MAC for providing quality of service differentiation across wireless clients. In particular, the MAC scheduler works with logical entities known as service flows which define a unidirectional flow of packets either from the BTS to the clients, or the other way around and allow the definition of certain QoS features, *e.g.*, maximum/minimum throughput, maximum delay, and other features such as the ARQ mechanism.

To extend the service flow idea to the virtual base stations (vBTSs), we define new virtual service flows. These virtual service flows are used to map the underlying service flows in the base station. For the specific WiMax solution [28], the identifier for these virtual service flows consists of the slice-id and the flow-number. In our work, we map virtual service flows to physical service flows are as depicted in the Figure 2. We define a broad set of service flows in the BTS and reuse them by mapping multiple service flows from the vBTSs to the same flow in the BTS. By using a port address classifier for service flow determination, we can multiplex multiple virtual service flows from different vBTSs on the same service flow in the BTS. This provides an opportunity for conservation of physical service flows at the BTS.

C. Radio Isolation

Once the virtual service flows from the vBTSs have been mapped to the physical radio, we need a mechanism by which we can limit the amount of total radio resources consumed by the virtual service flows within a slice. The problem can be formulated as the amount of time allocated to a single flow or wireless client. Where airtime is defined in terms of the set of time and frequency blocks of the radio. To enforce this radio airtime fairness across slices in WiMax, we proposed the virtual network traffic shaping mechanism (VNTS) [31]. This mechanism is responsible for adaptively determining usage of resources by every slice slice and limiting the traffic flowing into the BTS scheduler from every slice.

D. Named-object Based L2 Virtualization

We previously described the three mechanisms required for implementing L2 network virtualization: datapath tunneling, service flow mapping, and radio isolation. Ultimately, all three components require mapping virtualized resources using flow information (*e.g.*, IP pairs or slice-id/flow-number). Both techniques can be implemented by relying on a unified named-object abstraction and two built-in mechanisms: (1) GUIDs are used to uniquely identify the resources mapping across layers; (2) Information regarding the virtual network is stored

in the NRS, storing information on the intrinsic requirements to apply to the local flows.

In our previous work, we extended the vBTS framework to implement L2 network virtualization mapping via named-objects [32]. In our solution, we implemented a push-based mechanism, where a global service resource manager communicates with local virtual network controllers to inform them about upcoming traffic classes and the creation of their entries in the NRS. During the setup phase, the service resource manager initiates the setup by pushing to the participating nodes the unique identifier that characterizes the Virtual Network. The unique tag, called Service ID in our work, is used to notify the local controllers how to identify the GUID that will be found in packets belonging to the VN. In the following step, the local network controller creates the requested service specific vBTSs by configuring the physical BSs and network equipment in the backhaul.

In conclusion, this process closely reflects common tag-based virtualization techniques, commonly used in wired network environments. We extend this approach to cellular networks by supporting a logically centralized name repository that permits mapping local resources across networks to specific service requirements.

IV. Native L3 Virtualization

Virtual Networks are widely used to manage networks inside the Internet architecture. Thanks to their ability support the illusion of a customized network with a user-specified topology, they enable the ability to match deterministic QoS characteristics [33], a key requirement for modern applications. Unfortunately, existing VN solutions lack the ability to synchronize resources across multiple network domains, bypassing the inherent need for services and applications to interconnect distributed computing resources (*e.g.*, edge clouds) spread across multiple network vantage points. Overlay based solutions solve this problem, but at the expense of increased overhead and lack any access to the underlying network environment.

Ideally, VN solutions should work across multiple network domains to offer service providers the ability to cooperatively exploit network virtualization to enhance distributed service architectures enabling the same level of integration and performance guarantees as single domain virtual networks. From this analysis, we identify the network layer as the right level to host a Virtual Network design that can work across network borders. Layer 3 is by definition where protocols are used to interconnect networks resources. Extending it to support virtualization provides the most natural solution to conveniently support interconnecting resources that span multiple networks.

In our previous work [34], we presented the Named-Object Virtual Network architecture (*NOVN*), a solution that addresses the fundamental issues of virtual network management and deployment support at the network layer (L3) of infrastructure. Figure 3 lists the set of core design operations that are

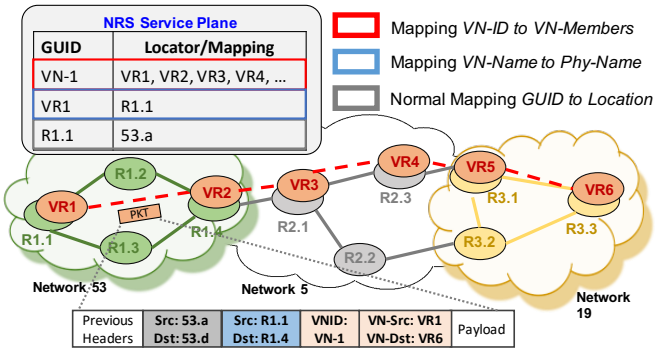

Fig. 3. NOVN design

at the base of the framework. NOVN exploits the named-objects abstraction to support network virtualization at Layer 3: NOVN takes advantage of the availability of a globally accessible NRS capable of storing mappings from *names* to list of *values* as well as the flexibility of accessing names and addresses as part of a network header to implement a unified virtualization language that can be exploited across networks. The next paragraphs highlight NOVN's main design characteristics: (1)The use of named-objects to define the VN topology (§IV-A); (2) The routing and forwarding mechanisms (§IV-B); and, (3) How QoS rules are enforced in the participating routers (§IV-C). Finally, we discuss how the proposed architecture could be incrementally deployed on the current network infrastructure (§IV-D).

A. Logical Definition of a VN through Naming.

NOVN simplifies the definition of the virtualized logical layer through information offloading to the NRS. This is done as a three step process: first, 1) a unique identifier is assigned to the VN and a mapping from such name (*VNID*) to all participating resources is stored in the naming service (red box in the Figure); referenced resources are identified with a name that has meaning only within the limits of the VN logic —*i.e.*, they are unique and not shared across different VN instances; this provides the dual function of simple access and distributed information recovery. 2) Each VN resource name, is then mapped into two values: a) the name identifying the resource the virtualized element is running on top and b) the list of its neighbors. Finally 3) these identifiers are mapped into physical a Network Addresses (NA) allowing for normal forwarding operations. Fully defining the network topology through naming allows to solve two core issues to be handled separately: the *local* problem of mapping virtual to physical resources and the *global* problem of coordinating the virtualized logic across domains.

B. Routing & Forwarding.

Routing information is exchanged across nodes through control packets encapsulated accordingly in order to reach participating nodes. Similarly, data forwarding happens on a hop-by-hop manner across routers of the virtual network.

When a data packet reaches one of these routers and a routing decision is taken, the packet is encapsulated within an external network header that contains information to reach the next VN router (shown in Figure 3). At nodes not participating in the protocol, normal routing decisions are taken using the external network header. As names identify each hop, forwarding can happen independently from the physical network configuration.

C. QoS Support.

To support QoS control and network slicing, NOVN introduces the concept of a VNID based mapping technique. The resource management is achieved by marking the incoming packets and then classifying them according to their VNIDs. The classified packets are stored into a buffer which are pulled by a bandwidth shaper at a specified rate before sending at the output port. This simple VNID based classification and shaping technique enables NOVN with the resource provisioning and traffic shaping, and therefore aids in the network slicing.

D. Incremental Deployment

The presented NOVN design relies on numerous functionalities available if deployed on top of a named-object based architecture, *e.g.*, MobilityFirst [20]. This includes: (1) packet network headers with dedicated space for accessing object names and (2) addresses and service identifiers that enable hybrid routing. While these functionalities simplify and enhance the framework design, and allows operating without the need for any additional overlay protocol, alternative solutions can be considered to incrementally deploy NOVN on the current network infrastructure. We identify two different approaches:

Overlay. Fully overlay approaches represent a flexible way for deploying experimental networks and protocols on top of the existing infrastructure. Through encapsulation of network packets on top of UDP packets and tunneling across participating nodes, they allow for the quickest solution to implement experimental protocols on top of the existing infrastructure. With this solution, flexibility and simplicity come at the cost of additional overhead. The named-object abstraction helps taking advantage of the abstraction layers presented to simplify the implementation requirements for such solutions. First, as the network address that represent tunnels are overloaded into the GNRS, no need to define a-priori tunnels is required, leaving forwarding decisions to be dynamically resolved at running time, as needed. This approach is similar in spirit to LISP [16], where multiple encapsulated headers can be used to traverse networks and reach participating routers, but extends the base name space to provide the more advanced named-object abstraction.

Shim Layer. Balakrishnan *et al.* [1] proposed to employ a service layer, called L3.5, in between the network layer and the above protocols. This new layer has the role of supporting advanced network delivery mechanisms and the direct addressing of computing services at the edge of the network. Similarly, the named-object abstraction used by NOVN could be implemented in such shim layer, integrating source and destination names, as well as concepts such as the VNID. In this case, packets are tunneled between hops and at each step nodes use the shim layer to obtain the name information and continue the forwarding process.

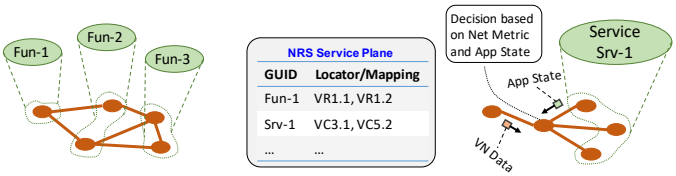

Fig. 4. Application Specific Advanced Routing in NOVN

V. End-to-End Virtualization

Mobile cloud services provide a scalable and dynamic way to support emerging technologies, *e.g.*, as IoT (Internet-of-Things), and applications, *e.g.*, augmented or virtual reality. Edge clouds require dealing with a mix of computing and networking resources with complex cross-layer interactions and considerable heterogeneity in both networking and computing metrics across the region of deployment. Conventional large datacenters have addressed this problem by requiring uniformity in the network fabric and using software-defined network (SDN) technologies to assign resources in a logically centralized manner. However, for a distributed architecture, a key requirement arises due to the need of dynamic allocation of cloud processing requests across available edge computing and networking resources [35]. Further, edge clouds promise to support tighter closed loop low latency applications which requires a seamless integration of services with the network entities whose performance can be monitored, reported, and enhanced. The listed requirements therefore require designing mechanisms that can blend service state parameters into the network to create a fully virtualized end-to-end QoS-enabled system.

In the remaining of this section, we first present ASR §V-A, the underlying technology used to address computing points distributed in the network. Finally, we discuss how all components are integrated into an end-to-end virtualization architecture §V-B.

A. Application-Aware Routing

To support edge cloud based services and applications, we implemented into NOVN advanced routing and forwarding mechanisms through a technique called Application Specific Routing (ASR) [36]. ASR defines a mechanism aimed at exploiting a comprehensive set of cross-layer information from both network and application layers to enable custom delivery mechanisms, giving service providers the flexibility to incorporate parameters which allow for utilizing information above the network layer for routing decisions. For example, consider the case of a service deployed at multiple locations across different domains: application state could be exploited to implement advanced anycast delivery based on network metrics as well as service load at the end points.

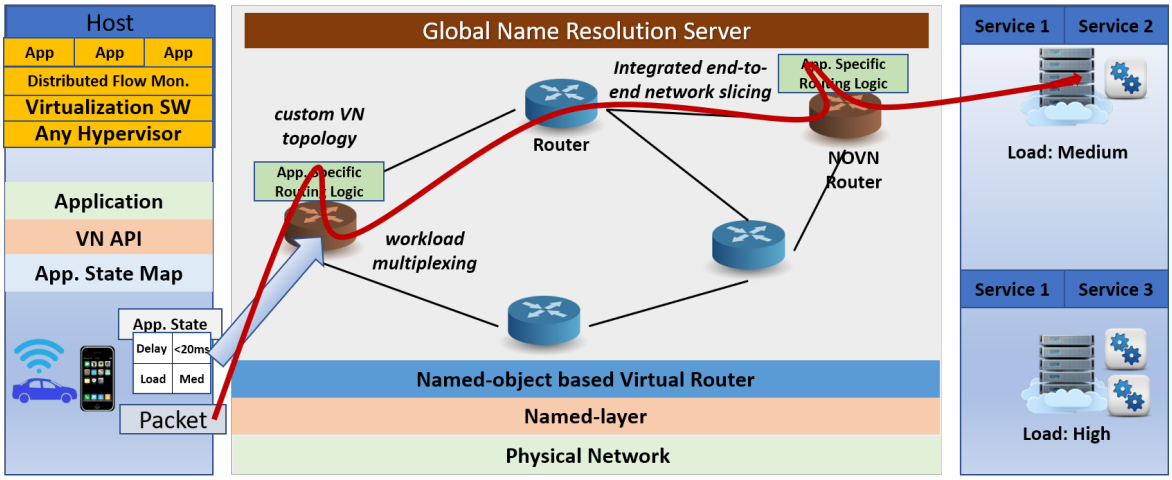

Fig. 5. Overview of an application-aware Full Stack Virtualization architecture

Two key technology components are required and introduced into the NOVN framework to support ASR: (1) the ability to aggregate multiple service instances under a single object name, a natural extension of the named-object abstraction. (2) the ability to make application nodes participate in the routing protocol by sharing their application state through the network API. NOVN supports the first one by offloading the list of participant locations under a single name into the name resolution service. The second component is supported through the implementation of a custom routing protocols deployed on top of any underlying infrastructure and integrating end point APIs to push application state into the VN.

The support of advanced cross-layer routing mechanism is illustrated in the Figure 4. Consider the scenario where a collection of servers offer a service to its clients. In this scenario, NOVN and ASR provide the base to deploy such distributed tools by: (a) allowing push of state to participating nodes and (b) make use of the named-object abstraction to support advanced anycast delivery to service instances based on both network and application metrics. The state information distributed across and within the network consists of routing metrics for full-stack virtualization such as latency (applicaition-level requirements), inter-edge cloud bandwidth (measured/estimated), server workload (utilization), server compute capacity (in GFLOPS [37]), and the router buffer size (for backpressure and flow control). Thus, at the distributed branching locations, routers can then take informed decisions thus providing low-latency and scalable support.

B. Dynamically Configuring a Name-Based Virtual Network

The different architectural components presented are orchestrated to create an end-to-end virtualization that can fulfill low-latency requirements of future applications holistically using compute, networking, and storage virtualization. To achieve a fully functional architecture a series of coordination techniques are required among the three core components. The key integration point is a mechanism for building an integrated virtual network by bridging a named-object based virtual network on the entire end-to-end path.

Figure 5 shows an overview of the virtualization framework developed and its core components. The architecture workflow can be summarized into three phases: (1) QoS managment, (2) metric injections, and (3) clients association.

End-to-end network QoS management. The first phase consists of the end-to-end VN setup phase. In this phase, a centralized or distributed resource manager is in charge of initiating the VN setup by pushing to the participating nodes the definition of requested resources and the unique identifier that characterize the Virtual Network, *i.e.*, the VNID. This configuration propagates to both the network routers as well as to the access network which sets up the physical BSs and backhaul to integrate the QoS requirements of the VN. Finally, the unique tag is then used by the network resources to identify packets that belong to the VN.

Service metrics injection. Once the VN network topology is established, service end points use the ASR protocol to propagate the compute performance indicators, *e.g.*, a function of server load or its compute capacity, to the participating virtual nodes. This information is used to compute and keep up to date the ASR routing tables that implement the service QoS specific delivery mechanisms.

Clients association. Finally, the clients of the service associate with the ingress VN node at the access network. The device unique identifier and its MAC address are registered in the entry point of the network so that the virtualized wireless technology can apply the transmission policies requested at configuration time. Once the association is completed, the end-to-end application service tasks can benefit for the datapath delivery mechanisms built into the VN.

VI. Evaluation

We evaluate an instance of the end-to-end named-object based virtualization architecture via a prototype deployment.

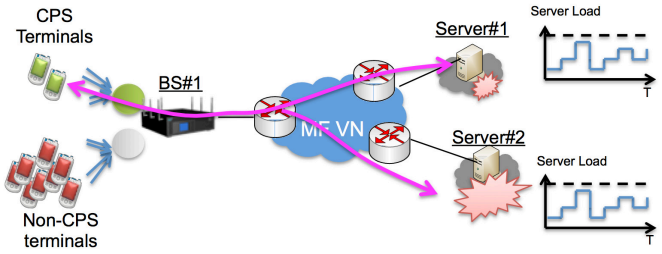

Fig. 6. Experimental Scenario

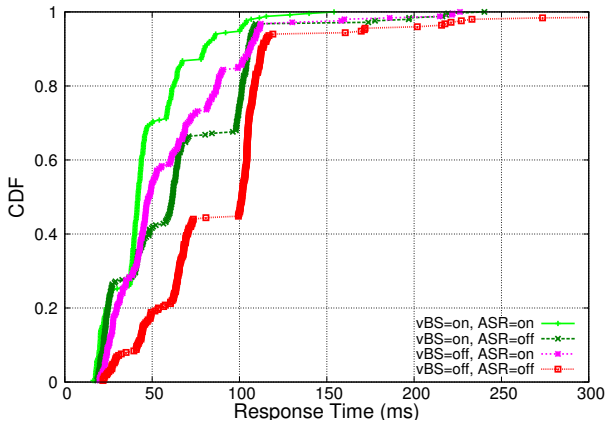

Fig. 7. CDF of CPS Response Time (Data Unit Size = 25KB)

This work was done in collaboration with NICT Japan, in a series of experimental ORBIT testbed evaluations first initially by Nakauchi *et al.* [32]. Our prototype integrates two components: (1) a WiFi based virtual Base Station (vBS) [27], a virtual network framework for L2 WiFi networks; and 2) the ASR enabled NOVN prototype [34] for L3 network virtualization with service aware routing mechanisms.

Figure 5

We model an experimental scenario where a Cyber Physical System (CPS), a typical application that requires strict service response time, sends compute requests to a service deployed using edge clouds. In the deployment, multiple service instances are present are across different networks and are interconnected using our virtualization architecture. We emulate the CPS application by generating closed-loop (round-trip) UDP traffic at periodic time intervals. In our experiments, we deploy a CPS-specific virtual network that shares the infrastructure with non-CPS traffic that is treaded in a best-effort fashion. When the dedicated vBSs services are activated, CPS and non-CPS traffic are completely isolated and the CPS response time can be reduced. On the other hand, without an active virtual network to handle CPS traffic, the shared bottleneck causes an increase in the response time.

We emulate edge computing cloud servers using nodes of the ORBIT infrastructure. Every 10 seconds, each server randomly chooses a server load from a preconfigured parameter set, *i.e.*, {0.2, 0.4, 0.6, 0.8}, linearly increasing processing times by {20, 40, 60, or 80} ms. Participating servers announce their load every two seconds to the VN routers. Thus, the ASR routing table is updated accordingly. When the ASR is activate, NOVN routers forward CPS traffic to the less-loaded server.

We setup three CPS terminals and 12 non-CPS ones per physical base station. We configure two physical base stations, *i.e.*, 30 terminals are configured in total. This means that when the VNs are enabled, the CPS-specific VNs accommodate six CPS terminals and the remaining 24 non-CPS terminals are accommodated by the non-CPS communication channels. On the other hand, without VNs, each physical BS accommodates three CPS terminals and 12 non-CPS ones. We generate non-CPS traffic in the form of 100KB data units transmitted every second by each non-CPS terminal. Finally, the the traffic load offered to the 12 non-CPS terminals is 9.65 Mbps.

Figure 7 shows the cumulative distribution function (CDF) of CPS response time. 300 data units with 25KB size are generated in this experiment. We notice that the combination of vBS, NOVN, and ASR outperforms all other cases. In our results, we focus on response times of less than 100 ms — *i.e.*, the time required for the human brain to perceive and interaction as instantaneous. In particular, we focus on how many data units meet the requirement. In the case of vMCN, *i.e.*, (vBS, ASR) = (on, on), 94% data units achieves less than 100 ms response time. On the other hand, in the case of (vBS, ASR) = (off, on), (on, off), and (off, off), the value decreases to 85%, 74%, and 46%, respectively. These results show ASR has larger impact to lift up the CDF line, We can conclude the vMCN can support up to 94% CPS cycles under the set goal of 100 ms, and outperforms the baseline system by almost 2x.

The specific percentile response time is also an important performance index to evaluate a real-time system. We evaluate the performance in the congested WiFi environment, we determines to evaluate 90 percentile response time. In the case of (vBS, ASR) = (on, on), the 90 percentile response time is 80 ms. On the other hand, in the case of (vBS, ASR) = (off, on), (on, off), and (off, off), the value increases to 106 ms, 105 ms, and 113 ms, respectively. These results show vBS and ASR have approximately the same level of impact. We can conclude the vMCN can achieve less than 100 ms for 90 percentile response time for 25KB CPS data units.

VII. Conclusions

This paper presents an approach to deploy integrated virtual networks using the named-object abstraction to create a holistic end-to-end virtualization platform. We exploit the inherent benefits of the named-object network abstraction to handle dynamic network mapping and resources reallocation across network technologies. Further, the framework design is augmented with core extensions such as application-aware cross layer routing (ASR) and QoS support, providing an architecture that can seamlessly support future application requirements. An evaluation is carried out on the ORBIT testbed for a sample application running over an emulation of an edge cloud network infrastructure. Performance results demonstrate a latency performance gain of as much as 60%

when compared with the baseline implementation without cross layer optimizations.

Acknowledgments

We are deeply grateful to all the many contributors that made this work possible, including all the contributors to the MobilityFirst project as well as Dr. Nakauchi and Dr. Shoji at NIST. This research was supported in part under the NSF Future Internet Architecture - Next Phase (FIA-NP) grant CNS-1345295 and the NSF JUNO grant CNS-1404118.

References

[1] H. Balakrishnan, S. Banerjee, I. Cidon, D. Culler, D. Estrin, E. Katz-Bassett, A. Krishnamurthy, M. McCauley, N. McKeown, A. Panda *et al.*, "Revitalizing the public internet by making it extensible," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 2, pp. 18–24, 2021.

[2] F. Bronzino, S. Mukherjee, and D. Raychaudhuri, "The named-object abstraction for realizing advanced mobility services in the future internet," in *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2017, pp. 37–42.

[3] V. Rajaravivarma, "Virtual local area network technology and applications," in *Proceedings The Twenty-Ninth Southeastern Symposium on System Theory*. IEEE, 1997, pp. 49–52.

[4] H. A. Seid and A. Lespagnol, "Virtual private network," Jun. 16 1998, uS Patent 5,768,271.

[5] L. Fang, N. Bitar, R. Zhang, and M. Taylor, "The evolution of carrier ethernet services-requirements and deployment case studies [next-generation carrier ethernet]," *IEEE Communications Magazine*, vol. 46, no. 3, pp. 69–76, 2008.

[6] N. F. V. ETSI, "Network functions virtualisation (nfv)," *Management and Orchestration*, vol. 1, p. V1, 2014.

[7] N. Shahriar, R. Ahmed, A. Khan, S. R. Chowdhury, R. Boutaba, and J. Mitra, "Renovate: Recovery from node failure in virtual network embedding," in *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 2016, pp. 19–27.

[8] L. D. Nguyen, N. Kim, S. Kim, and C.-K. Kim, "Rt-vne: A real-time strategy for virtual network embedding towards resource efficiency," in *2017 international conference on information networking (ICOIN)*. IEEE, 2017, pp. 185–190.

[9] M. Feng, J. Liao, S. Qing, T. Li, and J. Wang, "Cove: Co-operative virtual network embedding for network virtualization," *Journal of Network and Systems Management*, vol. 26, no. 1, pp. 79–107, 2018.

[10] E. Bell, A. Smith, P. Langille, A. Rijhsinghani, and K. McCloghrie, "Definitions of managed objects for bridges with traffic classes, multicast filtering and virtual lan extensions," *RFC 2674 (Proposed Standard), Internet Engineering Task Force*, 1999.

[11] E. Rosen, A. Viswanathan, R. Callon *et al.*, "Multiprotocol label switching architecture," 2001.

[12] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 3–14.

[13] K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "Live data center migration across wans: a robust cooperative context aware approach," in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*. ACM, 2007, pp. 262–267.

[14] V. Valancius, N. Feamster, J. Rexford, and A. Nakao, "Wide-area route control for distributed services." in *USENIX Annual Technical Conference*, 2010.

[15] X. Wu and J. Griffioen, "Supporting application-based route selection," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–8.

[16] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The locator/ID separation protocol (LISP)," Internet Requests for Comments, RFC 6830, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6830

[17] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol," Internet Requests for Comments, RFC 5201, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5201

[18] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*. ACM, 2007.

[19] J. Pan, S. Paul, R. Jain, and M. Bowman, "Milsa: a mobility and multihoming supporting identifier locator split architecture for naming in the next generation internet," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 2008, pp. 1–6.

[20] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[21] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 698–707.

[22] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," in *ACM SIGCOMM Computer Communication Review*. ACM, 2014, pp. 247–258.

[23] S. Paul and S. Seshan, "Virtualization and Slicing of Wireless Networks," *GENI Design Document 06-17, GENI Wireless Working Group*, September 2006.

[24] X. Wang, P. Krishnamurthy, and D. Tipper, "Wireless Network Virtualization," *Proc. ICNC '13*, January 2013.

[25] C. Liang and F. R. Yu, "Wireless Network Virtualization: A Survey, Some Research Issues and Challenges," *IEEE Comm. Surveys and Tutorials*, vol. 16, no. 3, July 2014.

[26] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey," *Mobile Networks and Applications*, September 2014.

[27] K. Nakauchi and Y. Shoji, "WiFi Network Virtualization to Control the Connectivity of a Target Service," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 308–319, June 2015.

[28] G. Bhanage, I. Seskar, and D. Raychaudhuri, "A virtualization architecture for mobile wimax networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 4, pp. 26–37, 2012.

[29] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte mobile network virtualization," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.

[30] "Ieee mobile wimax standard," http://standards.ieee.org/about/get/802/802.16.html.

[31] G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri, "Vnts: A virtual network traffic shaper for air time fairness in 802.16 e systems," in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–6.

[32] K. Nakauchi, F. Bronzino, Y. Shoji, I. Seskar, and D. Raychaudhuri, "vmcn: virtual mobile cloud network for realizing scalable, real-time cyber physical systems," in *Proceedings of the 4th Workshop on Distributed Cloud Computing*. ACM, 2016, p. 6.

[33] J. Hwang, K. K. Ramakrishnan, and T. Wood, "Netvm: High performance and flexible networking using virtualization on commodity platforms," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, 2015.

[34] F. Bronzino, S. Maheshwari, I. Seskar, and D. Raychaudhuri, "Novn: A named-object based virtual network architecture to support advanced mobile edge computing services," *Pervasive and Mobile Computing*, vol. 69, p. 101261, 2020.

[35] S. Maheshwari, S. Choudhury, I. Seskar, and D. Raychaudhuri, "Traffic-aware dynamic container migration for real-time support in mobile edge clouds," in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2018, pp. 1–6.

[36] F. Bronzino, S. Maheshwari, I. Seskar, and D. Raychaudhuri, "Novn: named-object based virtual network architecture," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*. ACM, 2019, pp. 90–99.

[37] S. Maheshwari, P. Netalkar, and D. Raychaudhuri, "Disco: Distributed control plane architecture for resource sharing in heterogeneous mobile edge cloud scenarios," in *40th IEEE International Conference on Distributed Computing Systems (ICDCS) November, 2020, Singapore. IEEE*. IEEE, 2020.