OpenEdge: A Dynamic and Secure Open Service Edge Network

Josh Kunz, Christopher Becker, Mohamed Jamshidy, Sneha Kasera, Robert Ricci, and Jacobus Van der Merwe
Flux Research Group, School of Computing, University of Utah
josh.kunz@utah.edu, {cbecker | jamshidy | kasera | ricci | kobus}@cs.utah.edu

Abstract—**High performance edge networks, such as fiber-to-the-premises (FTTP), are increasingly being deployed by municipalities and communities to support advanced services and applications. The complexity of operating these networks often means that their full potential is not being reached and they are relegated to being fast access pipes to the Internet. In this paper, we present our work on OpenEdge, a dynamic and secure open service edge network architecture. OpenEdge provides a control architecture that automates the configuration of the edge network in a cloud-like manner to simplify the introduction of new network services and applications.**

I. Introduction

Despite significant advances in edge network technologies—for example fiber-to-the-premises (FTTP) deployments—the service model of these networks remains largely unchanged.[1] Specifically, the primary purpose of these (monolithic) access networks is to provide access to the Internet and other services (like IPTV and VoIP) offered by the Internet access provider, who also owns and operates the physical network infrastructure. Access to the Internet does of course enable others, e.g., Netflix, to offer *over-the-top* services. While some over-the-top services have been successful, we argue that there is a need for a new edge network service model that more fully exposes the capabilities and distinctive characteristics of these networks, to enable an even wider variety of services. First, many services (e.g., smart grid applications) do not require conventional Internet access, and would in fact be better served by networks that are separated from the Internet. Second, edge networks have low latency and high capacity characteristics, which can enable services that are simply not feasible in an over-the-top scenario, e.g., networks aimed at content manipulation by clusters of digital media companies [1]. Third, we argue that a dynamic service model, where services are offered and consumed on-demand—for the duration of a tele-health session, or an immersive entertainment application—would be desirable.

A related trend we observe is that many municipalities and communities, often underserved by incumbent providers, are deploying their own network infrastructures [2]. These municipal edge networks are often first deployed to provide services to specific anchor institutions, like public safety [3], [4]. While these edge networks are inherently capable of supporting other services, the complexity of operating and providing services on them frustrates such efforts. Often, such networks are relegated to being access networks with limited service offerings.

[1]We use the term *edge network* intentionally in place of the more common *access network*; "access" implies that the primary (or only) purpose of these networks is to provide access to the Internet. As we argue below, reducing edge networks to just Internet access is a key part of the problem addressed in our work.

We argue that current FTTP and municipal edge network deployments are in an analogous state to where data centers were before the sea change brought about by cloud computing. Like cloud control architectures that transformed data centers into cloud computing infrastructures, what is needed to unlock the potential of edge networks is a network control architecture that not only automates the management of network resources, but also allows new services and applications to be created, deployed and decommissioned in a highly dynamic and secure fashion. Such a network control architecture enables a model where the barrier to entry for becoming a service provider is significantly lowered, empowering a wide variety of unique services to be provided on the infrastructure. In addition to today's long-lived ISP, IPTV and VoIP services, we envision a rich marketplace of dynamic services and applications. Specifically, services that are both "local" (exploiting the high capacity, low latency edge network capabilities), as well as highly dynamic services or applications that might exist for relatively short periods of time.

Towards realizing this vision, we present our work on *OpenEdge*, a dynamic and secure open service edge network architecture. OpenEdge is broadly modeled after earlier open access network approaches [5], [6], [7] in that it assumes the possibility of separating the entity that owns and operates the network, the network operator (NO), and the entity (or entities) that provide services and applications on the network, the service providers (SPs). Unlike earlier open access networks, OpenEdge provides a control architecture that automates the operation of the edge network in a cloud-like manner to simplify the introduction of new network services and applications. While inspired by the cloud, open edge networks are unique: the topology of the network and the location of devices within it are everything to its customers; most devices in the network (home and business computers, TVs, IoT devices, etc.) are neither owned nor controlled by the network operator; and the physically distributed nature of the network (on telephone poles and in roadside utility boxes) means that the physical security of many parts of the network is not assured.

In OpenEdge, we address a number of key challenges in realizing an open edge network architecture. First, finding the "right" abstraction to enable services and applications to utilize the network. Second, like in a cloud computing platform, the shared tenancy nature of an OpenEdge environment implies security concerns not present in monolithic access networks. The key security concern in a physically distributed, multi-tenant (or multi-service) OpenEdge environment is unauthorized use of a service and the network resources associated with it. For example, unauthorized use of a public safety network, e.g., by tech-savvy gamers, might jeopardize public safety when there is an emergency. Similarly, allowing unauthorized access to a power company network in a smart city setting could open the door for compromise of cyber physical systems associated with such a network. Our third challenge relates to the realization that end users are increasingly mobile and as such, require services obtained via an OpenEdge environment to move with them. Such *ubiquitous service access* would enable a physician from a teaching hospital, temporarily filling in at a health clinic, to receive her normal access to specialized resources on a health network while at the clinic location.

In OpenEdge we address these challenges via two complementary and interacting components. First, *FlowOps* implements the network abstraction offered to service and application providers and realizes the underpinnings of that abstraction on the physical network. FlowOps provides a simple API that allows service/application providers to treat the entire network as a "big switch". This leaves the service providers free to focus on the semantics and logic of the service/application they want to provide. The minimal semantics of the FlowOps API means that the network operator is free to implement the underlying connectivity using any technology they desire. The only requirement is that strong isolation is provided between network resources associated with different services.

The second OpenEdge architectural component, *SecureOps*, deals with authenticated access to the network and services/applications available on the network. The design of SecureOps is inspired by the security framework used in cellular networks. Specifically, in SecureOps, access to a network service is authenticated via security credentials contained in a virtual subscriber identity module (V-SIM). Unlike a physical SIM, the V-SIM allows separate credentials for different services, allowing a more fine-grained service model. Our V-SIM can be realized solely in software, or made more secure by utilizing a Trusted Platform Module (TPM) [8], [9]. A TPM is a hardware module that is installed in a device which can securely store and use cryptographic values.

Finally, because authentication and service creation happen dynamically in OpenEdge, ubiquitous service access follows from our basic design. With OpenEdge we can remove the tight coupling that exists in current access networks between the services being provided and the physical network location at which those services are being provided. In OpenEdge, subject to the availability of network resources, authentication and service provisioning can occur wherever an end-user attaches to the network.

We make the following contributions:

- We present the OpenEdge network architecture which enables services to be provided over edge networks in a cloud-like fashion.
- We present the design and implementation of FlowOps, the OpenEdge component that provides a simple network abstraction to service providers, easing the realization of edge network services and applications.
- We present the design and implementation of SecureOps, an OpenEdge component that uses a virtualized SIM abstraction and the OpenEdge authentication protocol to provide fine-grained network and service authorization and access control.
- We show the utility of our approach by developing a number of example services, including a "composed" service, consisting of two independent service providers.
- We present a detailed evaluation of our architecture. We perform a security analysis of the SecureOps component and present a performance analysis of the FlowOps component, as well as the OpenEdge architecture as a whole. Our results show that, in a simulated 10K node FTTP edge network, FlowOps can perform service creation that involves 10K end-nodes in about $6\,{}^1/_2$ seconds.

II. OpenEdge Architecture

An overview of the OpenEdge architecture is depicted in Figure 1 (a). As shown, OpenEdge builds on and controls a physical edge network. In OpenEdge we assume that this network is controlled and operated by a network operator (NO) using the OpenEdge control architecture and that independent service providers (SPs) provide services and applications on the infrastructure. Unlike conventional open access networks, the OpenEdge control architecture lowers the barrier-to-entry of becoming a service provider by allowing services to be dynamically created in a cloud-like fashion.

The functionality of OpenEdge is provided by two components: FlowOps and SecureOps. FlowOps provides connectivity across the NO's network to the various service providers who want to provide services on the network. FlowOps does this through a "virtual link" abstraction where service providers can request connectivity between any two points on the NO network. SecureOps provides authentication and location mapping services for the end-user, network operator, and service providers, while taking into account the complex business relationships that exist between these parties. SecureOps provides these services through the use of a Virtual SIM (V-SIM), a shared secret between a service provider and a user of that service. V-SIMs are used as part of the SecureOps protocol that allows a service provider and the network operator to verify the identity of a user. Together FlowOps and SecureOps provide the "user-token" abstraction, a binding between an identity (i.e., V-SIM) and a physical port on the network edge. These tokens are used by service providers as the endpoints of virtual links. This way, connectivity is based on the identity of the subscriber rather than the subscriber's physical location.

Figure 1 (a) depicts a walk-through of the OpenEdge system. Consider a service provider that has just signed up a new customer. In OpenEdge, the first thing the service provider would do is generate a new V-SIM and then assign it to that customer. The customer would then go to their home computer or router and install that V-SIM into their V-SIM manager (Section II-B2). Once the V-SIM is installed, the V-SIM manager will attempt to *authenticate* against the service provider that allocated the V-SIM. (Section II-C2). If that authentication is successful, a new user-token is generated, which *binds* that V-SIM to the port on the network on which the user's authentication request originated. Both FlowOps and the authenticating service provider are notified of this binding. The service provider can then use the user-token in subsequent *service setup requests*. For example, if the user had signed up for internet service, at this point the ISP would tell FlowOps to create a virtual link between the user—identified by the user's token—and its border router. Since FlowOps knows the mapping between a particular user-token and switch-port pair, it can configure the forwarding elements in the network to *realize* this service.

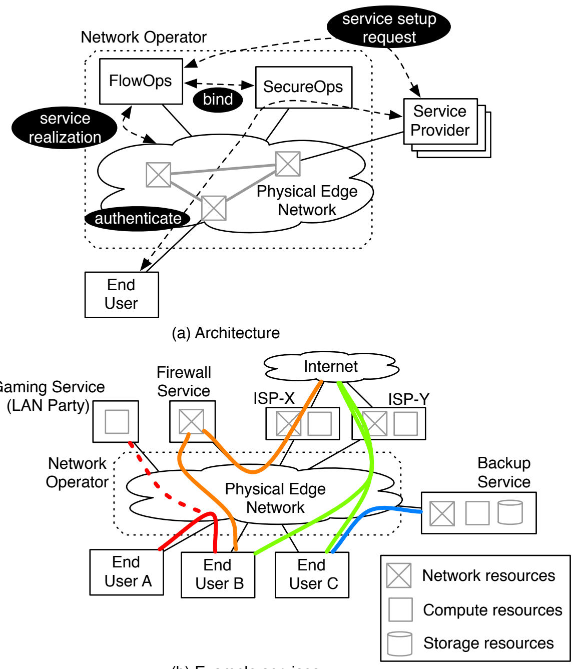

Fig. 1. OpenEdge Architecture and Example Services

Example services: To illustrate the open service environment enabled by OpenEdge we depict a number of example services in Figure 1 (b). We envision that current long-lived services, like Internet access, will still be provided in this environment. As illustrated by ISP-X and ISP-Y, multiple such services can be operational concurrently in OpenEdge. Additionally, the separation of identity and physical location allows OpenEdge to provide services that are not possible on standard edge networks. For example, OpenEdge can dynamically re-bind user-tokens to any point on the network at the request of an authenticated service provider allowing service providers to "chain" themselves into a user's already existing service. For example a firewall can re-bind a user-token that was bound when a user authenticated against an ISP to one of their edge ports, create a virtual link to the user from a separate edge port, and then perform a firewall action between the two.

Due to OpenEdge's increased dynamicity, we can also have services that are only created for short periods of time, or the lifetime of a particular action. Consider a backup service that only needs connectivity to a user's device during off-peak early morning hours when pricing may be cheaper. Such a service could dynamically create a link to the user, backup the user's device, and then destroy the virtual link with no human intervention. OpenEdge can also be used to realize services with no physical presence on the network. For example, consider a "Gaming" service that connects various users to play a networked game over the high-speed edge network. The service provider doesn't need to have any physical presence at the edge, it can create virtual links between the players as its business logic sees fit. We envision that service provider resources (such as servers to host "Gaming" match-making services) might be provided by a cloud platform operated by the network operator, or a separate "platform" service provider.

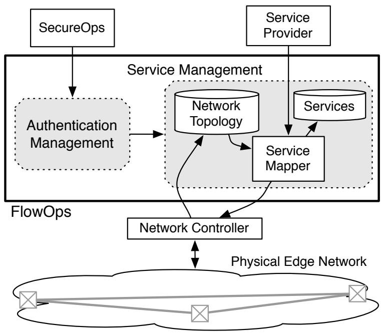

Fig. 2. FlowOps Architecture

In the remainder of this section we first describe the OpenEdge components before detailing the OpenEdge Authentication Protocol.

A. *FlowOps: Network Management*

Figure 2 shows the main components of the FlowOps architecture. The functionality of FlowOps divides into two main functions: Authentication Management, which cooperates with SecureOps to implement the user-token abstraction, and Service Management, which deals with the virtual link abstraction provided by FlowOps.

1) Authentication Management: As shown in Figure 2, FlowOps maintains some state about what user-tokens have been bound, and what switch-port pairs on the network the user-tokens have been bound to. As part of the authentication process SecureOps delivers two pieces of information to FlowOps (Section II-C2): a notification that a new authentication request has been sent via a particular switch and port on the network, and a notification that the request has completed (if the request completes successfully). Each notification has a unique (per-request) identifier so that FlowOps can identify which "new authentication request" notification a "successful authentication" notification applies to. FlowOps uses these notifications to maintain a mapping between users (identified by their V-SIM) that have successfully authenticated and the network location (i.e., port on a switch) from which they authenticated (i.e., user-tokens).

2) Service Management: FlowOps provides the logic that maps the high level desires of various service providers (SPs) to their low-level realization in the network operator's (NO's) edge network. The FlowOps API has minimal semantics, allowing SPs to connect endpoints using "virtual links" into complex topologies. The NO is free to realize this abstraction using any underlying network technology, e.g., SDN, VLANs or MPLS.

As shown in Figure 2, FlowOps maintains a network topology database. This database can be filled from an already-existing topology database, or built by performing network topology discovery. When the Service Mapper component of FlowOps receives a request to create a set of virtual links, it uses the network topology and user location information to build a logical topology composed of the virtual links, mapped onto the physical edge network.

To do this, the mapper first translates the user-tokens given in the request into physical user locations using the previously created user-token bindings stored in a database in the Authentication Management component of FlowOps. Having obtained the location of each user-device, the Service mapper then builds routes through the network using a spanning tree algorithm optimized for star and ring topologies common in edge networks (In Section III-B we show our method scales to large edge networks). Once the Service Mapper discovers a set of physical devices that can realize this logical topology, it logs this information in the Services database. Thereby, any future modifications to this logical topology know its current configuration. Finally, the Service Mapper contacts the network controller and informs it of the logical network configuration in order to realize the service request.

B. SecureOps: Security Management

SecureOps provides security for all role players in an OpenEdge environment while taking into account the relationships between participants. It provides a mechanism for authenticating requests for network resources and services, allowing only authenticated use of the network, with fine-grained access control and ubiquitous service access.

1) Threat Model: The security risks associated with OpenEdge stem from two fundamental properties of this environment. First, an edge network deployment is physically more vulnerable than a datacenter, enterprise or provider backbone deployment. In an edge network, edge switches (or optical network terminals (ONTs)) are typically mounted in a relatively unsecured manner on residential or business premises. This affords relatively easy physical access to the network, e.g., to tap or snoop on the access link, or to insert a rogue inline device in the access link. Second, in the OpenEdge environment, we assume different actors: end users (subscribers), service providers, and a network operator. Below we consider adversarial models from the perspective of each of these actors.

Network Operator and Service Provider Perspectives: In both of these perspectives, we assume that the goal of the adversary is to use the resources and bandwidth without authorization. The adversary does not have valid credentials for the intended target and is not using (or behind) an end-device that has already been authenticated by the target. We assume the adversary only has access to limited resources such as an authentication server and provider front ends, which is enforced by FlowOps using traditional traffic isolation techniques such as VLANs or MPLS. We also assume the network operator trusts all network components it has control over and that service providers trust the network operator, particularly the information on authentication attempts from the network operator.

End-user Perspective: We consider a number of different scenarios from the end user perspective. The first scenario involves *passive listening for credentials*, where the goal of the adversary is to intercept subscriber credentials and data using untrusted hardware. E.g., the adversary is able to place itself between the subscriber and the service provider (e.g., a compromised edge switch or rogue device on the access link). For this scenario we assume that the service provider is trusted and that the end-device is trusted for storing the credentials and performing session data encryption.

The second scenario involves a *rogue authentication server*. The goal of the adversary is to pose as an authentication server between the user and the real service provider in order to retrieve authentication information. We assume the adversary is able to insert a node that mimics the functionality of an authentication server and can get the subscriber to send credentials to it.

The third scenario involves an *untrustworthy service provider*. In this case, we assume that a legitimate service provider, i.e., a service provider who is authorized to provide services on the edge network, attempts to provision services on behalf of an end user, without having received an actual service request from the user in question. Since this is a legitimate service provider, we assume that the service provider is able to access FlowOps in order to request the creation of network resources.

The final scenario involves *tampering with the end-device*. Here we assume the goal of the adversary is to copy the subscriber credentials (including the shared secret) from one end-device to another. Note that the legitimate end user might be complicit with the adversary, e.g., copying their credentials to friends to allow them access to certain services.

2) SecureOps Architecture: Our SecureOps authentication framework was inspired by the cellular security architecture [10]. Applying this framework to an open edge network environment required a substantial redesign. We adopt the notion of a subscriber identity module (SIM) to store authentication information on the end-user device, and generalize the authentication protocol to support multi-player authentication. Given the multiple role players in an OpenEdge network, in SecureOps we virtualize the SIM to allow authentication with both the network operator and multiple service providers. One virtual subscriber identity module (V-SIM) is stored on the end-user device for the network operator, with additional V-SIMS for each service provider the user subscribes to. In a cellular environment, the SIM is stored on a physical card (SIM card) in the mobile device. In SecureOps, it is managed by a piece of software which runs on the end-user device called a *V-SIM manager*, with additional security optionally provided by a trusted platform module (TPM). The network operator acts as a proxy for authentication to collect the information necessary to set up the connection and provide the user-token abstraction used by the service provider without requiring knowledge of the shared secret between the user and the service provider.

Typical workflow: To attach to the network, a new user needs a V-SIM from the network operator. The user obtains this V-SIM by interacting with the network operator's portal, a website that collects account details such as name, address, payment information, etc. The protocol used to bootstrap the new V-SIM is described in Section II-C1. With this V-SIM, the user authenticates to the network as described in Section II-C2. For each new service that the user wishes to subscribe to, a similar process is repeated, except that the user interacts with the relevant service provider's portal instead of the NO's portal. Once the user has a V-SIM and has authenticated with the service, the service can contact FlowOps and/or its own infrastructure to set up connectivity to the user as needed.

Service Provider setup: We note that for an SP to become operational on the network, it goes through a similar authentication process to receive V-SIMs from the network operator for all its devices connected to the edge network. Additionally, the SP will allocate V-SIMs to all of the infrastructure devices it controls, and have them authenticate against the SP's authentication server, so that it can prove to the OpenEdge system that it owns the devices and obtain user-tokens for them. Since it is essentially the same process and uses the same authentication protocol, we omit the details.

C. OpenEdge Authentication Protocol

The OpenEdge authentication protocol enables ubiquitous service access by authenticating users' network locations. Service providers tie services to users' V-SIMs, rather than physical locations, allowing fine-grained access control wherever the user is.

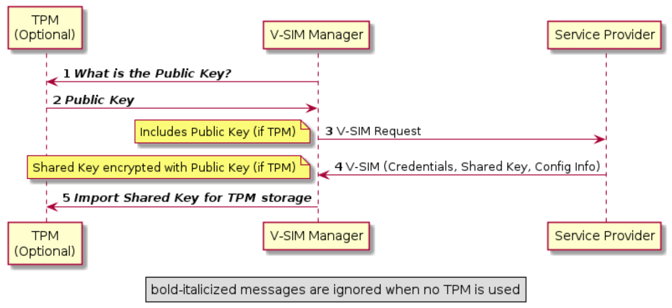
TPM (Optional)
V-SIM Manager
Service Provider
1 What is the Public Key?
2 Public Key
Includes Public Key (if TPM)
3 V-SIM Request
Shared Key encrypted with Public Key (if TPM)
4 V-SIM (Credentials, Shared Key, Config Info)
5 Import Shared Key for TPM storage
bold-italicized messages are ignored when no TPM is used

Fig. 3. OpenEdge V-SIM Creation Process

1) Bootstrapping: Figure 3 provides an overview of the process for creating a new V-SIM, which will authenticate a user to an SP. A V-SIM manager, running on the user's device, requests a new V-SIM from the SP. The V-SIM returned by the service provider includes user credentials (Provider ID and User ID) and a shared secret key (for authentication).

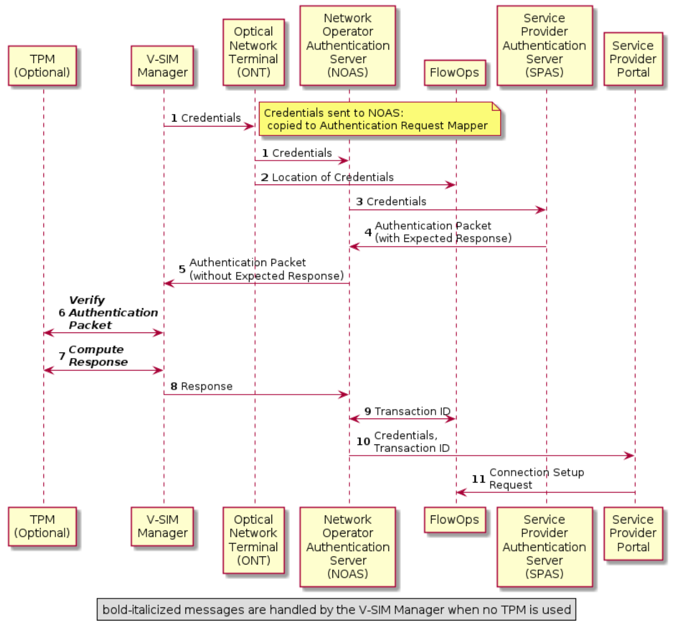
TPM (Optional)
V-SIM Manager
Optical Network Terminal (ONT)
Network Operator Authentication Server (NOAS)
FlowOps
Service Provider Authentication Server (SPAS)
Service Provider Portal
1 Credentials
Credentials sent to NOAS: copied to Authentication Request Mapper
1 Credentials
2 Location of Credentials
3 Credentials
4 Authentication Packet (with Expected Response)
5 Authentication Packet (without Expected Response)
6 Verify Authentication Packet
7 Compute Response
8 Response
9 Transaction ID
10 Credentials, Transaction ID
11 Connection Setup Request
bold-italicized messages are handled by the V-SIM Manager when no TPM is used

Fig. 4. OpenEdge Authentication Protocol

2) Authentication Protocol: The OpenEdge authentication protocol (Figure 4) is initiated by the user's V-SIM manager, and authenticates against a particular service; this is repeated for each service. The initial message, which includes the credentials established during V-SIM creation, is forwarded through the network to the Network Operator Authentication Server (NOAS). As part of the initial exchange, the user's first-hop switch (the ONT) informs FlowOps of the network location the request originates from. The NOAS mediates the rest of the authentication process, forwarding the credentials to the Service Provider Authentication Server (SPAS), which creates a challenge for the V-SIM and informs the NOAS of the expected response. Assuming the V-SIM produces the correct response to the challenge, the NOAS informs FlowOps that the authentication was successful and receives a user-token for the authentication. This token is sent (along with the user's credentials) to the service provider, which uses it to request the creation of network resources to realize its service for the user.

The authentication process must be initiated by the user: this prevents a rogue service provider from establishing connections to users who have not requested them. Even if the rogue authentication server tried to mimic a user authentication (since it knows the shared secret) the generated user-token will still be bound to an edge-port controlled by the rogue service provider, not by the end user. Similarly, the connection setup request at the end of the protocol must come from the service provider since they are the only party that knows the user-tokens for both ends of the virtual link. This prevents users from connecting to services they have not subscribed to.

3) TPM: Using a TPM (Trusted Platform Module) in the device hosting the V-SIM manager is optional, but it improves security by making it impossible to 'clone' V-SIMs or copy them from one device to another. During V-SIM creation, the service provider encrypts the secret it generates with the end-user's TPM's public key. Since the TPM's private key never leaves the TPM, neither does the secret, and the TPM is responsible for computing the challenge response during authentication. (The necessary functions are available in the TPM 2.0 standard.) It should be noted that, while use of a TPM does prevent a compromised device from reading the secret itself, it does not ensure that software running on the end-device is an unmodified V-SIM manager. This does not affect the security of the protocol, but does mean that a compromised V-SIM manager could operate contrary to the user's wishes.

III. EVALUATION

In this section we present an evaluation of our prototype of the OpenEdge architecture. The OpenEdge architecture implements novel features that do not exist on access networks today. In order to provide these features on a network without OpenEdge significant manual configuration would be required, making these features impossible to provide on a modern access network at a reasonable cost. We argue that any system that allows multiple service providers to form a truly *open* edge network must use an automatic configuration system like OpenEdge. To illustrate the utility of OpenEdge, we first describe a number of services we have prototyped on our architecture, and evaluate their end-to-end performance on our test network. We then perform a performance evaluation of the FlowOps component and end the section with a security analysis of OpenEdge.

A. OpenEdge Services

We have built three services using the OpenEdge framework: an ISP service, a Firewall service, and a LAN-Party service. These services have been deployed and performance tested in a small lab network (Figure 5) within a regional phone company. This section describes how these services are realized on OpenEdge.

The nodes labeled *Alice* and *Bob* are standard Linux hosts that serve as endpoints. They run V-SIM managers that authenticate against the NOAS and SPASes running on the *Compute Platform*. The *Compute Platform* also runs our ISP provider portal, and the LAN-Party provider portal. The Firewall provider portal runs on the node labeled *Firewall Equipment* so that it can more easily control the VMs running on that host.

ISP Service: Our ISP is a simple service that connects its clients to an Internet border router. The service bootstraps itself by authenticating the edge network port its border router is connected to against its own SPAS. When a user signs up for (and subsequently authenticates against) the service, it uses OpenEdge to create a logical link between the user and its Internet border port. This service was implemented in only 43 lines of code.

The timings in Table I are from 100 trials where the node labeled *Alice* in Figure 5 authenticates against the ISP and a link to the Internet border route is created. We see that overall

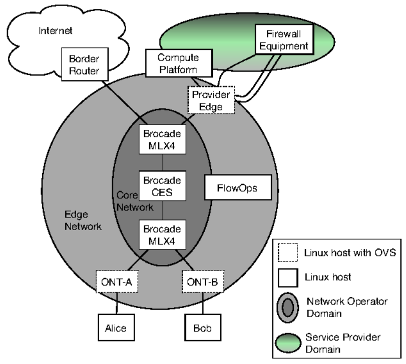

Fig. 5. OpenEdge Test Lab Network

TABLE I
END-TO-END ISP SERVICE CREATION (100 TRIALS)

Action	Median (ms)	Std. Dev.
Security Protocol	51.127	14.154
Service Setup	146.506	5.361
Overall	198.150	15.249

this operation can be performed in about 200ms, with 50ms being taken by the security protocol and 150ms by FlowOps.

LAN Party Service: The LAN party service allows users to create arbitrary LANs without any additional equipment on-site; uses could include low-latency network gaming or multi-site corporate networks. In an actual product, this service would likely come with sophisticated matchmaking capabilities; our proof-of-concept connects every subscriber to every other subscriber.

The timings in Table II are from multiple trials where the two endpoint nodes—*Alice* and *Bob*—authenticated against the service, which used FlowOps to create a logical link between them. We see that overall this service can be created in about 300ms. 200ms are used by the security protocol and 90ms are used by FlowOps.

Firewall Service: The firewall service provider is the most complex service we have prototyped. To realize its service, the firewall uses OpenEdge's chaining functionality to insert itself between a subscriber and one of the subscriber's upstream services. When a subscriber authenticates against the firewall service, it instructs FlowOps to relocate that user's service (e.g. ISP service) to its "outside" port. Then, it creates a logical link between the user and a separate "inside" port. It accepts packets on the inside and outside ports, applies firewall rules accordingly, and then passes the traffic to the opposite port. In this way, the service's network becomes part of the subscribers path to the upstream service.

TABLE II
END-TO-END LAN-PARTY SERVICE CREATION (75 TRIALS)

Action	Median (ms)	Std. Dev.
Security Protocol	201.605	79.204
Service Setup	91.529	6.291
Overall	288.535	114.508

TABLE III
END-TO-END FIREWALL SERVICE CREATION (14 TRIALS)

Action	Median (ms)	Std. Dev.
Security Protocol	107.439	10.636
ISP Setup	1119.387	4.727
Relocation	960.632	4.645
Other Service Setup	158.817	0.872
VM Overhead	737.413	6.382
Overall	3204.464	15.700

To realize the firewall we use a host on the lab network running Xen. The firewall service portal (running in domain 0) dynamically creates a new VM for each authenticated user; this VM runs an instance of the Click [11] modular router to realize user-specific firewall functionality.

To evaluate this service, we had the node *Alice* authenticate against the ISP service, then the firewall service, which then performed a relocation. The results of this evaluation are located in Table III. We find that a firewall service can be created (from initial request, to filtering packets) in about 3 seconds. 2 seconds are FlowOps overhead, 740ms are overhead from Xen and Click, and 100ms or so is taken by the security protocol.

B. Service Creation Performance

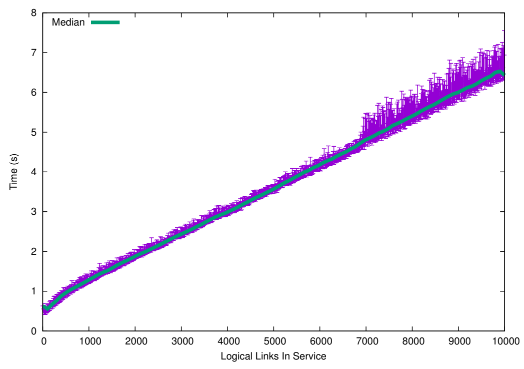

Fig. 6. OpenEdge Service Creation Time

To enable dynamic service creation and ubiquitous service access, FlowOps must quickly render service provider requests into network configurations. Our evaluations of FlowOps showed that this service creation is mostly dependent on the size of the physical edge network and the number of logical links in the requested virtual network.

We evaluate the service creation time of our implementation as a function of the size of the requested virtual network (in number of logical links). A virtual network with 10K logical links (the maximum in our evaluation) would be the size of an ISP on a medium sized edge network that had every user as a subscriber. Evaluating a real, or even emulated, network of this size is beyond our capabilities, so we opt for a simulated test. We evaluated the time taken to push an entire switch configuration to a target Brocade switch using a Ryu OpenFlow driver. This configuration time varied between $\approx 25 - 60$ milliseconds. To simulate this delay in our evaluation, we implemented a dummy driver in FlowOps that sleeps for a random time between $25 - 60$ milliseconds per-configuration request. To aid in scaling to networks of this size, we push the switch configurations in parallel. Since the switch configurations are generated in the previous step, all switches can be configured simultaneously. In our experiments, we assume only 100 switches can be configured in parallel to approximate a realistic scenario.

The network topology we use for this evaluation is a typical star edge network with five core switches, two distribution switch layers, the first with a fanout of 10, and the second with a fanout of twenty edge switches which are connected to each distribution leaf. These edge-switches represent the on-the-premises devices. This translates to 10K edge switches, and 555 non-edge switches, of which 5 are core switches and the remaining 550 are distribution switches.

We perform the evaluation on an Emulab d820 node (2.2Ghz, 8-core 64 bit Xeon “Sandy Bridge”, with 128Gb RAM). We use the `pypy` python interpreter to perform the evaluation. The results for this evaluation are shown in Figure 6. The figure shows the number of seconds (y-axis, in wall clock time) it takes to create a virtual network with the given number of logical links (x-axis) in the median case.

Our results show that FlowOps can easily scale to realistic edge network sizes, and that service creation times are low enough to readily support dynamic and ubiquitous access to services.

C. Security Analysis

We use the threat model from Section II-B1 to guide a security analysis of OpenEdge.

Network Operator and Service Provider Perspectives: Users are required to authenticate using a shared secret before connectivity is granted to service provider-specific network resources. In the case of trying to access SP network resources, the user has to authenticate against both the network operator and the SP via the NOAS. Using traffic separation, we provide strong isolation of network operator and SP network resources. For an unauthorized user, it is as if the SP network resources do not exist.

Subscriber Perspective: To protect against the *passive listening for credentials* scenario, the secret is shared between only the service provider and subscriber. A network operator can authenticate the subscriber while having no knowledge of that shared secret. Rogue NOs similarly cannot access the shared secret. Intermediate nodes are only able to access the challenge (and expected response if the adversary is between the service provider and network operator) of the authentication vector. The use of a TPM will improve the security of stored credentials on the end-device.

OpenEdge protects against the *rogue authentication server* scenario by having the authentication occur over a secure connection (e.g., SSL/TLS). Communication between a user’s V-SIM manager and a rogue authentication server would only occur if the latter was able to produce a certificate that is considered valid by the user’s V-SIM manager.

The *untrustworthy service provider* scenario is prevented via the user-token mechanism. Since user-devices have to initiate the authentication flow with a service provider, we can be sure that an SP can only include user-devices in its setup request that have signaled a desire be part of the SP's service. This prevents a rogue SP from attaching its network to a user device that has not explicitly requested services from the SP.

Using a TPM prevents the *tampering with the end-device* scenario. When the shared secret is imported into the TPM, it is tied to that TPM and cannot be cloned. In order for the shared secret to be moved to another device, it would have to be prepared for TPM import by the SP who issued it.

IV. Related work

Our work is inspired by the open access network (OAN) model, which has been proposed as an alternative to the pervasive monolithic network access approach [5], [6], [7]. Earlier efforts have met with limited success. A key technical limitation of these approaches is the lack of a control framework that would enable automated operation of such networks. As a result OANs offer limited choice, both in terms of the number of services and the types of services available. The OpenEdge cloud-like control architecture addresses these limitations by making it easy to provide services on an edge network. In future work, management of these services could be done through integration with the OpenEdge architecture.

In the commercial world, companies that provide turnkey solutions over high speed fiber networks are emerging. For example, services that target content and digital media companies are being offered in the Vancouver area [1]. With OpenEdge we share the goal of these efforts to provide services that exploit high performance networks. However, unlike these efforts, OpenEdge adopts an open cloud-like approach where services or applications that utilize these networks can be readily provided by third parties.

Our work is also related to software defined networking (SDN) [12], [13] and network function virtualization (NFV) efforts [14], [15]. In particular, efforts related to control architectures for such networks [16], [17]. We use OpenFlow to realize the OpenEdge virtual network abstraction. However, our approach is not fundamentally coupled with OpenFlow (or SDN) and could be realized with other underlying technologies. Our goal is to provide a new edge network service abstraction which eases the realization of edge services and applications. Current NFV efforts are focused on virtualizing network element functionality, while with OpenEdge we use network virtualization to enable new edge service abstractions.

Our work also relates to security frameworks such as OpenID and OAuth. These frameworks require the end user to be able to directly communicate with both of the other parties during the authentication flow. In the case of OpenID, more information is required to be shared between the OpenID provider (our Service Provider) and the client (our Network Operator). In the case of OAuth, the client (our end user) is responsible for all parts of the authentication. These properties make these approaches unsuitable for an open edge environment.

V. Conclusion

In this paper, we presented the OpenEdge architecture to realize a new open service abstraction for edge networks. OpenEdge provides a control architecture which allows edge networks to be controlled in a cloud-like fashion. Specifically, service providers can interact with FlowOps, the network management component of OpenEdge, to dynamically request the creation of network resources to enable their services and applications. OpenEdge also provides a comprehensive security component, SecureOps, which provides fine-grained access control and authentication between end users and both the network operator and service providers. Our evaluation shows that OpenEdge can scale to realistic edge network topologies and effectively mitigate the security concerns for such a multi-player environment.

References

[1] Stratuscore, "Service and Components," http://www.stratuscore.com/services/.

[2] Institute for Local Self-Reliance, "Community Broadband Networks," http://www.muninetworks.org.

[3] M. Manic *et al.*, "Next generation emergency communication systems via software defined networks," in *Research and Educational Experiment Workshop (GREE), 2014 Third GENI*, 2014.

[4] Christopher Mitchell, "Broadband At the Speed of Light: How Three Commmunities Built Next-Generation Networks," http://ilsr.org/wp-content/uploads/2012/04/muni-bb-speed-light.pdf, April 2012.

[5] A. González *et al.*, "Prospects on ftth/ep2p open access models," in *Federation of Telecommunications Engineers of the European Union (FITCE) 49th Congress*, 2010.

[6] W. Lehr *et al.*, "Broadband open access: Lessons from municipal network case studies," in *Proceeding of the TPRC conference*, 2004.

[7] M. Forzati *et al.*, "Open access networks, the swedish experience," in *Transparent Optical Networks (ICTON), 2010 12th International Conference on*. IEEE, 2010, pp. 1–4.

[8] T. Morris, "Trusted platform module," in *Encyclopedia of Cryptography and Security*, H. van Tilborg and S. Jajodia, Eds. Springer US, 2011, pp. 1332–1335. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-5906-5_796

[9] "TPM Library Specification," http://www.trustedcomputinggroup.org/resources/tpm_library_specification.

[10] "3G security; Security architecture," http://www.3gpp.org/DynaReport/33102.htm.

[11] E. Kohler *et al.*, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[12] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[13] N. Feamster *et al.*, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602204.2602219

[14] R. JAIN and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *Communications Magazine, IEEE*, vol. 51, no. 11, pp. 24–31, November 2013.

[15] W. Zhang *et al.*, "SmartSwitch: Blurring the Line Between Network Infrastructure & Cloud Applications," in *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, 2014.

[16] C. Monsanto *et al.*, "Composing software-defined networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482629

[17] N. Gude *et al.*, "Nox: Towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625