NOVN: A named-object based virtual network architecture to support advanced mobile edge computing services

Francesco Bronzino, Sumit Maheshwari, Ivan Seskar, Dipankar Raychaudhuri

NOVN: A Named-Object Based Virtual Network Architecture to Support Advanced Mobile Edge Computing Services

Francesco Bronzino[+*], Sumit Maheshwari[#§*], Ivan Seskar[#], Dipankar Raychaudhuri[#]

[+] *Université Savoie Mont Blanc*

[#] *WINLAB, Rutgers University*

Abstract

Achieving advanced Mobile Edge Computing (MEC) services such as dynamic resource assignment and slicing, maintaining Quality of Service (QoS), and enabling heterogeneous virtual functions are some of the technical challenges associated with edge-cloud enhanced 5G architectures now under consideration. This paper proposes a named-object based virtual network (NOVN) architecture to support low-latency applications in the MEC. Software router implementation running on the ORBIT testbed validates the named-object approach, showing low VN processing and control overhead, and making it possible to achieve low latency. A latency performance improvement of 30% is achieved as compared to the baseline implementation without NOVN. The results also validate feasibility of using the advanced MEC services for an example latency constrained edge cloud scenario.

Keywords: Virtual Networks; Name-object Networking; Mobile Edge Computing; Network Slicing;

1. Introduction

Mobile Edge Computing (MEC) is envisioned to be a core component in future cellular architectures, 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, virtual or mixed reality (AR/VR/MR) [38, 4, 44]. MECs exploit resource locality and have been embraced by infrastructure and service providers for their network evolution. In particular, providers are increasingly aiming to distribute their service points of presence (i.e. processing and storage) in order to 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 [46, 31, 13], or fog computing [7], that can better scale and provide low delay services to real-time applications.

Edge clouds distributed at the periphery of the network represent a conceptually simple and scalable solution for delivering computing services to mobile users. Moreover, because of the lower network delay in reaching cloud resources, MEC offers the potential to meet strict service requirements (e.g. low latency). These advantages come at the cost of 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. While existing solutions such as network slicing [55] and service chaining [40] provide ways to interconnect distributed, heterogeneous resources, end-to-end quality remains a concern as existing large-scale in-network measurement techniques do not provide sufficient insights required to achieve cross-layer optimizations such as seamless service access via customized routing. A second key challenge arises from the heterogeneity of computing resources and the limited amount of computational power they are equipped with. In contrast to the previous data center driven cloud model, edge clouds are often co-located with existing network equipment and deploy limited computational

[§]Corresponding author. Email address: sumitm@winlab.rutgers.edu.

[*]co-first author

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. User mobility further exacerbates this issue, as link or node failures and network congestion induce additional challenges in the edge clouds inhibiting seamless service access.

To support the large-scale, distributed, and localized nature of edge cloud a general technology and architectural solution for edge clouds will thus require: (a) Control as well management plane protocols to provision these heterogeneous resources in real-time; (b) Distributed or centralized resource assignment strategies, for traffic load balancing, orchestration of computing functions and related network routing of data; (c) Mobility management techniques such as dynamic network slicing, and (d) Low-overhead mechanisms to reach the heterogeneous set of distributed resources in real-time.

This paper designs, integrates, and evaluates edge cloud components aimed at fulfilling the requirements of advanced services over the MEC architecture. First, we design the Named-Object Virtual Network architecture (NOVN), a Layer 3 virtual network solution. The named-object based networking abstraction has been introduced in the literature [42] as a way to support seamless mobility in the network. By dynamically resolving their names to the network addresses, the named-object abstraction provides a way support services such as distributed caching, faster multihoming, efficient multicast, and delay-tolerant content distribution. We extend this general abstraction to design a virtual network solution that can provide the control mechanisms at Layer 3 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 Application Specific Routing (ASR), supports routing service requests based on cross-layer information extracted from network and application. Through this technique network operators can achieve dynamic resource management using in-network resource slicing and devising mechanisms for the quality of service (QoS) control. Finally, we develop a working implementation of NOVN, ASR, and advanced service scenarios using the Click [24] modular router and the MobilityFirst network architecture software prototype [11]. As part of this effort, we present experimental results obtained to validate the feasibility of this architecture and demonstrate significant latency improvements for real-time applications.

The rest of this paper is structured as follows. Section 2 presents edge cloud requirements to support the advanced services such as cross domain connectivity, dynamic re-routing and cross-layer network support, and align them with the key MEC architectural components. Section 3 introduces the core design of *NOVN*; starting from the definition of named-objects, the high level design choices taken are described. The scalability and consistency challenges of name resolution server (NRS) are discussed in Section 4. Section 5 introduces how *NOVN*, integrated with techniques such as application specific routing and network slicing could be exploited to support advanced network services. To support the proposed design, in Section 6 a comprehensive set of experiments based on a working prototype deployed on the ORBIT testbed [43] is presented. Performance evaluation of the proposed NOVN architecture and the related scenarios including comparison with the overlay based solutions are described in the Section 7. Finally, in Section 8 a further discussion on the design choices and a comparison to related work is provided. Section 9 concludes the paper.

2. Edge Cloud Requirements

In this section we discuss the requirements imposed by edge clouds on developing an architecture involving virtual network and advanced MEC services to interconnect and manage distributed resources. Starting from a review of existing virtualization techniques, we discuss the need for the introduction of Layer 3 network virtualization.

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 such as network or service providers.

Dynamic Re-Routing. Due to the nature of IP addresses, any configuration change caused by failure or resource migration requires to reconfigure 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 [51, 2], 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. On the contrary, 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 [30].

Seamless Service Integration. Edge clouds promise to support tighter close loop low latency applications which require a seamless integration of services with the network entities whose performance can be monitored, reported and enhanced. The key requirements therefore is to design mechanisms which can blend service state parameters into the network to create a fully virtualized end-to-end QoS-enabled system.

The mapping of these requirements to the corresponding architectural component of MEC is further discussed as follows.

2.1. Enabling QoS using Virtual Networks:

Virtual Networks (VNs) are now days the main technology used to connect resources across the Internet. VNs support the illusion of a customized network with a user-specified topology, offering the ability to share the the underlying infrastructure through network slicing. Further, depending on the solution employed, VNs can offer the ability secure the communication channel between member of the virtualized network as well as to achieve deterministic QoS characteristics matched to application requirements [22].

Depending on the purpose, different techniques have been applied at different layers of the networking stack in order to realize virtual networks. Existing VN 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 [6] and MPLS [45]. 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 [25] (and similarly by FlowN [15]) that exploits it to implement a network management system, within an enterprise data center. Thanks to the built-in mechanisms available within the network domain fabric –e.g. Ethernet's class of service (CoS)– L2 VNs offer the ability to implement fine grained QoS scheduling of the traversing traffic. 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 [5, 14] or point to point connectivity between remote cloud locations [41, 52], 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

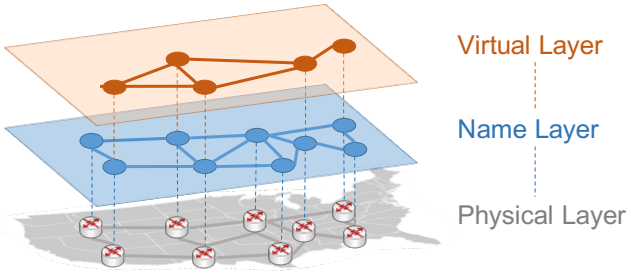


Figure 1: *NOVN* layers of abstraction.

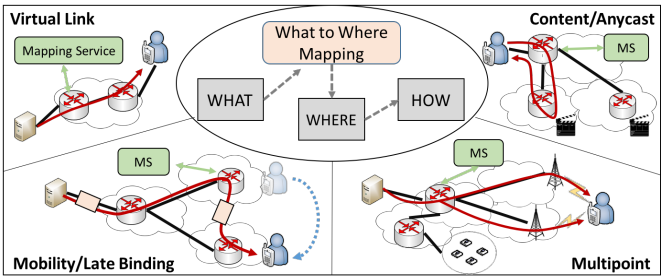


Figure 2: The *named-object* abstraction applied to core use cases.

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 their utility in scenarios that might benefit from custom metrics and deeper cross layer optimization [49, 53].

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; further, when extended to support larger networks, they either 2) need full control of the network environment, or 3) rely on overlay solutions that are expensive 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.

3. Named-Object Based L3 Network Virtualization

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* [8], a virtual network solution that exploits the concept of named-objects [9] introduced in the MobilityFirst future Internet architecture [42] 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.

For more than a decade, the research community has advocated the separation of names (identities) from network addresses [33, 16, 27, 19, 42]. Named-objects [9] are a powerful abstraction achieved through the use of a dynamic globally available Name Resolution Service (NRS) for mapping names to routable network entities. 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 [10] as shown in Figure 2. 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

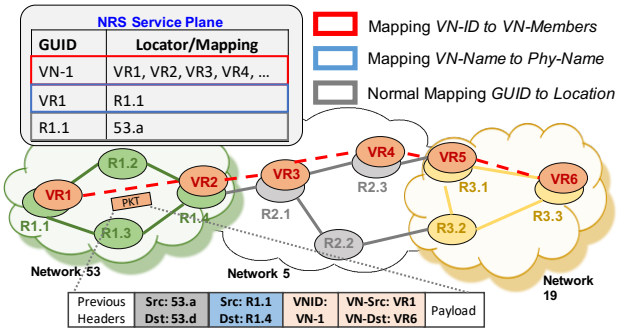


Figure 3: NOVN design

can support multiple virtual network policies simply by indexing their routing table to the Virtual Network Identifier (VNID) 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 named-object based virtualization has many advantages. Traditional QoS control mechanisms require complex packet sniffing and processing to manage network resources for achieving traffic prioritization and resource reservations. The name-based network virtualization technique described here simplifies the QoS control by mapping each virtual network to a unique identifier (Virtual Network Identifier – VNID) encapsulated in the packet header. Striping VNID from the packet header and querying a Name Resolution Server (NRS) for the VNID's allowed traffic capacity provides a L3 in-network support for the scanarios such as shaping network traffic, limiting bandwidth for a VN, and ensuring QoS using the ASR metrics. Furthermore, with a network entity (NE) such as router allowed to be a part of multiple name-based virtual networks, techniques such as network slicing can be achieved by the statistical multiplexing of the resources while pushing the VNID to the resource mapping to the NRS during VN instantiation or dynamically, and run-time retrieval of the same by the NE involved in the VN. The prime advantage of this approach is that the resource provisioning and their chaining need not be done in advance. At each NE hop, both the resource metric and the next hop information is obtained from the NRS which thereby inherently handles scenarios such as node failure, link failure and traffic congestion without affecting the ongoing virtual network connection.

3.1. NOVN General Design

NOVN addresses the fundamental issues of virtual network management and deployment support through the use of named-objects. Figure 3 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 [42].

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 (*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. 3) Finally, these identifiers are mapped into physical Network Addresses (NA) 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

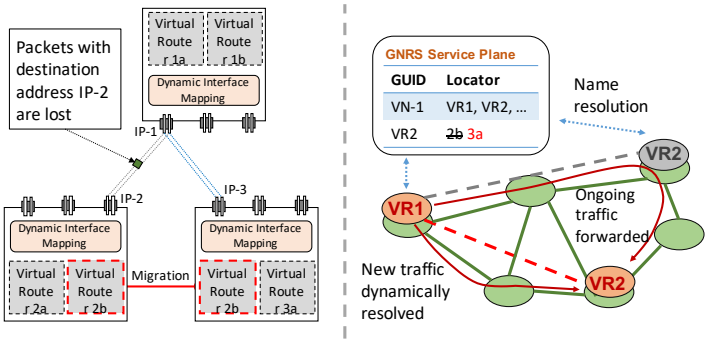


Figure 4: The effect of router migration on overlay deployments (left) and *NOVN* (right).

is required to share at each layer of the network infrastructure is limited in comparison to other solutions, e.g. [5]. 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 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.

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. VINI [5]) 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 4). 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 [51], 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

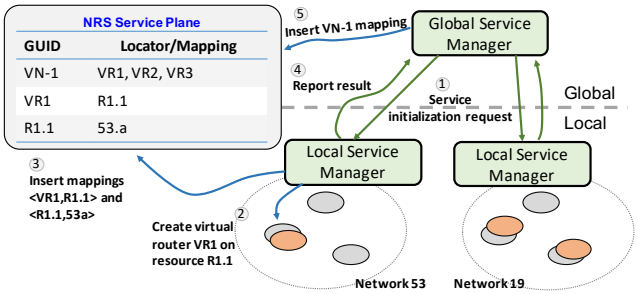


Figure 5: Separation of local and global scale problems through a distributed coordination plane.

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 [32] 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 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 [25]. 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 5 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 [5], *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 [5]; 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. Name Resolution Service Impact on the Architecture Scalability

The named-object abstraction of *NOVN* relies upon the use of a Name Resolution Service (NRS). Therefore, the performance of the NRS becomes critical to achieve consistent performance. Multiple previous projects have demonstrated how different NRS designs [50, 47] achieve low resolution latency goals of less than 100ms on average for lookup operations. Moreover, additional studies demonstrate how to further reduce response time exploiting concepts such as caching and locality [20]. Commercial [16] and experimental [47] versions of such services are currently running and are available for use. This section provides an overview of the different implementation approaches and details how to handle consistency and scalability issues while relying on the NRS to deploy the NOVN architecture.

4.1. NRS Implementations

NRS designs can be classified as either hierarchical or flat based on their naming structure. For hierarchical namespaces, commercially proven implementations such as DNS are available. Unfortunately, it has been demonstrated that DNS is not suitable as an NRS implementation in a highly mobile environment due to its static placement strategies inherently, e.g. time-to-live (TTL) based caching, limiting its effectiveness upon end-host mobility. This work relies on the use of a flat namespace for which there are several NRS implementations available in the literature. We select two designs as candidate implementations for NOVN.

Auspice [47]. Auspice's logic uses a demand-aware replica placement engine to distribute GUID to NA records across available caches to provide low lookup latency, low update cost, and high availability by carefully choosing the number and locations of required replicas for each GUID as per the lookup and update request rates, the existing replicas for a GUID, and aggregate load at a replica. This geo-distributed engine is implemented as a logically centralized authority which tracks query demands using a recursively mapped key-value store. In Auspice, a GUID belongs to a number of replica-controllers (fixed) and active replicas (variable). The replica-controllers maintain information about active replicas such as their number and locations whereas the actives replicas maintain a GUID record and process a request. The placement algorithm is computed locally at each replica using lookup to update ratio of a GUID thus limiting the update cost. The replica location is chosen such that the lookup latency is minimal by selecting some replicas closer to the higher demand zones while others placed randomly for load balancing.

DMAP [50] / GMAP [21]. In DMAP, name mappings to network addresses are distributed among participating Autonomous Systems (ASes) while also choosing a deputy As which has minimum IP distance to the current hash value of an IP address. DMAP routers apply K consistent hash functions (where K is the number of replicas desired) to map names to the gateway routers wherein they are are stored. GMAP builds over DMAP by organizing the name to NA mappings hierarchically in three levels – local, regional, and global – to exploit spatial locality. Furthermore, the server lookups are load balanced using a concept of probabilistic caching, thus improving scalability over the baseline solution.

4.2. NRS Challenges

The implementation of a large scale database such as a Name Resolution Service creates challenges of information freshness as well as lookup delay that might compromise the requirements of the NOVN architecture. In the following section we describe how both architectures could be safely employed to deploy NOVN.

Consistency. Inconsistency may arise when a query reaches a cache that does not hold an up to date name/address mapping due to host mobility and late update, or incorrect prefix cache in a BGP table, thus incurring additional query response delay.

In Auspice [47], the consistency issue is handled using an explicit coordination mechanism between the consensus engines of the replica-controllers and active replicas, where each NRS node propagates information to a set of replica servers. In DMAP [50], consistency is quantified as the probability of BGP churn by varying the percentage of prefixes that are newly announced or withdrawn from 0 to 10%. If a GUID is not found

at an AS, it replies with a "GUID missing" message and the querying node contacts another replica. The fragmentation of IP address space may lead to an unannounced IP as a hash causing the IP hole problem. To maintain consistency, the withdrawing AS sends a GUID insert message to the deputy AS while deleting its own entry. Later, the subsequent queries hit a IP hole and thus the deputy AS is queried to obtain the latest mapping. In case a query is not found at an AS, the querying AS sends a one-time migration message to the deputy to self-assign a mapping thus removing an inconsistency.

GMAP [21] further enhances DMAP's consistency mechanisms by using a sequential scheme which piggybacks the server availability updates in the query replies along the request path. It is shown that for up to 5 replicas, there is only a 5% failure rate which shifts the median from 40.5ms to 41.3ms which is acceptable for consistency in NOVN as there are practically a very few origin changes for an AS prefix according to [39, 29].

Scalability. An increase in number of replicas improves NRS scalability but creates a consistency problem. In Auspice, the dynamic nature of its placement algorithm maintains a balance between the cost and the performance. At lower loads, the lookup latency is minimized by selecting maximum number of replicas whereas at higher loads, only the popular GUIDs are replicated at multiple locations thus keeping cost under control without sacrificing performance a lot, making the solution scalable.

In DMAP the balance between consistency and scalability is provided by (a) using a single overlay-hop path to a storage location and (b) not adding a table maintaining traffic unlike other DHT implementations. The query response time evaluation of 10^5 name insertions and 10^6 queries shows that with K=5, 95% of the queries complete within 86ms which is reasonable for a large scale system. The query response delay in DMAP is low because the updates do not introduce additional delays, and a name/address mapping is stored at multiple locations which can be queried by a node from a location closest to the itself. GMap provides scalability at the cost of not maintaining per-GUID state at all the servers, and keeping the cache size low for popular GUIDs. As even commercially available VN techniques [48] introduce a delay in the order of ∼150-200ms when using 3–4 network hops —almost twice as compared to the delay value of less than 100ms achieved by DMAP— we argue that these implementations are acceptable for NOVN.

5. Advanced MEC Techniques

The increasing softwarization of the network infrastructure, enabled by the advancements in computing power and virtualization techniques, has facilitated the support of new applications and services inside the network. Among different opportunities, network vendors and researchers have looked at solutions that explore how to better integrate inputs from the application logic to optimize network functionality [49, 53, 54, 18]. While this is a useful direction, more general solutions capable of extending beyond the limits of single networks are still lacking. Such working solutions would highly benefit distributed service scenarios, where advanced control mechanisms are required. In this section, techniques to efficiently utilize the named-object based virtualization in the MEC architecture are described.

5.1. Application Specific Routing

MEC architecture using NOVN is extended to support advanced routing 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

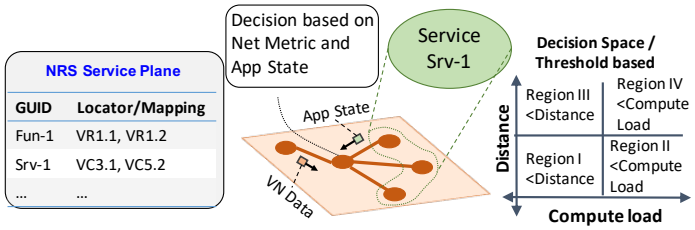


Figure 6: Application Specific Routing as an advanced routing service for the edge cloud use cases

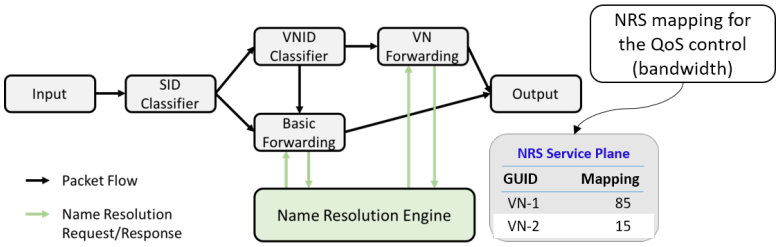


Figure 7: QoS control (traffic shaping) example in NOVN

deployed on top of any underlying infrastructure and integrating end point APIs to push application state into the VN.

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 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 6, 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 6). At branching locations, routers can then take informed decisions. For example, Figure 6 shows a decision space scenario where given thresholds define different states that can influence how routing decision. While the effectiveness of the ASR approach has been proposed in our previous work in the context of cyber physical systems [34], coupling *NOVN* with ASR can support a low latency and scalable solution for any service that would benefit of the locality of edge clouds.

5.2. Quality of Service Control

Maintaining QoS is a key requirement in the MEC architecture specifically to support low-latency applications. General QoS control mechanisms favor class-based approach where each traffic flow is assigned a QoS class identifier (QCI) to tackle issues such as admission control, queue management, and limiting bandwidth. The QCI value is pre-configured and cannot be adjusted dynamically during the run-time therefore lack required flexibility in the QoS control. Edge clouds are often co-located with the existing network equipment and often have limited computation and storage. 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. *NOVN* maps the VNID to its control parameter by querying NRS at the run-time, thereby providing a per-flow as well as per chuck based fine-grain QoS control. Figure 7 illustrates a traffic shaping example using NOVN. A service ID (SID) can be classified either as a part of virtual network or as a best effort traffic at the router during run-time. The VN forwarding engine queries the NRS service plane for the traffic rate information for the VNID and thus achieves the traffic shaping function inherently.

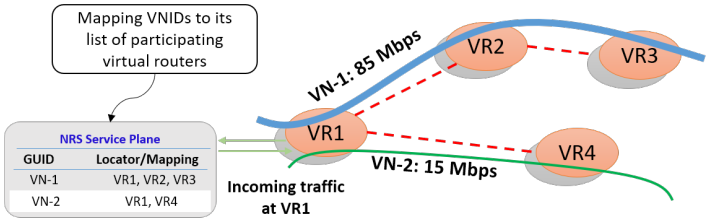


Figure 8: Network Slicing in NOVN

5.3. Network Slicing

In a multi-provider network scenario, to support a variety of services, network slicing allows statistical multiplexing of the available resources. NOVN allows a virtual router to be a part of multiple virtual networks thus enabling network slicing implicitly. The resource provisioning to each of the network slice is similar in spirit to the QoS control mechanism described above. Figure 8 illustrates as example of two network slices with a common virtual router (VR1). The VNID header lookup at the NRS provides the information about the participating virtual routers. Finally, each of the VN traffic is handled according to its own QoS policy thereby ensuring a cleaner approach to a sliced network.

5.4. Inter-Domain Peering

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 [12] and private interconnections over IXP locations via VLANs [6] 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.

6. Prototype and Experiment Set-up

In order to understand the achievable performance and feasibility of the proposed *NOVN* architecture and its associated advanced techniques, we implement a fully working prototype of the framework. The *NOVN* prototype uses as its foundation the MobilityFirst (MF) future Internet architecture [42] prototype [11]. The MF architecture is an example of how the named-object abstraction can 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 [35] 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 DHT based NRS 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. The open access code repository of the prototype is available for more details [1].

6.1. Core Prototype Components

Name Resolution Service. This replaceable component currently supports IPv4 and MF routing enabling the possibility of running both overlay and native virtualization solutions. We employ DMap's [50] DHT based implementation to evenly distribute mapping entries across all service instances. DMap's NRS is implemented

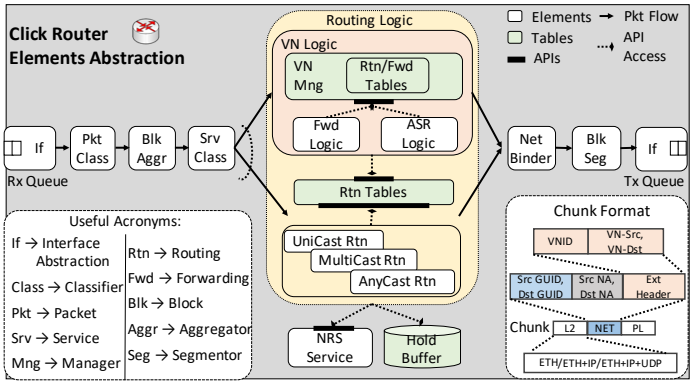


Figure 9: Click router elements graph for data plane flow

in Java and achieves the same performance guarantees demonstrated in simulation [11]. Further, this implementation ensures that each server is able to operate over any networking layer/technology without changes to the core code.

NOVN Routers. The software router is implemented as a set of forwarding elements and table objects within the Click modular router [24] run at user-level. As a baseline, the router implements dynamic-binding using GNRS, hop-by-hop reliable transport using a HOP [28] inspired protocol (by aggregation and segmentation of large chunks of data), and storage-aware routing [35]. 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 9). 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 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.

Advanced Service Extensions. The core NOVN prototype is further extended to support QoS control and network slicing by introducing a VNID based mapping technique. An MF chuck consists of number of packets. As shown in the Figure 10, resource management is achieved by marking the incoming packets in a chunk and then classifying them according to their VNIDs. The classified chucks are stored into a buffer which are pulled by a bandwidth shaper at a specified rate before aggregating them back as a chunk and sending at the

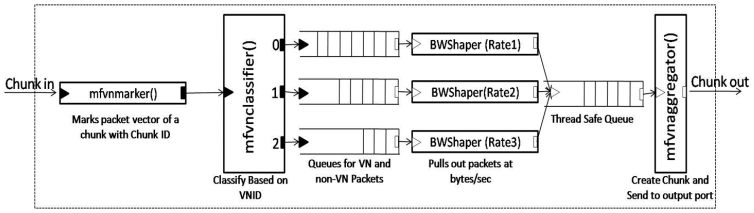


Figure 10: A sample traffic shaper implementation in *NOVN*

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.

Clients. In similar fashion, the baseline client network stack and API [10] 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.

6.2. Overlay based VN Implementation

For performance evaluation and comparison purposes, we implement an overlay based virtual network as follows. We integrate OpenVPN [17] based tunnels on top of a barebone IP router implementation using Click [24]. Tunnels connecting nodes are set-up between each pair of virtual routers (VR). Upon transmission, data is encrypted, encapsulated and tunneled to the neighboring virtual router. An encapsulation table maps an UDP tunnel to the public IP of the adjacent router at the overlay virtual router. An OSPF-like (Open Shortest Path First) protocol is used for routing at the IP layer. Predefined virtual paths are set using the aforementioned tunnels between virtual routers. Finally, the VN packet is implemented in click with the following fields: virtual source IP, virtual destination IP, transport identifier (UDP), OpenVPN header, source IP, destination IP and the payload.

The named-object based VN is evaluated by running an additional Name Resolution Service, which for simplicity we deploy using a single server. The network topology information consisting physical router connectivity, physical to virtual router mapping, and participating VN and service identifier, is disseminated at all the routers before the network bootstrap. A named-object VN packet has the following fields: source NA, destination NA, service ID, source GUID, destination GUID, VNID, source VGUID, destination VGUID and the payload, as described in the previous sections. Each virtual router is mapped to its physical router's GUID whose network address is queried from the NRS during run-time.

7. Performance Evaluation

A combination of routers, clients, and NRS servers have been deployed on the ORBIT testbed [43]. We select 19 nodes of the testbed and deploy different networks for the different use cases analyzed. Throughout all cases, we co-locate one NRS server with each router deployed. All nodes are interconnected via 1 Gbit Ethernet switches, creating a single L2 network. As the testbed provides a single L2 network, a logical split has been implemented within the click routers to enforce the topology. We present the following evaluation results: (a) a set of micro-benchmark experiments aimed at demonstrating the baseline computation overhead of our VN implementation against the baseline MF prototype, (b) an analysis of how to achieve network slicing in which different VNs can co-exist on the deployed network, (c) results on the traffic shaping to achieve QoS control, (d) an ASR edge cloud use case deployment scenario, and (e) a comparative analysis of NOVN with the traditional VN deployed as an overlay network on top of the current Internet architecture.

7.1. NOVN Performance Benchmarks

In order to understand the basic overhead introduced by running the virtual network logic on top of the baseline prototype, two 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 11 is used, but traffic generation is limited to VN-2 (blue color).

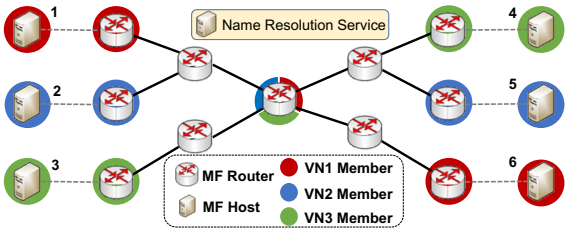


Figure 11: Network topology used for benchmarks

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

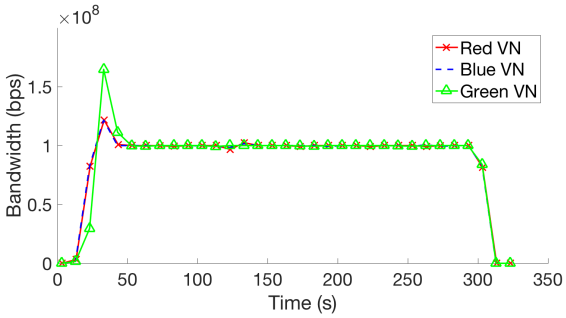
Latency and Throughput: Total values reported in Table 1 account for the sum of three time components: 1) the processing time of the software router (including potentially the VN logic); 2) the queries to the NRS (2ms RTT from the routers to the NRS with query results cached on the routers for 30 seconds); 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.

7.2. QoS Control

A major advantage inherent to the *NOVN* design is the possibility of performing multiplexing across different VNs by switching traffic based on a single header field, i.e., the *VNID*. 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 11. Best effort & managed traffic scenarios are evaluated without and with the QoS control mechanisms.

Multi VN Co-existence: Each traffic source (left side nodes), generates traffic at 100 Mbps. Figure 12 shows the results after running a five mins. experiment without employing traffic shaping. 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.

Managed Traffic Network Slicing: Using the topology described in Figure 11, traffic is generated at the rate of 100 Mbps at all three sources and managed in-network using the traffic shaper. VNID to allowed traffic rate mapping information is updated at the start of the experiment and dynamically retrieved during the run-time by querying NRS. As shown in Figure 13, each of the red, blue and green VNs pushed traffic up to their allowed limits of 0.5, 10 and 20 Mbps respectively. Similar to our previous observation, while the initial competition on the wire shoots up the traffic, due to the rate limiting implementation in the traffic shaper, all three VN's



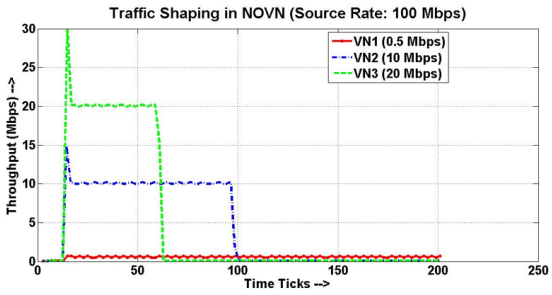


Figure 12: Multiplexing NOVN benchmark

Figure 13: QoS control benchmark in NOVN

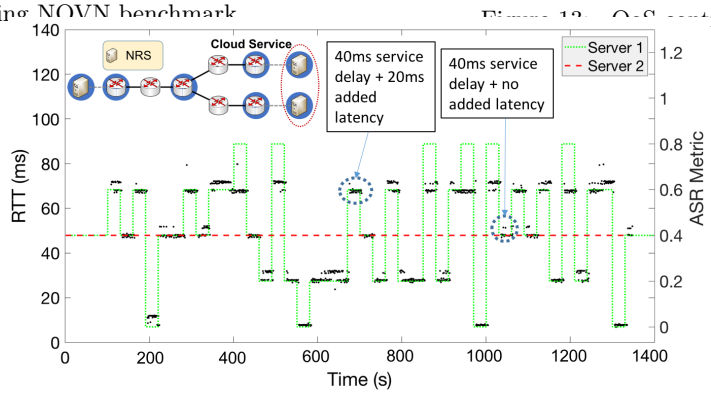


Figure 14: ASR edge cloud use case example

traffic stabilizes to reach up to their allowed capacity.

7.3. 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 14, 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 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 14 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 15. 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 across 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 16 shows the obtained results. The following should be observed: 1) up to ∼50ms, 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).

The *NOVN* framework, as described in Section 3, provides a clean way to define a virtual network topology through the use of the named-object abstraction. While using this technique it is possible to achieve the purpose

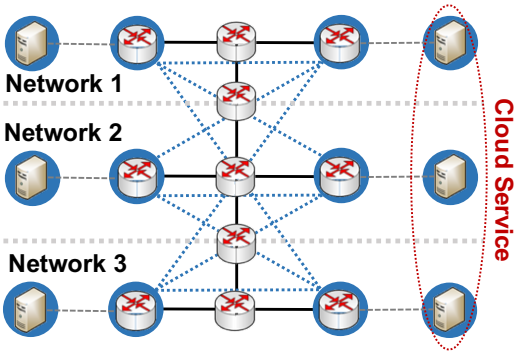


Figure 15: Network topology used for edge cloud deployment

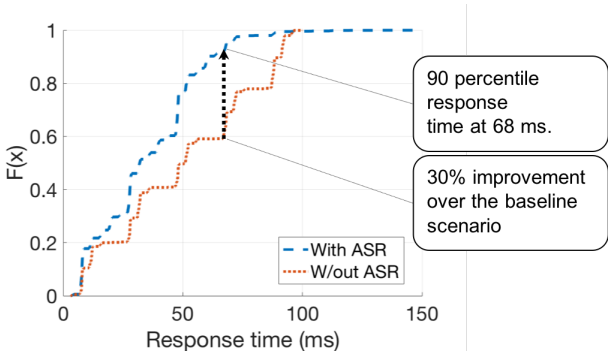


Figure 16: Response time for edge cloud deployment

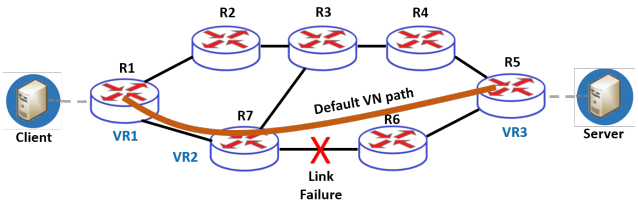


Figure 17: Network topology for VN comparison

of providing the high level mechanisms that characterize the system, additional details are required to provide a better sense of how *NOVN* can fully overcome the issues presented and how it could be deployed on top of the current TCP/IP Internet architecture.

7.4. Comparing NOVN with Overlay VN Solution

Overlay based virtual networking approaches rely upon complex packet processing at the router and the setting flags to carry extra information such as fragmentation. These approaches increase the round trip time (RTT) of a packet in the network and lowers data throughput, but may also fail the integrity of a tunneled packet for a larger size due to fragmentation flag set. This is generally avoided using a no fragmentation flag which causes loss of packets which are bigger than the MTU. Furthermore, overlay based solutions rely upon tunnels which are set up a priori. In case of a run-time failure, the tunnel needs to be set up again.

During link failures, overlay virtual networks lose packets until the link becomes active again or the route converges, incurring packet loss and lowering system throughput. In case of short duration link failures, the route converges to the same path and therefore the packet loss is directly proportional to the duration of the failed link. For the permanent link failures (equivalently, long duration link failures), the route converges to a different path and therefore the packet loss is proportional to the sum of losses due to timer expiration and route convergence time. Due to the slow start behavior of TCP, it is time expensive to create new tunnels in case of route change impacting throughput and delay.

In the embedded NOVN approach, network addresses are dynamically retrieved using a logically centralized geographically distributed NRS. The route is therefore resolved at the run-time by querying NRS which strictly decouples network functions from the hardware functions, shifting focus from complex packet processing to a simple packet forwarding. This also alleviates network configuration issues as assigning a GUID to a node is as simple as declaring a variable.

VN comparison experimental set-up We deploy both, the overlay as well as named-object based VN architectures described in the Section 6.2 on a small network on ORBIT as shown in the Figure 17. Seven routers form the core network and are connected via the Ethernet with 900 Mbps bandwidth. A simple ping application is run from the client to the server with different packet size to compare both the approaches in terms of protocol data plane overhead and recovery time from link failure.

Latency comparison. The round-trip delays associated with the data traversed across the network capture the encryption, tunneling, encapsulation and any other packet processing; therefore, the RTT can be approximated as protocol overhead for the architecture comparison. Table 2 shows round trip times (RTTs) obtained for

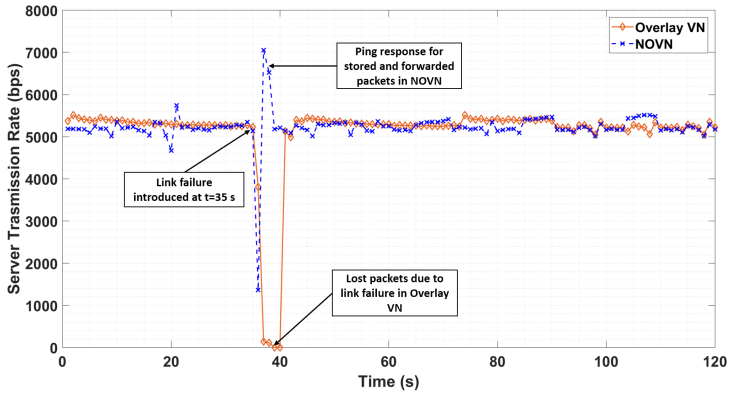


Figure 18: Comparing effect of link failure for overlay VN and NOVN

different packets sizes for overlay and *NOVN*. The ping latency is averaged over a large number of pings (>1000). We notice that *NOVN* experiences increased latency compared to the results obtained by the overlay network. We attribute the added latency to the periodic NRS queries in case of NOVN whereas overlay network is pre-configured and the only overhead it experiences is in replacing headers. Moreover, the MF based solution uses 160 bits long names for objects identification, a large increase in headers overhead compared to the other solution. Even considering these elements, *NOVN* still achieves close performance results compared to the overlay network.

Packet Size	Overlay (RTT in ms)	NOVN (RTT in ms)
64 B	6.1	7.2
1400 B	6.3	7.5
5 KB	7.9	9.6
10 KB	9.4	12.9
50 KB	13.6	17.8
100 KB	20.1	24.4
500 KB	72.4	78.5

Table 2: Overhead comparison.

Link Failure. We emulate link failures by introducing packet loss at the link between the routers R6 and R7. We analyse two cases: (i) 100ms (short term failure) and (ii) 100s (long term failure), using the RandomSample element in click router, sampling packets at the loss rate 1 for the specified duration. For the first case, neither of the approaches had enough time to react to the failure and converging to a new path; both cases simply recover once the link is re-established. Packet loss observed in the overlay case is $MTTR*rate$ while for *NOVN* there is no loss due to store and forward capability of the router inherited from the MF architecture. In the 100s case, the overlay approach has to wait for the routing protocol to re-converge to a new path and set-up new VPN [3] tunnels before a client can get ping responses back from the destination. In contrast, *NOVN* reacts much faster as the next node's network address is dynamically resolved by querying the NRS. Figure 18 compares the effect of link failure for both the cases. The server transmission rate is a ping response to the ten 64 B packet ping requests sent by the client shown the Figure 17. The failure is introduced at time t=35 seconds. NOVN recovers in about 1 second without losing any packets due to its in-network store and forward scheme. Overlay VN loses the packets equivalent to the mean time to recover (MTTR) which is more than 5 seconds in this case.

8. 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 [5, 23] to lower layer integrations using tag switching [25, 15]. *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 allowing greater control and interaction to application and services populating networks. SDN [32] and its extensions [18] have provided contributions to this research area, but have been limited their scope to single domains. Active networks [37] had also been proposed as an extreme solution to the problem, allowing packets to carry instructions interpreted by the network fabric. Multi-domain approaches have mostly focused on single specific issues, such as anycast delivery or path selection to distributed services [49, 53]. Similar to ASR, Internet standardization organizations have also introduce overlay approaches for custom routing [27]. 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 [26, 36, 42] and name separation [33, 16] works.

9. 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, *NOVN* provides a solution that offers the logical simplicity of L2 network virtualization which augmented with the advanced mobile edge cloud (MEC) techniques such as application specific routing, network slicing and QoS control, achieves 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. Performance evaluation of various MEC scenarios are presented and the solution is compared with the overlay based virtual networks. Results show that *NOVN* provides faster path recovery and incurs no packet loss during link failure. The ASR improves the latency performance by 30% as compared to the baseline approach for a 90 percentile response time at 68 ms. Future work includes evaluating ASR techniques applied to large scale edge cloud scenarios.

Acknowledgement

This research is supported by NSF Future Internet Architecture–Next Phase Award CNS-1345295.

References

References

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

[2] Thomas Anderson, Larry Peterson, and others. 2005. Overcoming the Internet impasse through virtualization. *Computer* 38, 4 (2005), 34–41.

[3] YL Andersson, T Madsen, and AB Acreo. 2005. *Provider Provisioned Virtual Private Network (VPN) Terminology.* RFC 4026. `https://tools.ietf.org/html/rfc4026`

[4] Ronald T Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments* 6, 4 (1997), 355–385.

[5] Andy Bavier, Nick Feamster, and others. 2006. In VINI veritas: realistic and controlled network experimentation. *ACM SIGCOMM Computer Communication Review* 36, 4 (2006), 3–14.

[6] E. Bell, A. Smith, and others. 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`

[7] Flavio Bonomi, Rodolfo Milito, and others. 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.

[8] Francesco Bronzino, Sumit Maheshwari, and others. 2019. NOVN: named-object based virtual network architecture. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*. ACM, 90–99.

[9] 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.

[10] Francesco Bronzino, Kiran Nagaraja, and others. 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.

[11] 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.

[12] Ignacio Castro, Juan Camilo Cardona, and others. 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.

[13] Abhishek Chandra, Jon Weissman, and Benjamin Heintz. 2013. Decentralized edge clouds. *IEEE Internet Computing* 17, 5 (2013), 70–73.

[14] Brent Chun, David Culler, and others. 2003. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review* 33, 3 (2003), 3–12.

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

[16] Dino Farinacci, Darrel Lewis, and others. 2013. *The locator/ID separation protocol (LISP)*. RFC 6830. `https://tools.ietf.org/html/rfc6830`

[17] Markus Feilner. 2006. *OpenVPN: Building and integrating virtual private networks*. Packt Publishing Ltd.

[18] Andrew D Ferguson, Arjun Guha, and others. 2013. Participatory networking: An API for application control of SDNs. In *ACM SIGCOMM computer communication review*, Vol. 43. ACM, 327–338.

[19] V Fuller, D Farinacci, and others. 2013. *Locator/ID separation protocol alternative logical topology (LISP+ ALT)*. Technical Report.

[20] 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.

[21] Yi Hu, Roy D Yates, and Dipankar Raychaudhuri. 2015. A Hierarchically Aggregated In-Network Global Name Resolution Service for the Mobile Internet. *WINLAB: New-Brunswick, NJ, USA* (2015).

[22] Jinho Hwang, K K_ Ramakrishnan, and Timothy Wood. 2015. NetVM: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management* 12, 1 (2015), 34–47.

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

[24] Eddie Kohler, Robert Morris, and others. 2000. The Click modular router. *ACM Transactions on Computer Systems (TOCS)* 18, 3 (2000), 263–297.

[25] Teemu Koponen, Keith Amidon, and others. 2014. Network virtualization in multi-tenant datacenters. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 203–216.

[26] Teemu Koponen, Mohit Chawla, and others. 2007. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 181–192.

[27] 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`

[28] Ming Li, Devesh Agrawal, and others. 2009. Block-switched Networks: A New Paradigm for Wireless Transport.. In *NSDI*, Vol. 9. 423–436.

[29] Ratul Mahajan, David Wetherall, and Tom Anderson. 2001. A study of BGP origin as changes and partial connectivity. *Slide Presentation, University of Washington, Asta Networks,(ritual@ cs. washington. edu)(22 pages)* (2001).

[30] Sumit Maheshwari, Shalini Choudhury, and others. 2018. 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, 1–6.

[31] Sumit Maheshwari, Dipankar Raychaudhuri, and others. 2018. Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Applications. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 286–299.

[32] Nick McKeown, Tom Anderson, and others. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.

[33] R. Moskowitz, P. Nikander, and others. 2008. *Host Identity Protocol*. RFC 5201. `https://tools.ietf.org/html/rfc5201`

[34] Kiyohide Nakauchi, Francesco Bronzino, and others. 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.

[35] 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.

[36] Jianli Pan, Subharthi Paul, and others. 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.

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

[38] Zhijing Qin, Grit Denker, and others. 2014. A software defined networking architecture for the internet-of-things. In *2014 IEEE network operations and management symposium (NOMS)*. IEEE, 1–9.

[39] Sophie Y Qiu, Patrick D McDaniel, and others. 2006. Characterizing address use structure and stability of origin advertisement in Inter-domain routing. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*. IEEE, 489–496.

[40] Paul Quinn and Jim Guichard. 2014. Service function chaining: Creating a service plane via network service headers. *Computer* 47, 11 (2014), 38–44.

[41] 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.

[42] 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.

[43] Dipankar Raychaudhuri, Ivan Seskar, and others. 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.

[44] Howard Rheingold. 1991. *Virtual reality: exploring the brave new technologies*. Simon & Schuster Adult Publishing Group.

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

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

[47] Abhigyan Sharma, Xiaozheng Tie, and others. 2014. A global name service for a highly mobile internetwork. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 247–258.

[48] Daniel Turull, Markus Hidell, and Peter Sjödin. 2014. Performance evaluation of OpenFlow controllers for network virtualization. In *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 50–56.

[49] Vytautas Valancius, Nick Feamster, and others. 2010. Wide-Area Route Control for Distributed Services.. In *USENIX Annual Technical Conference*.

[50] Tam Vu, Akash Baid, and others. 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.

[51] Yi Wang, Eric Keller, and others. 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.

[52] Timothy Wood, KK Ramakrishnan, and others. 2011. CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In *ACM Sigplan Notices*, Vol. 46. ACM, 121–132.

[53] 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.

[54] Yiannis Yiakoumis, Sachin Katti, and Nick McKeown. 2016. Neutral Net Neutrality. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 483–496.

[55] Haijun Zhang, Na Liu, and others. 2017. Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE communications magazine* 55, 8 (2017), 138–145.