NOVN: Named-Object Based Virtual Network Architecture[*]

Francesco Bronzino[†]
Inria, Paris
francesco.bronzino@inria.fr

Sumit Maheshwari
WINLAB, Rutgers University
sumitm@winlab.rutgers.edu

Ivan Seskar
WINLAB, Rutgers University
seskar@winlab.rutgers.edu

Dipankar Raychaudhuri
WINLAB, Rutgers University
ray@winlab.rutgers.edu

ABSTRACT

This paper presents *NOVN*: a novel network virtualization framework aimed at providing an efficient and low overhead solution for deploying Virtual Networks (VNs) based on the concept of named-objects. Name based communication paradigms, realized by separating object names and network addresses through a logically centralized globally distributed Name Resolution Service (NRS), can be used to build a natural and efficient architecture for virtual networks. The proposed VN framework exploits the name based abstraction to create customized networking for distributed services which benefit from an awareness of network topology and routing. A specific example of edge cloud computing is presented in which VN's are used to realize "application specific routing" (ASR) for efficiently connecting users with cloud resources. Experimental results are presented for validation of the proposed VN architecture using a software router implementation running on the ORBIT testbed. The results validate the feasibility of the named-object approach, showing minimal VN processing, control overhead, and latency. The results also validate application aware ASR routing functionality for an example latency constrained edge cloud service scenario.

CCS CONCEPTS

• **Networks → Network services**; *Cross-layer protocols*; *Overlay and other logical network structures*;

ACM Reference Format:
Francesco Bronzino, Sumit Maheshwari, Ivan Seskar, and Dipankar Raychaudhuri. 2019. NOVN: Named-Object Based Virtual Network Architecture. In *International Conference on Distributed Computing and Networking (ICDCN '19), January 4–7, 2019, Bangalore, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3288599.3288637

[*]Research supported under NSF Future Internet Architecture - Next Phase (FIA-NP) grant CNS-1345295 and NSF JUNO grant CNS-1404118
[†]Work done while at Rutgers University

1 INTRODUCTION

Mobile cloud services are expected to grow rapidly in the next few years due to continuing large-scale adoption of smartphones as well as emerging technologies such as IoT (Internet-of-Things) and augmented or virtual reality (AR/VR). Faced with increased service requirements, infrastructure and service providers have responded to the network architecture evolution by starting a process of *localization* of their resources. In particular, providers are increasingly aiming to distribute their service points of presence (i.e. processing and storage) in order to exploit locality and serve their clients right at the edge of the networks they are connected to. The industry and research communities alike have embraced this approach and are proposing solutions known as edge clouds [32] or fog computing [4] that can better scale and provide low delay services to real-time applications.

Distributed solutions like edge clouds are conceptually simple and elegant. Moreover they offer the potential to meet strict service requirements (e.g. low latency). This comes at the cost of facing significant technical challenges associated with moving cloud processing from a centralized data center to a loosely coupled set of servers located at the edge of the network. One central challenge is that of distributed control: by their very nature, edge clouds are placed in multiple network domains with heterogeneous bandwidth and latency properties without a single point of control. A second key challenge arises from heterogeneity of computing resources and the limited amount of computational power edge systems can be equipped with. In contrast to the previous data center driven cloud model, edge clouds are often colocated with the existing network equipment and deploy limited computational resources. This implies the need for distributed resource management (i.e. task assignment, load balancing and application-level quality-of-service management) across heterogeneous edge computing resources.

Virtual Networks (VNs) have been proposed as a means of connecting resources across the internet, supporting the illusion of a customized network with user-specified topology, security and performance characteristics matched to application requirements. Depending on the purpose, different techniques have been applied at different layers of the networking stack in order to realize virtual networks. Cloud networks have been one of the main adopters of virtual networks, with VN techniques being used to abstract the distribution of physical and logical resources - e.g. applications, databases and more - within data centers, allowing for flexible management techniques [9, 16]. Thanks to the simplicity of the solution, together with new technologies like SDN, LAN based VNs allow for a powerful and efficient framework for coordinating resources within a data center.

While VLAN based solutions employed within data centers would be highly attractive to solve this challenge, they cannot scale outside of a single domain. Existing solutions that can work across domains either only support point to point connectivity between remote cloud locations [27, 38] or are based on overlay solutions (e.g VINI [2]). Overlay VN solutions, while flexible, may incur high overhead and lack the visibility of underlying network layer performance parameters, limiting their utility in scenarios that might benefit from custom metrics and deeper cross layer optimization [35, 39].

Recognizing the need to provide a solution that offers the logical simplicity of L2 network virtualization while offering the flexibility to control traffic across network domains, this paper presents *NOVN*, a virtual network solution that exploits the concept of named-objects [5] introduced in the MobilityFirst future Internet architecture [28] to realize a logically clean, easily deployable, virtual networking framework at Layer 3. *NOVN* tackles the control mechanism challenge by applying name indirection to create clean partitions across logical layers (Figure 1). First, physical network resources are mapped to globally unique names, eliminating the need of continually tracking routers addresses and possible configuration changes. A second layer of abstraction then maps network elements to the participants of the virtual network, creating a logical network on top of the infrastructure.

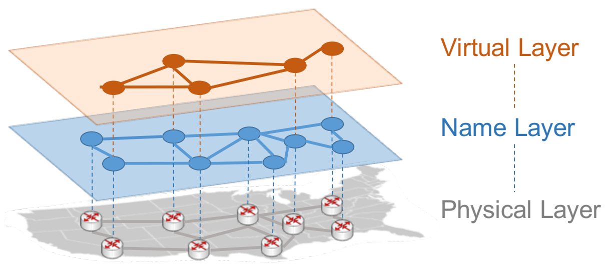


Figure 1: *NOVN* layers of abstraction.

Building on top of this layering concept, *NOVN* takes inspiration from recent attempts to enhance applications by allowing them to provide hints to the network to optimize routing decisions [11, 35, 39] and implements a novel technique called Application Specific Routing (ASR). ASR offers applications a solution for pushing small snapshots of compute status data into the virtual routing fabric providing a control environment for distributed services on top of limited edge resources. For example, consider a mobile edge cloud scenario where the application goal is to connect mobile devices to the "best" edge cloud server: while in a normal networking environment "best" might correspond to the "nearest", in heterogeneous environments, varying computing loads might require delivery to a lightly loaded cluster which is not necessarily the closest one in terms of network distance. Through ASR, *NOVN* can support advanced *anycast* delivery service allowing virtualized routers to consider application status and perform custom routing decisions.

In summary, the main contributions of this paper are:

- Design of *NOVN*, a Layer 3 virtual network framework that can provide the control mechanisms to connect distributed resources across network domains.
- Starting from the *NOVN* framework, we develop routing mechanisms that exploit the abstractions of the architecture to support distributed edge-cloud services. This technique, called ASR, supports routing service requests based on cross-layer information extracted from network and application.
- Finally, we develop a working implementation of *NOVN* and ASR based on Click [15] and the MobilityFirst network architecture software prototype [7]. As part of this effort, we present experimental results obtained to validate the NOVN and ASR concepts, demonstrating significant latency improvements for real-time applications.

2 EDGE CLOUD REQUIREMENTS

In this section we discuss the requirements imposed by Edge Clouds on developing a Virtual Network to interconnect distributed resources. Starting from a review of existing virtualization techniques, we discuss the need for the introduction of Layer 3 virtualization.

2.1 Edge Cloud Requirements

Edge clouds are highly distributed architectures that require loosely coupled coordination mechanisms to operate. Resource allocation in edge clouds is more difficult than in a data center. This is due to the fact that edge clouds do not have the law-of-large-numbers advantage of a data center which aggregates requests from tens of thousands of users. Instead, they must deal with requests from smaller numbers of users characterized by significant randomness in both the spatial and temporal dimensions. Due to their physical presence in multiple network domains and the type of resources they deploy, the following requirements are identified in contrast to the ones usually presented by datacenter based clouds.

Cross Domain Connectivity. Management of distributed cloud resources becomes more complex when the edges extend across multiple domains. A key requirement for this scenario is to be able to synchronize resources to coordinate and communicate state potentially across multiple domains managed by different commercial entities.

Dynamic Re-Routing. Due to the nature of IP addresses, any configuration change cased by failure or resource migration requires to reconfiguration of connectivity between edge computing resources. The new information has to be propagated across all the participating entities. This can – and often does – cause all ongoing traffic to be lost. This is due to packets not being able to carry the necessary information to self-correct temporary errors. Approaches to reduce this impact have been explored [37], but require the creation of dedicated control channels to maintain persistent traffic flow.

Support Cross-Layer Interactions. 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. A key requirement of the distributed architecture is that of dynamic allocation of cloud processing requests across available edge computing and networking resources.

2.2 Existing Network Virtualization Solutions

Existing solutions can be roughly grouped into two categories: tag based virtualization at Layer 2 and overlay based Layer 7 solutions.

Tag Based Virtualization. Tag based approaches exploit flat unique identifiers placed at different layers of the network stack to uniquely identify packet flows. Example of this are VLANs [3] and MPLS [30]. 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 - e.g. applications, databases and more - within data centers, allowing for flexible management techniques. This approach is exemplified by NVP [16] (and similarly by FlowN [9]) that exploits it to implement a network management system, within an enterprise data center. The core issue with these solutions is the limited scope in which they can be applied, as the 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.

Overlay Networks. Overlay networking approaches, e.g. VINI [2], 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 visibility of the underlying network environment, not providing support for the aforementioned cross-layer interactions.

The need for Layer 3 network virtualization. Looking at the two available solutions, we identify three limitations: 1) Most virtualization techniques are limited to single domain scopes, e.g. a data center or an access network. 2) When extended to support larger networks, they either need full control of the network environment, or 3) they rely on overlay solutions that are costly due to the generated overhead and lack any access to the underlying network environment. The overall goal is to provide a solution that enables the exchange of information between the virtualized environment, the applications that run on top and the underlying network. This solution should offer service providers the ability to exploit network virtualization to enhance deployed solutions like edge clouds, where applications might benefit from affecting routing decisions based on custom metrics and cross layer optimization. From this analysis, we identify the network layer as the right level to host a Virtual Network design. 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. The next section defines how a VN can be integrated into the network layer.

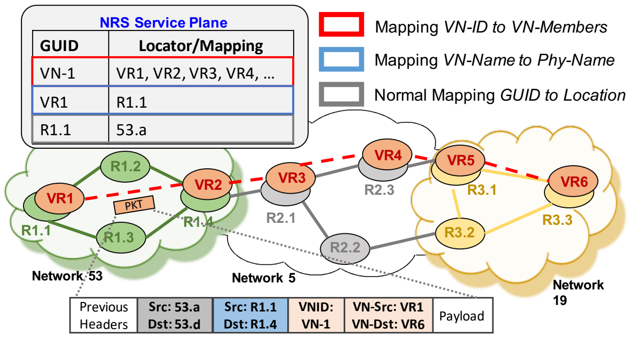


Figure 2: NOVN design

3 NAMED-OBJECT BASED VIRTUALIZATION

Named-objects [5] are a powerful abstraction achieved through the use of a dynamic globally available Name Resolution Service (NRS) for mapping names to routable network entities. The research community has advocated for the separation of names (identities) from addresses [10, 21, 28] for quite some time. This separation has inherent benefits in handling mobility and dynamism for one-to-one communications. The general concept of named-objects can be extended to achieve considerable flexibility in creating a variety of new service abstractions [6]. First, names can be used to represent many different Internet objects; for example, a cell-phone, a person, or a group of devices; the latter concept also applies in the context of network virtualization, as it provides the basis for *NOVN*'s solution of defining participation of network elements to the logical network. In this case, the named-object abstraction can be used to define entire VNs and store the corresponding topology directly into the NRS. The routers' job is then simplified as they can support multiple virtual network policies simply by indexing their routing table to the Virtual Network Identifier (VN-ID) associated with a given network. This makes it possible to operate VNs without the need for any additional overlay protocols, creating the sense of VNs as an integral feature of the network protocol stack. The following sections provide the general concepts of how this process is defined. Moreover, more details information is provided on how *NOVN* addresses the requirements presented in Section 2.1

3.1 NOVN General Design

NOVN addresses the fundamental issues of virtual network management and deployment support through the use of named-objects. Figure 2 lists for clarity the set of core design operations are at the base of the framework. To simplify the discussion, three basic assumptions are considered throughout this section: (1) the availability of a globally accessible NRS capable of storing mappings from *names* to list of *values*; (2) the ability to identify network classes based on a unique identifier (SID); and (3) the flexibility of accessing names and addresses as part of a network header to enable hybrid routing, similar in spirit to the one employed in the MobilityFirst architecture [28].

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: 1) first, a unique identifier is assigned to the VN and a mapping from such name (*VN-ID*) 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. 3) Finally, these identifiers are mapped into physical Network Addresses allowing for normal forwarding operations. Items 1 and 2 above define the higher abstraction level shown in Figure 1 and their mapping into the mid-layer, while item 3 provides the last translation to the bottom layer, that is, the physical infrastructure.

Bootstrap Process & Management. As the topology information is made available at a global scale through the NRS and can be dynamically retrieved from participating resources, the scope of what information is required to share at each layer of the network infrastructure is limited in comparison to other solutions, e.g. [2]. This allows 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. The first one can be handled either in a network-by-network basis or by a centralized authority while the second one is offloaded to the NRS. To this end, the bootstrap process in *NOVN* is then limited to allocating on participating nodes instructions on how to retrieve the VN topology, i.e. the VN unique identifier used to query the NRS, and the information about the physical resources that are required. Similarly, management operations, e.g. migration, of resources can be handled through NRS offloading too, whereas local changes are reflected into the globally accessible service and dynamically resolved at forward time.

Routing & Forwarding. Providing full flexibility for different routing configurations, *NOVN* does not constrain VN users to employ specific routing protocols. 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 chunk reaches one of these routers and a routing decision is taken, the chunk is encapsulated within an external network header that contains information to reach the next VN router (shown in Figure 2). 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.

3.2 An Embedded Virtualization Abstraction

Conventional network virtualization techniques suffer from the fundamental shortcomings of the underlying IP architecture and address structure, limiting their flexibility and increasing deployment complexity. Consider the case of overlay based solutions (e.g.

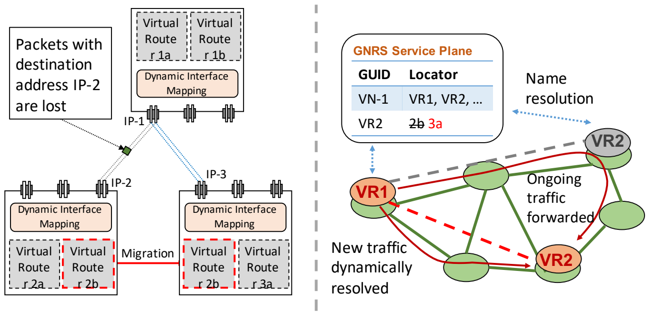
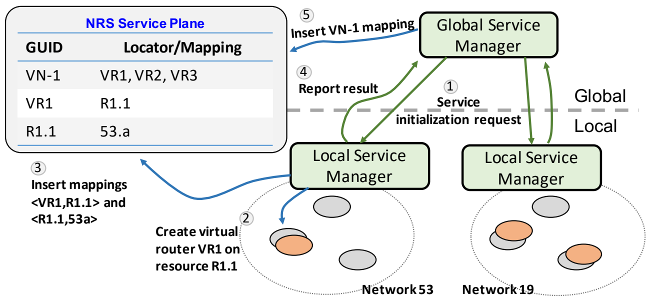


Figure 3: The effect of router migration on overlay deployments (left) and *NOVN* (right).

VINI [2]) where virtual router interfaces are assigned private IP addresses and then mapped to public ones that can be used to tunnel packets across participating resources (Figure 3). Due to the nature of IP addresses, any configuration change due to failure or resource migration requires the tunnel to be reconfigured, the new information to be propagated across all the participating resources, causing the loss of all ongoing traffic. This is due to packets not being able to carry the necessary information to self-correct temporary errors. Approaches to reduce this impact have been explored [37], but require the creation of dedicated control channels to maintain persistent traffic flow.

NOVN solves this issues by creating clean partitions across logical layers, as previously shown in Figure 1. This is obtained by recursively mapping from VN dedicated names, to network elements names and finally to the physical addresses. These layers of abstraction are critical in allowing a separation of management issues. Consider, for example, the case of virtual router migration. In *NOVN*, the process is simplified by limiting the impact of the migration to remapping identifiers between the top two layers. Once the required migration process is defined, the entry mapping the VN element to the network element is re-written to the new location. If in-flight packets are forwarded during the transfer process, name indirection allows for fast recovery without need of end-to-end retransmission, by resolving the delivery location through the NRS. Similarly, if a physical machine needs to be replaced due to failure or an address change is required, a new one can be instantiated and the state transferred.

One could argue that the employment of multiple layers of abstraction can introduce additional overhead due to the resolution costs of crossing the different logical layers through name resolution and due to the additional headers employed. The impact of these is alleviated though by the employment of two separate techniques: 1) While name resolution can become costly if performed for each forwarding decision, the action is not required as for the majority of the time the resources do not change; hence, information can be pre-cached on the participating routers and only once resources are notified of occurring changes they have to update their mappings by querying the NRS. 2) As tag switching and SDN techniques [20] have demonstrated, matching multiple fields in hardware is a feasible task and as software components take over, this becomes an even easier task. An empirical demonstration of



NRS Service Plane

GUID	Locator/Mapping
VN-1	VR1, VR2, VR3
VR1	R1.1
R1.1	53.a

Figure 4: Separation of local and global scale problems through a distributed coordination plane.

the feasibility of the approach will be given as part of the prototype deployment presented in later sections.

3.3 Separating Local and Global Tasks

Managing resources in virtualized environments increases in complexity when extended to multiple domains. This is true for overlay approaches, where resources need to be coordinated and communicated potentially across multiple networks in order to synchronize, and it is mostly untreatable for tag based solutions that are usually optimized for small domains, e.g. a data center or an access network [16]. This is a consequence of the complexity of assigning coherent resources across multiple domains that can be managed by different commercial entities.

NOVN approaches the problem by creating a distinction between the *local* problem of assigning network and computing resources and the *global* problem of providing coordination mechanisms across domains. The NRS and the named-object abstraction are the key elements employed to offer ways for eliminating the complexity as they provide the infrastructure a way to offload the sharing of the virtualized topology and the mapping of the underlying elements. With this, network administrators can then separately focus on deploying techniques for optimizing the management of their infrastructure and the placement of the resources while relying on globally available mappings for coordinating with partnering networks.

Figure 4 outlines the resource allocation process when a hierarchical set of service coordinators is employed. In this example, each network domain exposes an interface that services deploying a multi-network VN can invoke to allocate resources that span across the participating networks. While this example employs the concept of a single service interface per network with a centralized controller for requesting and coordinate resources across networks, the same tools can enable more distributed mechanisms for allocating and deploying virtual networks.

3.4 Network State Exchange

Similar in spirit to previous attempts of providing full control of the deployed routing protocols on top of virtualized networks [2], *NOVN* has been designed to offer routing independent network abstractions. In other words, administrators of virtual networks can independently choose which routing protocols better suit their needs as long as they have ways of learning the underlying network conditions, e.g. virtual links costs. This latter problem could be approached in multiple ways: a) resorting to over the tops approaches where measurement tools are used to extract the information, as done in VINI [2]; b) by allowing routing information sharing across layers, through the use of APIs exposed by the underlying networking logic.

The current *NOVN* design favors the second approach, acknowledging the increasing reliance of software based routing tools that can support APIs used by the virtual layers on top to extract link state information.

4 APPLICATION SPECIFIC ROUTING

Edge clouds are often colocated with the existing network equipment and hence deploy limited computational resources. For this reason, they are solely capable of hosting a limited amount of applications at any point in time, requiring service orchestrators to engage in dynamic traffic management. To approach this problem and support a new wave of traffic management techniques, we extend the core named-object based abstraction layer of the *NOVN* framework to support more integrated routing mechanisms and facilitate the deployment of advanced services through a technique called Application Specific Routing (ASR). ASR defines a mechanism aimed at exploiting a comprehensive set of 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. Consider, for example, 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 and service load at the end points.

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 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. *NOVN* supports the first one by offloading the list of participant locations under a single name into the name resolution service and the second one by allowing custom routing protocols to be deployed on top of any underlying infrastructure and integrating end point APIs to push application state into the VN.

Details follow on how *NOVN* supports edge cloud scenario through the employment of *ASR*.

4.1 Edge Cloud Scenarios

For edge clouds to scale well and deploy easily, it is necessary to develop a robust and self-organizing distributed architecture analogous to the way in which inter-domain protocols in the Internet enable networks to cooperate on routing while retaining some measure of local policy control. Of course, the distributed algorithm design problem for edge clouds is a more difficult one because we are dealing with a mix of computing and networking resources with complex cross-layer interactions and considerable heterogeneity

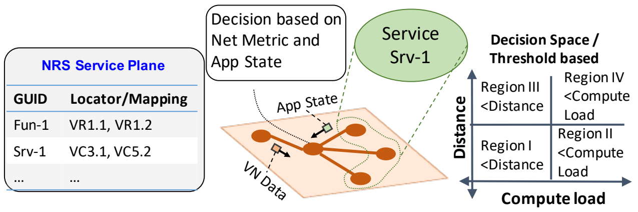


Figure 5: Application Specific Routing as an advanced routing service for the edge cloud use cases

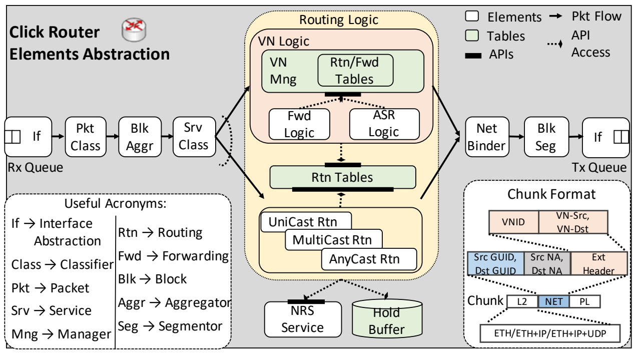


Figure 6: Click router elements graph for data plane flow

in both networking and computing metrics across the region of deployment.

ASR supports edge cloud solutions through the support of advanced cross-layer routing mechanisms. Consider for example the scenario depicted in Figure 5, where a collection of servers offer a service to its clients. *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 (Figure 5). At branching locations, routers can then take informed decisions. For example, Figure 5 shows a decision space scenario where given threasholds define different states that can influence how routing decision. While the effectivness of the ASR approach has been proposed in our previous work in the context of cyber physical systems [22], coupling *NOVN* with ASR can support a low latency and scalable solution for any service that would benefit of the locality of edge clouds.

5 PROTOTYPE AND EXPERIMENTS

In order to understand the achievable performance and feasibility of the proposed *NOVN* and *ASR* designs, a fully working prototype of the framework has been implemented. The *NOVN* prototype uses as its foundation the MobilityFirst (MF) future Internet architecture [28] prototype [7]. The MF architecture is an example of how the named-object abstraction could be integrated into an Internet network design and for this reason provides the perfect environment to natively deploy the features at the base of *NOVN*. At the core of the architecture is a new name-based service layer which serves as the "narrow waist" of the protocol stack. The name-based service layer uses flat Globally Unique Identifiers (GUIDs) of 160 bits to identify all principals or network-attached objects. Names are resolved through a Global Name Resolution Service (GNRS) that provides APIs to insert and query for <*key,value*> mappings and support hybrid routing schemes [23] that exploit availability of both names and addresses in the network header for dynamic resolution of destination locations. A Service Identifier (SID) flag placed in network header allows network components to be aware of different service types in order to apply different forwarding modes.

The main components of the architecture prototype are three: a Java based GNRS that uses DMap's [36] DHT based implementation to distribute mapping entries, a software router implementing MF's hybrid name/address routing logic and a host guid based API and network stack [6] to run applications on the architecture; while the complexity of the individual components of the prototype is not irrelevant, due to space constraints this paper solely focus on the components that have been extended to support *NOVN*'s design, referring to previous work [7]. The open access code repository and code wiki are also fully available for more details [1].

5.1 Prototype Components

Routers. The software router is implemented as a set of forwarding elements and table objects within the Click modular router [15] run at user-level. As a baseline, the router implements dynamic-binding using GNRS, hop-by-hop reliable transport using a HOP [19] inspired protocol (by aggregation and segmentation of large chunks of data), and storage-aware routing [23]. It integrates a large storage, via an in-memory *hold buffer*, to temporarily hold data blocks for destination endpoints during short-lived disconnections or poor access connections. A particular instance of this system, implements what we call an MF access router, a router providing access connectivity to clients.

The base router has been extended to introduce the *NOVN* logic (Figure 6). Multiplexing across different delivery services is handled via the Service ID (SID) tag available in the MF routing header (*Srv Class*). Encapsulation of the *NOVN* required headers has been implemented exploiting extension fields in the MF network layer. When traversing a non-VN enabled router, the SID is not recognized and the data is forwarded based on normal unicast rules. Once packets enter the VN logic layer, the router checks whether a) the packet is intended for itself (destination GUID) and b) if the VN belongs to the ones currently active; the simple field base matching exploits VN native concepts as explained in section 3.2 allowing for a performant decision logic, as shown in the results of the next Section. VN tables (Routing/Forwarding/ASR) are stored and quickly retrieved via a Hash Map, guaranteeing high performance; when invoked, the routing logic (and if deployed, the ASR one) can access the information and take fast decisions.

The control plane (not shown in the picture) is handled in similar fashion: the current design implements a Link State like Protocol (LSP), to exchange routing information between routing instances; routers periodically generate and distribute the aggregated cost view of each virtualized link to neighbors that, following the logic of the protocol, store and forward the information. Path costs are

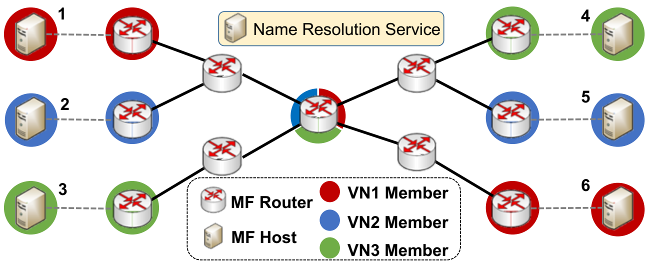


Figure 7: Network topology used for benchmarks

Size	RTT without NOVN	RTT with NOVN
64 B	7.6 ms	8.8 ms
1 MB	128.1 ms	128.1 ms
	Throughput without NOVN	**Throughput with NOVN**
64 B	14 mbps	11 mbps
1 MB	916 mbps	903 mbps

Table 1: Latency and throughput NOVN Benchmarks

extracted from the underlying unicast routing tables (*Rtn Tables*) via APIs. Initialization of the logic for a given VN can be done via two different methods: either statically within the click configuration files using as inject point or based on a managing protocol exposed via the Click software control interface.

Finally, the routers have been enabled with interchangeable *Interface* classes that can adapt to different networking environments, supporting different deployment scenarios; these include: a) native support of the MF protocols on top of a L2 network and overlay support both on top of b) barebone IP network or c) a full overlay solution on top of UDP.

Clients. In similar fashion, the baseline client network stack and API [6] have been extended to support *NOVN* operations including: a) exposure of the required API options during socket initialization (i.e. open) to b) instantiate resources in the network stack and c) encapsulation of messages as required by the protocol.

5.2 Benchmarks

A combination of routers and clients have been deployed on the ORBIT testbed [29]. On the testbed, all nodes are interconnected via 1 Gbit ethernet switches, creating a single L2 network. Selecting 19 nodes, different networks have been deployed for the different use cases analyzed. As the testbed provides a single L2 network, a logical split has been implemented within the click routers to enforce the topology. We present two core results: 1a) a set of micro-benchmark experiments aimed at demonstrating the baseline computation overhead of our VN implementation against the baseline MF prototype and 1b) an analysis of how different VNs can co-exist on the deployed network; then 2) an ASR edge cloud use case deployment scenario has been analyzed. In this paper, no direct comparison with IP networks is provided, but, as mentioned, the prototype could potentially be deployed as an overlay network on top of the current Internet architecture.

In order to understand the basic overhead introduced by running the virtual network logic on top of the baseline prototype, three sets of benchmarks are performed: first, a latency evaluation using a *ping*-like application that collects RTTs for a small (64B) and a large (1MB) chunks size; second, using a port of *iperf* that uses the new API and stack to transmit data, achievable bandwidth is estimated. For both scenarios the network shown in Figure 7 is used, but traffic generation is limited to VN-2 (blue color). Third, as a big advantage inherent to the *NOVN* design is the possibility of performing multiplexing across different VNs by natively switching traffic based on a single header field, i.e. the *VN-ID*, the overhead and functionality of this switching in the prototype has been evaluated. *Latency & Throughput:* Total values reported in Table 1 account for the sum of three time components: 1) the processing time of the software router (including potentially the VN logic); 2) the queries to the NRS (2ms RTT from the routers to the NRS with query results cached on the routers for 30 seconds); and 3) the HOP like protocol which requires the transmission of initial and final control packets for each chunk to provide a reliable transmission on a hop-by-hop basis. For this experiment, RTTs for the smaller chunk size do suffer some small increase in the *NOVN* case due to the overhead generated by the processing of the added logic and the additional queries to the NRS (to resolve the higher layer mappings). The effect of the NRS queries is limited though, as they are averaged over the number of total collected samples (1000, one every second), even considering that a 30s cache is quite conservative, especially for VN like scenarios where changes are unlikely to happen in the order of seconds. The bigger size is less impacted by the additional overhead. The performance impact of *NOVN*'s overhead on the achievable throughput is also minimally noticeable, but with increasing chunk size the effect is proportionally minimized. For this metric, the impact of the queries to the NRS is a lesser factor (at 1MB, ~113 chunks per second are transmitted and only one time every 30s or ~3400 chunks the NRS is queried). The decrease in throughput has then to be attributed to the additional header and processing overhead caused by the VN logic. Even though these do factor for a decrease in performance, this is small enough that the evaluated scenario does not causes concern for the effectiveness of the design. *Multi VN Coexistence.* To test the overhead and functionality of the VN switching mechanisms in the prototype, three VNs have been deployed on the network shown in Figure 7. Each traffic source (nodes on the left side), generates traffic at 100mbps. Figure 8 shows the results after running a five minutes experiment. While initial competition on the wire, causes some overshooting of the goal throughput, the traffic stabilizes shortly after and it is maintained until the experiment is completed (at around 300s). The overshooting is introduced by the chunk base nature of the protocols implemented, where a sudden arrival of large chunks (1MB) requires time to adjust.

5.3 ASR Use Case.

To exemplify the implementation of the ASR concept, a closed-loop (round-trip) application has been deployed on the network pictured in Figure 9, where clients send requests of 10KB each in size to a set of two servers representing a cloud service. ASR is deployed to consider in its forwarding decisions both network metrics used in

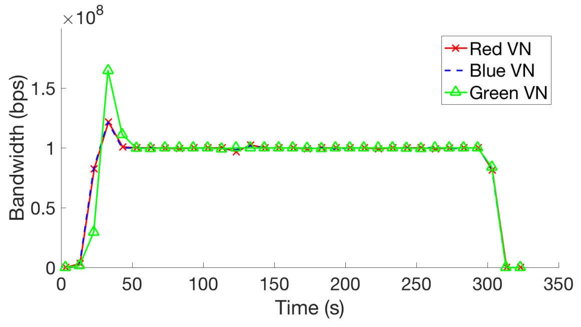


Figure 8: Multiplexing NOVN Benchmark

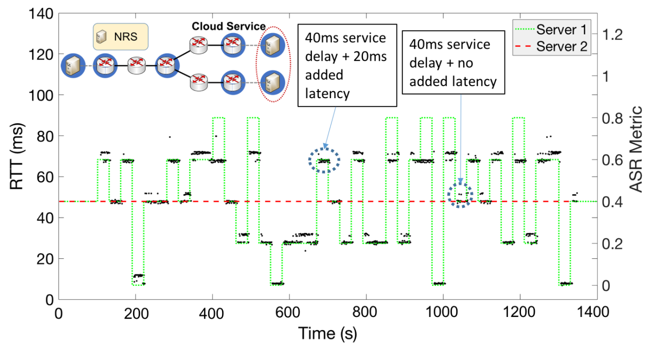


Figure 9: ASR edge cloud use case example

the normal routing scheme (latency and delay) and the servers load. Cloud servers loads are emulated by adding emulated delays before sending responses of 10KB back to the client. Server-1 has dynamic load chosen uniformly every 30 seconds from the set of values 0, 0.2, 0.4, 0.6, 0.8, representing linearly increasing delays of 0, 20, 40, 60, 80 ms. Server-2 is statically configured to always select parameter 0.4. A 20 ms extra RTT has been added in the path to the bottom server by using *tc* to emulate different path distance between the servers. Servers announce their load via the ASR protocol every 2 seconds. Figure 9 shows the performance obtained, representing the taken decisions by the ASR logic; at the bifurcation, requests are forwarded based on a simple threshold logic, where potential destinations are divided into a decision space in which different regions have higher priority: if there are servers with load lower than 0.5, choose the one with the best path; otherwise simply choose the best path. This guarantees for the experiment setup that all requests are sent to a router with load lower than 0.5 capping response time to ~70ms.

This setup has then be extended to represent a more realistic scenario as shown in Figure 10. In this case, three clients are deployed, connecting to three networks each equipped with a local service instance. Crossing border routers introduce a 5ms delay each way, replicating the cost of traversing accross domains. The server loads are dynamic with the same parameters. Each case has been run for one hour and collected results show how the combination of *NOVN* and ASR impact the service response time. Figure 11 shows the obtained results. The following should be observed: 1) up to ~50ms,

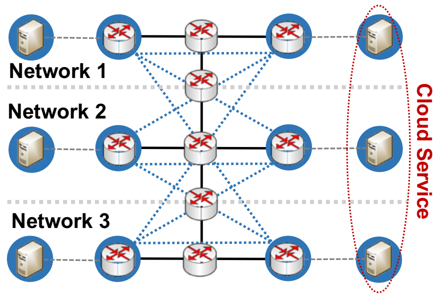


Figure 10: Network topology used for edge cloud deployment

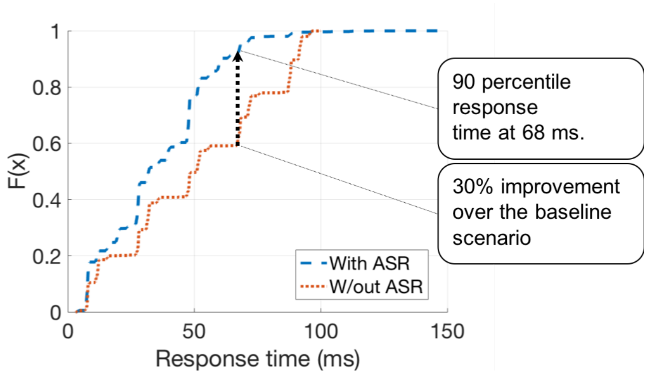


Figure 11: Response time for edge cloud deployment

the difference between the two lines should be recollected to the local servers' load variations over time (i.e. if the load is below 50%, the local server is chosen) and should converge over a longer time; 2) the ASR impact is very noticeable above such threshold, where 90% of requests are serviced in less than 68ms, a more than 30% improvement from the baseline case (where the local server is always selected).

6 DISCUSSION AND RELATED WORK

The Name Resolution Service. At the crux of the *NOVN* framework is the named-object abstraction enabled by a globally accessible Name Resolution Service (NRS), which is used by objects to both announce their latest location/address and lookup end points they wish to communicate with. The performance of the NRS becomes critical for the success of the presented solution. While not at the center of this paper, it is important to report that multiple projects have demonstrated how different incarnations of of such service are possible [33, 36], all achieving low resolution latency goals of less than 100ms on average for lookup operations. Moreover, additional studies are in progress aiming to further reduce response time exploiting concepts such as caching and locality [12]. Commercial [10] and experimental [33] versions of such services are currently running and are available for use.

Inter-Domain Peering Agreements. Inter-domain connections might require additional coordination across parties involved if no overlay solution is implemented. For this, it is arguable that the increasing reliance of ISPs on point to point agreements via Remote Peering [8] and private interconnections over IXP locations via VLANs would well serve this type of architecture. Both techniques rely on the use of tag based forwarding, e.g. long distance MPLS for the first, to interconnect networks, providing a suitable environment to map higher level VNs defined in *NOVN* to these channels.

Related work. *NOVN* takes inspiration from within two broad categories of works: 1) virtual network designs and management techniques and 2) software based solutions to enhance services on networks. Most recent VN designs in general span from overlay solutions [2, 14] to lower layer integrations using tag switching [9, 16]. Software based management is often used to either provide the required tools [34] (in which case, it is often called network slicing). *NOVN* differs from all these works by offering a native network-layer solution based on separating names identifying VN resources from the underlying infrastructure. No other work has looked at this type of generalization, providing capabilities that can extend across multiple domains.

ASR takes inspiration from the broad variety of software enhanced solutions aimed at broadening capabilities and allowing greater control and interaction to application and services populating networks. SDN [20] and its extensions [11] have provided the greatest amount of contributions to this research area, but have been limited their scope to single domains. Before the SDN trend, active networks [26] had been also proposed as an extreme solution to the problem, allowing packets to carry instructions interpreted by the network fabric to forward data in the network. Multi-domain approaches have mostly focused on single specific issues, such as anycast delivery or path selection to distributed services [35, 39]. Internet standardization organizations have also taken inspiration from previous work on ASR to introduce overlay approaches for custom routing [18]. ASR in *NOVN* differs from previous work by providing a distributed and integrated solution for deploying both advanced network control and allowing applications to influence network layer decisions.

Lastly, *NOVN*, through the employed named-object abstraction, belongs to the categories of Information Centric Networking [13, 17, 24, 28] and name separation [10, 21] works. Two recent works [25, 31] have provided solutions to support virtualized networks in ICN. Saldara et al. [31] presented vICN, a solution solely intended to support virtualized abstractions of NDN/CCN [13] networks. In similar fashion, Partridge et al. [25] proposed a solution to support Virtual Private Networks. No ICN research effort has looked at how to generalize virtual network support at Layer 3.

7 CONCLUSIONS

This paper presents *NOVN*, a novel network virtualization architecture aimed at providing a clean and logically simple solution for deploying virtual networks. Exploiting the named-object abstraction, together with Application Specific Routing, *NOVN* provides a solution that offers the logical simplicity of L2 network virtualization while achieving a high degree of flexibility in creating customized topologies and routing of traffic in an application-aware manner.

Results based on a working prototype deployed on the ORBIT testbed demonstrate that the new framework provides an efficient realization for defining and managing virtual networks without compromising performance or incurring excessive control overhead. Future work will focus on comparing the presented solution with overlay networks as well as running the *NOVN* prototype framework on top of the legacy IP networks and on evaluating ASR techniques applied to large scale edge cloud scenarios.

REFERENCES

[1] [n. d.]. MobilityFirst Wiki. http://mobilityfirst.orbit-lab.org/.

[2] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. 2006. In VINI veritas: realistic and controlled network experimentation. *ACM SIGCOMM Computer Communication Review* 36, 4 (2006), 3–14.

[3] E. Bell, A. Smith, P. Langille, A. Rijhsinghani, and K. McCloghrie. 1999. *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions.* RFC 2674. https://www.ietf.org/rfc/rfc2674.txt

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing.* ACM, 13–16.

[5] Francesco Bronzino, Shreyasee Mukherjee, and Dipankar Raychaudhuri. 2017. 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, 37–42.

[6] Francesco Bronzino, Kiran Nagaraja, Ivan Seskar, and Dipankar Raychaudhuri. 2013. Network service abstractions for a mobility-centric future internet architecture. In *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture.* ACM, 5–10.

[7] Francesco Bronzino, Dipankar Raychaudhuri, and Ivan Seskar. 2015. Experiences with testbed evaluation of the mobilityfirst future internet architecture. In *Networks and Communications (EuCNC), 2015 European Conference on.* IEEE, 507–511.

[8] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. 2014. Remote peering: More peering without internet flattening. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies.* ACM, 185–198.

[9] Dmitry Drutskoy, Eric Keller, and Jennifer Rexford. 2013. Scalable network virtualization in software-defined networks. *IEEE Internet Computing* 17, 2 (2013), 20–27.

[10] Dino Farinacci, Darrel Lewis, David Meyer, and Vince Fuller. 2013. *The locator/ID separation protocol (LISP).* RFC 6830. https://tools.ietf.org/html/rfc6830

[11] Andrew D Ferguson, Arjun Guha, Chen Liang, Rodrigo Fonseca, and Shriram Krishnamurthi. 2013. Participatory networking: An API for application control of SDNs. In *ACM SIGCOMM computer communication review*, Vol. 43. ACM, 327–338.

[12] Yi Hu, Roy D Yates, and Dipankar Raychaudhuri. 2015. *A Hierarchically Aggregated In-Network Global Name Resolution Service for the Mobile Internet.* Technical Report. WINLAB TR 442.

[13] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. 2009. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies.* ACM, 1–12.

[14] Xuxian Jiang and Dongyan Xu. 2005. Violin: Virtual internetworking on overlay infrastructure. *Parallel and Distributed Processing and Applications* (2005), 937–946.

[15] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. 2000. The Click modular router. *ACM Transactions on Computer Systems (TOCS)* 18, 3 (2000), 263–297.

[16] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan Jackson, et al. 2014. Network virtualization in multi-tenant datacenters. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14).* 203–216.

[17] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. 2007. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 181–192.

[18] Michael Kowal, Dino Farinacci, and Parantap Lahiri. 2018. *LISP Traffic Engineering Use-Cases.* Technical Report. https://tools.ietf.org/html/draft-ietf-lisp-te-02

[19] Ming Li, Devesh Agrawal, Deepak Ganesan, Arun Venkataramani, and Himanshu Agrawal. 2009. Block-switched Networks: A New Paradigm for Wireless Transport.. In *NSDI*, Vol. 9. 423–436.

[20] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow:

enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.

[21] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. 2008. *Host Identity Protocol.* RFC 5201. https://tools.ietf.org/html/rfc5201

[22] Kiyohide Nakauchi, Francesco Bronzino, Yozo Shoji, Ivan Seskar, and Dipankar Raychaudhuri. 2016. vMCN: virtual mobile cloud network for realizing scalable, real-time cyber physical systems. In *Proceedings of the 4th Workshop on Distributed Cloud Computing.* ACM, 6.

[23] Samuel C Nelson, Gautam Bhanage, and Dipankar Raychaudhuri. 2011. GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet. In *Proceedings of the sixth international workshop on MobiArch.* ACM, 19–24.

[24] Jianli Pan, Subharthi Paul, Raj Jain, and Mic Bowman. 2008. 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, 1–6.

[25] Craig Partridge, Samuel Nelson, and Derrick Kong. 2017. Realizing a virtual private network using named data networking. In *Proceedings of the 4th ACM Conference on Information-Centric Networking.* ACM, 156–162.

[26] Konstantinos Psounis. 1999. Active networks: Applications, security, safety, and architectures. *IEEE Communications Surveys* 2, 1 (1999), 2–16.

[27] KK Ramakrishnan, Prashant Shenoy, and Jacobus Van der Merwe. 2007. Live data center migration across WANs: a robust cooperative context aware approach. In *Proceedings of the 2007 SIGCOMM workshop on Internet network management.* ACM, 262–267.

[28] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. 2012. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review* 16, 3 (2012), 2–13.

[29] Dipankar Raychaudhuri, Ivan Seskar, Max Ott, Sachin Ganu, Kishore Ramachandran, Haris Kremo, Robert Siracusa, Hang Liu, and Manpreet Singh. 2005. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *Wireless Communications and Networking Conference 2005*, Vol. 3. IEEE, 1664–1669.

[30] E. Rosen, A. Viswanathan, and R. Callon. 2001. *Multiprotocol Label Switching Architecture.* RFC 3031. https://tools.ietf.org/html/rfc3031

[31] Mauro Sardara, Luca Muscariello, Jordan Augé, Marcel Enguehard, Alberto Compagno, and Giovanna Carofiglio. 2017. Virtualized ICN (vICN): towards a unified network virtualization framework for ICN experimentation. In *Proceedings of the 4th ACM Conference on Information-Centric Networking.* ACM, 109–115.

[32] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[33] Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, Arun Venkataramani, David Westbrook, and Aditya Yadav. 2014. A global name service for a highly mobile internetwork. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 247–258.

[34] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru M Parulkar. 2010. Can the production network be the testbed?. In *OSDI*, Vol. 10. 1–6.

[35] Vytautas Valancius et al. 2010. Wide-Area Route Control for Distributed Services.. In *USENIX.*

[36] Tam Vu, Akash Baid, Yanyong Zhang, Thu D Nguyen, Junichiro Fukuyama, Richard P Martin, and Dipankar Raychaudhuri. 2012. 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, 698–707.

[37] Yi Wang, Eric Keller, Brian Biskeborn, Jacobus van der Merwe, and Jennifer Rexford. 2008. Virtual routers on the move: live router migration as a network-management primitive. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 231–242.

[38] Timothy Wood, KK Ramakrishnan, Prashant Shenoy, and Jacobus Van der Merwe. 2011. CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In *ACM Sigplan Notices*, Vol. 46. ACM, 121–132.

[39] Xiongqi Wu and James Griffioen. 2014. Supporting application-based route selection. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on.* IEEE, 1–8.