A Resource Delegation Framework for Software Defined Networks

Ilya Baldin
RENCI/UNC Chapel Hill
100 Europa Dr., Ste 540
Chapel Hill, NC 27517
ibaldin@renci.org

Shu Huang
RENCI/UNC Chapel Hill
100 Europa Dr., Ste 540
Chapel Hill, NC 27517
shuang@renci.org

Rajesh Gopidi
RENCI/UNC Chapel Hill
100 Europa Dr., Ste 540
Chapel Hill, NC 27517
rajesh@renci.org

ABSTRACT

In this paper we address the problem of multi-domain multi-provider SDN-based networks and propose an architecture for controlling them using a collection of agents responsible for ownership and use of SDN resources. Instead of posing the problem in terms of controller coordination for the purpose of establishing connections across the network, we propose to treat it as a resource-management problem with explicit delegations of consumable resources by domains to the users of those resources. The advantage of our approach is in explicitly exposing the resource delegation abstraction. It exposes the control of network elements in different domains by different controllers and permits generalizing several existing multi-domain architectures, making the selection of which one to apply a deployment choice, rather than an architectural principle. We propose a rigorous algebraic formulation for the SDN resource delegation problem and describe the prototyping work in implementing this framework and some of its applications.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.1 [**Network Architecture and Design**]: Distributed Networks

Keywords

Software-Defined Networking; Multi-Domain Networks; Integer Linear Programming

1. INTRODUCTION

SDN and specifically OpenFlow established itself as a significant player in the datacenter networking, however its deployment in WANs has been limited to a few large providers [6]. Harder still is the problem of using SDN in a multi-domain multi-provider environment, where the problem of offering services across multiple domains is frequently reduced to that of controller placement and coordination [4, 11] or representing a multi-domain network as a single programmable 'big-switch' [10, 12]. While these approaches allow SDN networks to mimic services available in today's routed and transport networks with some extensions, they do not go far enough in exposing the potential advantages that SDN concept provides to network.

Specifically, SDN and OpenFlow explicitly allow the insertion of customer-imposed forwarding policies into the core of the provider network by associating customer controllers with some of the provider network elements and allowing them to manage some portion of the flow space. This level of flexibility provides opportunities for offering novel network services by providers and for supporting network virtualization, with virtual providers offering unique value-add services on top of physical provider infrastructure using their own SDN controllers. Of special interest is *nested virtualization*, where a virtual provider supplies resources to higher-level virtual providers. To enable this novel capability it is critical to create a framework that (a) allows the slicing of the network along multiple resource dimensions, that guarantee the isolation of providers from one another and (b) allows providers to reason about the resources they have or are willing to *delegate* to other providers. Essentially, it is necessary to *separate the ownership of network resources from their use* - the concepts that today are frequently conflated in designing multi-domain SDN architectures.

In this paper we propose a framework that treats offering multi-domain services and virtualization in SDN networks as a *resource-management problem* with explicit delegations of consumable resources by domains to the users of those resources, which can be either (a) SDN controllers belonging to some provider that operate on traffic using those resources, or (b) virtual domains which can create their own sub-delegations of resources to enable nested network virtualization, without operating on them. We present:

- *A rigorous algebraic formulation of the resource delegation problem in SDN.*
- *An architecture of the delegation framework that showcases several advantages of our approach*
- *A GENI prototype implementation to help analyze the design.*

2. BACKGROUND AND MOTIVATION

In order to properly isolate the activities of one provider from the others sharing the same infrastructure, it is important to account for *consumable network resources*. Those

HotSDN'14, August 22, 2014, Chicago, IL, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2989-7/14/08 ...$15.00.
http://dx.doi.org/10.1145/2620728.2620737 .

are *labels* (e.g. IP addresses, VLAN tags, MPLS tags, wavelengths, etc) and parameters related to delivering the proper quality of service (e.g. outgoing bandwidth and buffer space on interfaces). Resource delegations represent *volumes within these multi-dimensional parameter spaces*, which can be tested for intersection and enclosure or subdivided along multiple dimensions to create sub-delegations that serve particular applications with *QoS and resiliency requirements or topological scopes*, for improved scalability.

Our approach explicitly exposes this resource delegation abstraction. In today's routed network a flow traveling across the network can be thought of as taking advantage of a delegation of flow space to the flow owner by the network for the duration of transit. A single flow on a particular interface represents a point in a multi-dimensional flow space constrained by available network labels. With explicit delegation of flow spaces the operator/owner of the network gains control over the issues surrounding the authorization of this delegation: the duration of the delegation becomes an explicit parameter, as does the chain of delegations that allows it to be traced to the original owner using cryptographic mechanisms [5].

Importantly, the approach explicitly exposes any *switching constraints* that an SDN controller must be aware of in order to efficiently move traffic inside a domain; constraints that are typically hidden in e.g. the 'big-switch' approach. By explicitly separating various resource pools it makes it possible to combine within a single architecture the different ways of controlling the network - multiple coordinating controllers belonging to individual domains or multiple controllers that are specific to applications in a single domain or user groups that have control over delegated flow spaces across multiple domains, or even the 'big-switch' approaches mentioned above. It makes the decision which solution to apply a *deployment choice*, rather than an architectural principle.

Our framework can be thought of as a generalization of a number of networking architectures, including (G)MPLS and SDN i.e. FlowVisor[14], ADVisor [13] and VeRTIGO[2]. In particular, VerTIGO allows two extreme views: 1) a virtual network that the user can control; 2) a single abstract node. In the context of Cloud Computing, authors in [8] propose a SDN-based framework to facilitate delegation of some network controls to users. It leverages two instantiation methods, one uses FlowVisor, the other uses ADVisor, and illustrates the tradeoffs between security and the level of network abstractions provided to users.

The work by Chase et al on brokered resource leases [5] can be viewed as a precursor to our architecture. The difference lies in the focus of our work on multi-dimensional delegations of resources and their efficient processing, compared to predominantly one-dimensional delegations and the focus on state machine operations between elements of the architecture, considered in their work.

3. FRAMEWORK DESCRIPTION

Central to our thinking is the concept of a *label*. A single label can be thought of as a sequence of bits within a particular offset in the frame header, while a *label set* describes a collection of values a label can take on. Thus a label set can be an IP subnet or a VLAN range or an enumeration of MAC addresses. Similar to GMPLS, the label concept generalizes to wavelength identifiers and timeslots within TDM transport systems. SDN *network elements* populating a network are assumed to be capable of matching labels to rules, consistent with SDN architecture and acting upon matching flows. Network elements can in some cases perform *label translation* by rewriting labels in the header or changing channels/wavelength.

We also add the concept of a *technology level*, which binds to a particular label offset in the header or type. Labels within technology level A can enclose labels within technology level B, thus making A capable of providing transport to flows distinguished by labels only in B. Technology levels aren't required to map on ISO/OSI stack. For example, within what is commonly known as Layer 2 ethernet, there are several header fields available that can serve as their own technology levels (e.g. VLAN tags and src/dst MAC addresses). They can serve to enclose e.g. Layer 3 packets independently of each other, depending on the matching rules used by network elements. Tunneling packets is a form of an additional enclosing technology level. Based on this ability to carry traffic for each other, we impose a *partial order* relationship on the technology levels formalized below. Our framework does not dictate the existence of specific technology levels.

From the topology standpoint, in our system model we consider the network substrate consisting of *network elements* and links connecting them. Network elements have *uni-directional ports* that terminate the links, a formulation consistent with ITU G.805 and G.809 standards. The network substrate is divided into multiple *domains* owned by different providers, which peer with each other. Resources within domains are managed by *Resource Managers* or RMs. Domain RMs can delegate their resources directly to SDN controllers or to RMs of *virtual domains*. A controller operating within a particular resource delegation is aware of and constrained in the labels and other resources. This means it can operate on flows and create SDN flow rules only based on the labels delegated to it, and it can perform label translations only if they lie within the delegated label spaces. The RM of the virtual provider may choose to operate its own controller or controllers to manage traffic, or subdivide the delegation to create smaller virtual domains, thus enabling nested virtualization of resources in the network.

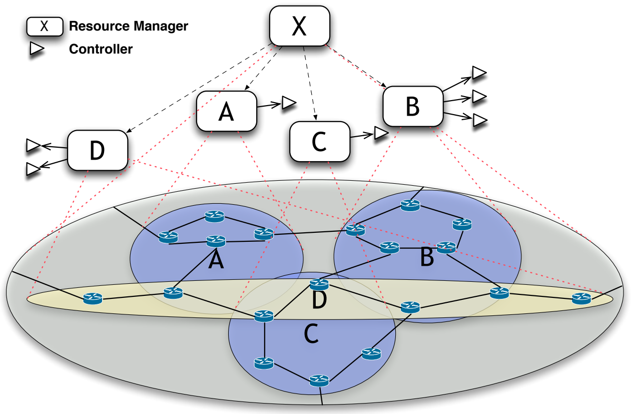

Figure 1: Network domains, resource managers and controllers

Figure 1 demonstrates a simple example of a single network domain with several enclosed scope-based virtual domains based on topology (A, B and C) and a single application-specific virtual domain D, created e.g. to support *fast transit services across the parent domain*. The top-level RM in X is responsible for all resources in the domain. X subdelegates resources to A, B, C and D, varying the allocation over time based on e.g. load and flow statistics or direct requests from resource managers in other domains.

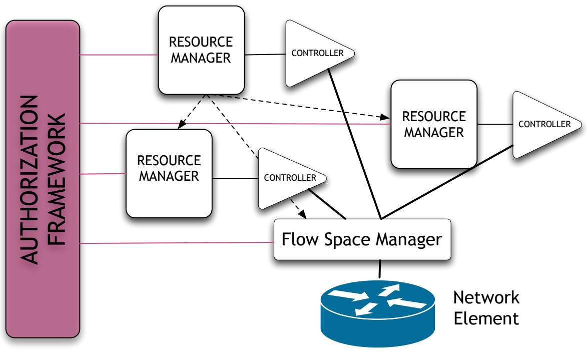

Figure 2: Communicating and enforcing delegations

Conceptually, the framework requires several elements illustrated in Figure 2: RMs representing *resource ownership*, embodying policy decisions about resource delegations and controllers representing *resource use*, embodying the network policy control over the delegated resources via flow rules. One more element must act as an *enforcement mechanism* acting on behalf of the owner of a network element to ensure resource delegations are respected, that we call *Flow Space Manager* or FSM. It is a Policy Enforcement Point (PEP [3]) similar to the various FlowVisor-like slicing mechanisms we mentioned. It receives resource delegations, verifies their validity and monitors the conformance of flow rules inserted by the various controllers in accordance to those. Importantly, all elements must be tied together by an *authorization framework* that creates trust between them and allows for specification of authorization policy restrictions on delegations, i.e. terms, intended uses, ability to subdelegate and so on. One example of such framework is Attribute-Based Access Control or ABAC [7].

4. MATHEMATICAL MODEL

In this section we present the notation for describing delegations and the formulation of an optimization problem based on it. The network has the following resources:

$\boldsymbol{T}$: a set of technology levels, $\{t \in T\}$

$\preceq$: a partial order on T. If $t_i \preceq t_j$, this implies that t_i can provide connectivity for t_j.

$\boldsymbol{INP^t}$ $(\boldsymbol{OUTP^t})$: a set of input (output) ports on network elements associated with a particular technology level, $\{inp^t \in INP^t\}(\{outp^t \in OUTP^t\})$.

$\boldsymbol{N}$: a set of network elements

$\boldsymbol{NE(p)}$: a mapping from a network-unique port identifier to a specific network element. $NE(p) : INP^t \cup OUTP^t \to N$.

$\boldsymbol{L^t}$: a set of mono-typed labels allowed within a technology level. [1]

[1]For convenience, if a particular network layer offers several address schemes (e.g. VLANs and MAC addresses at Ethernet layer), they are considered two different levels L^{MAC} and L^{VLAN}.

$\boldsymbol{Tr^t(p)}$: a 0-1 function defining whether label translation is allowed on a particular port: $Tr^t(p) : \{INP^t \cup OUTP^t\} \to 0 \cup 1$.

$\boldsymbol{B^t(p)}$: a function defining available buffer space on a port $B^t(p) : INP^t \cup OUTP^t \to \mathbb{N}$

$\boldsymbol{M_{l,m}^{t,\{a,b\}}}$: a 0-1 variable defining whether label l can be transmitted on port a and received as m on port b: $M_{l,m}^{t,\{a,b\}} a \in INP^t, b \in OUTP^t, l, m \in L^t$. It encodes the delegated topology and can be transformed into e.g. an adjacency matrix or function.

$\boldsymbol{W^t(a, \lambda)}$: a function defining maximum transmit bandwidth for a set of labels λ on a given port a: $W(a, \lambda) : OUTP^t \times \{L^t\} \to \mathbb{N}$

$\boldsymbol{F^t(n)}$: a function describing resources available for installing forwarding/stitching rules on a particular network element e: $F^t(n) : N \to \mathbb{N}$.

A delegation given to a resource manager in a particular virtual domain or a controller represents a portion of network resources over which its owner exerts policy controls (e.g. packet forwarding or sub-delegation) described in terms of these parameters. The first delegation is created by the owner of the resources subject to the types of services intended to be supported within this delegation.

More formally, a *resource delegation* to a virtual domain c within technology level t is a tuple:
$D_c^t =< INP_c^t, OUTP_c^t, Tr_c^t, B_c^t, M_c^t, W_c^t, F_c^t >$.

Without the loss of generality we assume that network elements and ports are not renumbered in the delegation.

A delegation may be associated with meta information describing its *duration and authorized uses*, it may be cryptographically signed. As we mentioned, the resource delegations can be nested, such that RM c can sub-delegate the volume of its resources to an RM d. In order to produce a delegation for a customer, the owner of resources must map D_d^t onto D_c^t according to some optimization goal while satisfying several constraints. Figure 3 helps illustrate the problem.

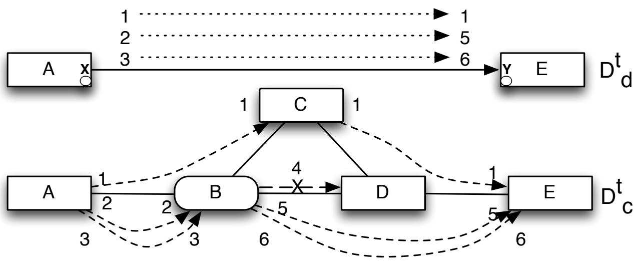

Figure 3: Satisfying path continuity

The top figure shows a delegated link between ports X and Y on nodes A and E, where the set of allowed transmit labels is $\{1, 2, 3\}$ is routed by two different paths to labels $\{1, 5, 6\}$ s.t. $M_{1,1}^{x,y} = 1$, $M_{2,5}^{x,y} = 1$ and $M_{3,6}^{x,y} = 1$. The lower part of the figure shows the topology from which this delegation is derived, which consists of nodes A, B, C, D, E. Note that label 1 is routed via node C without translation, because e.g. C is incapable of label translation or because label 1 was available. On the other hand, labels 2 and 3 are routed via nodes B and D with intermediate translation. As delegations come and go over time, we would expect the fragmentation of label space to occur, requiring translations in order to achieve required connectivity.

We say that delegation D_d^t is a *feasible subdelegation* of D_c^t, or, formally $D_d^t \subseteq D_c^t$ as long as it satisfies a number of rules:

$$(1)\begin{cases} INP_d^t \subseteq INP_c^t \\ OUTP_d^t \subseteq OUTP_d^t \\ Tr_d^t(p) \leq Tr_c^t(p) & \forall p \in INP_d^t \cup OUTP_d^t \\ B_d^t(p) \leq B_c^t(p) & \forall p \in INP_d^t \cup OUTP_d^t \\ F_d^t(n) \leq F_c^t(n) & \forall n \in N \end{cases}$$

Ruleset (1) preserves resource *volume inclusion*, ensuring the delegated resources do not exceed those owned by the producer of the delegation.

Additionally a feasible subdelegation must satisfy several conditions related to label continuity and label and bandwidth accounting between the original and delegated topologies, which makes the mapping from D_d^t to D_c^t *homeomorphic*, i.e. label paths in the former become links in the latter, which we formulate as an ILP below. There are two flavors of this routing problem: 1) label path splitting is allowed, as shown in Figure 3; 2) path splitting is not allowed. The routing becomes simpler if splitting is not allowed, however it is inefficient in using the D_c^t resources, similar to flow-splitting in routing problems. For brevity we formulate only the former.

In order to guarantee the conservation of labels in the delegation, we create an ILP formulation using the binary variable $k_{x,y,l,m}^{i,j}$ defined as follows: it indicates if an arc connecting ports i and j ($i \in OUTP^t$, $j \in INP^t$) in D_d^t uses the arc from $x \in OUTP^t$ to $y \in INP^t$ in D_c^t and expects label l at x and m at y. We list the ILP rules below, using superscript (c) or (d) whenever we need to refer to the original or the delegated topology. So $\forall a \in OUTP^t, b \in INP^t$ in the D_d^t:

$$\begin{cases} M_{l,m}^{a,b(d)} = \sum\limits_{n\in L^t} \sum\limits_{p \in INP^t} k_{a,p,l,n}^{a,b} \forall l,m \in L^t & (2) \\ M_{l,m}^{a,b(d)} = \sum\limits_{n\in L^t} \sum\limits_{p \in OUTP^t} k_{p,b,n,m}^{a,b} \forall l,m \in L^t & (3) \\ \sum\limits_{l,m\in L^t} k_{w,x,l,m}^{a,b} = \sum\limits_{m,n\in L^t} k_{y,z,m,n}^{a,b} & (4) \\ \forall w,y \in OUTP^t x,z \in INP^t \text{ if } NE(x) = NE(y) \\ \text{ and both } Tr(x) = 0 \text{ and } Tr(y) = 0 \\ \sum\limits_{w}\sum\limits_{x}\sum\limits_{l,m\in L^t} k_{x,y,l,m}^{a,b} = \sum\limits_{y}\sum\limits_{z}\sum\limits_{n,o\in L^t} k_{y,z,n,o}^{a,b} & (4') \\ \forall w,y \in OUTP^t x,z \in INP^t \text{ if } NE(x) = NE(y) \\ \text{ and either } Tr(x) = 1 \text{ or } Tr(y) = 1 \\ \sum\limits_{a,b} k_{x,y,l,m}^{a,b} \leq M_{l,m}^{x,y(c)} & (5) \end{cases}$$

Constraint (2) says that the use of a label at the head port of arc a, b in the delegation must come from one of the label paths in the original topology. Constraint (3) is similar for the tail of the same arc. Constraints (4) and (4') speak to the conservation of labels in the intermediate links offered in the original topology to support arc a, b in the delegation, with the former defined for the case when the specific ports offer tag translation, and the latter - not. Finally, constraint (5) simply states that labels on an arc must be owned before it can be used in a delegation. For brevity, we omit the similar treatment of bandwidth constraints in the delegation, which in some cases can be left unspecified if bandwidth provisioning is not supported by the network elements.

Thus the owner of resources may compute D_d^t for some customer based on the above formulation and by adding additional objective functions. In particular, it could be desirable to minimize the resource usage for a given D_d^t and D_c^t, i.e.,

$$\min \sum_{x,y} \sum_{i,j} \sum_{l,m} k_{x,y,l,m}^{i,j}$$

5. PROTOTYPE

Woe have implemented a prototype that consists of two parts: a delegation framework that allows creating SDN resource delegations *consistent with the model we described* and a network application that takes advantage of the delegations. The prototype was implemented on ExoGENI [1] testbed using a topology of OVS-based switches.

The delegation framework consists of a GUI application that allows creating SDN resource delegations and saving them in GraphML DEX graph database format [9] and a Floodlight controller module capable of ingesting these delegations to create virtual topologies. We also used FlowVisor in place of FSM for prototyping purposes only, although it does not efficiently handle flow volumes, rather enumerations of point flows and hyperplanes (with wildcards). The prototype does not include an authorization framework.

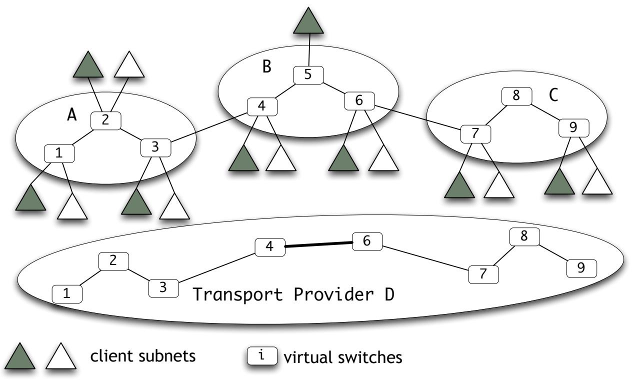

Figure 4: Prototype topology.

Figure 4 shows the topology of the prototype application that uses the delegation framework. It has 3 provider domains (A, B and C) with clients attached to them. A *virtual transit provider* D is created using delegations of resources from A, B and C. The delegation contains labels of two types - IP subnets for serving L3 customers and portions of destination MAC address field. Provider D uses its own Floodlight controller to control some of the switches in the provider domains, subject to the delegated flowspaces, to create a *virtual transport service* between clients (shown white) connected to these domains by utilizing the delegated MAC address header space to support *path-based transport functionality* with path IDs written in place of the destination MAC address, while the frames transit through it. MAC address rewriting is performed at ingress and egress from D. At the same time controllers belonging to domains A, B and C support their own transport service only between clients connected to their own domains (shown shaded). Using the same controller logic in all domains was done for simplicity, however controllers belonging to the individual providers are free to use their delegated header spaces and other resources as they wish.

Notably, by resource delegations, D could insert flow rules into all switches except switch 5 in provider B, with provider B instead choosing to expose a virtual link between switches 4 and 6. In addition, this virtual link included label translation performed by switch 5 as instructed by B's own controller, because B refused for policy or resource scarcity reasons to expose the same IP address space as part of its delegation on both ends of this link, unlike other providers on other links. Other providers supported the full range of IP addresses of clients of provider D. Controller D had to adapt to this situation, by introducing its own translation via additional flow rules in switches it could control. *Explicitly exposing this constraint* as part of B's delegation to D made this adaptation possible.

To support transport functionality we used a modified Floodlight with the following specialized modules:

Topology delegation module, which accepts the resource delegation expressed in GraphML and forms the *initial notion of the delegated topology in the controller*.

Topology verification module, which validates that its notion of the delegated topology is accurate, i.e. that label continuity across delegated virtual links is assured according to the rules described in the previous section. The validation is performed by *probing*, with the controller inserting specially marked packets matching randomly sampled points from the delegated flowspace at one end of the link and verifying their transit on the other end. This is similar to the LLDP-based method typically used by SDN controllers, however we must note that LLDP header represents *its own flowspace* and cannot be used unmodified to probe a flowspace delegated to a controller. Since we are dealing with virtual links, the successful transit of label X delegated on a particular link, does not mean that label Y will also transit successfully, due to possible misconfigurations of intermediate equipment. Our verification module uses *stochastic probing* of the delegated flowspace to ensure that the delegation is valid and only after it is satisfied, the controller topology module becomes aware of a particular link.

ARP resolution module, which listens for client ARP requests and creates proper responses to them so packets intended from clients of one domain to the other can enter the virtual transport provider network. It is also responsible for rules rewriting MAC headers into transport path IDs as packets enter the virtual provider and replacing them with actual destination MAC ids when packets egress the provider.

Circuit module responsible for computing shortest path circuits between domains, globally assigning them path IDs.

Besides proving that the prototype functioned as expected we also benchmarked our topology acquisition module against the traditional LLDP method, as shown in Figure 5. We used several topologies with N nodes and L links as shown on the horizontal axis. LLDP provides the shortest times. Our stochastic method, using random sampling of the delegated flow space is only marginally slower. The process, however, is dominated by the processing of the GraphML-based descriptions of resource delegations at startup. While the penalty is significant, we believe it can be improved upon

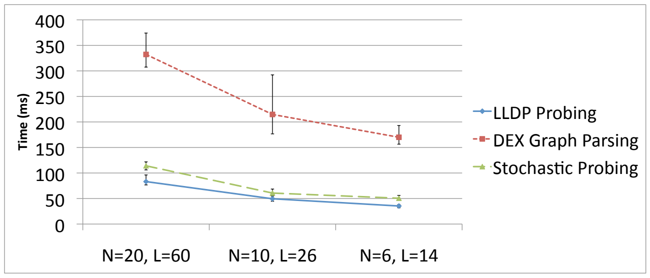

Figure 5: Comparison of probing times.

by using more efficient forms of representations. Also, this penalty is amortized over the time a particular resource delegation exists, which can be significant (minutes, hours or days), since we expect the process of revising a delegation to be primarily driven by surges in demand, the need to introduce new services and other relatively infrequent events.

6. CONCLUSIONS

We believe our approach shows several important use cases critical to deploying SDN in multi-domain environment:

Provide explicit direct control over provider equipment with verifiable constraints: when proper control mechanisms are in place, like the authorization framework and the FSM module, this provides both customers and providers with an environment where customers can operate in a manner that is enforceable and verifiable by the original resource owner.

Explicit constraints to controllers: our framework permits explicit communication of complex switch constraints that involve explicit and implicit label translations that allows customer controllers to flexibly adapt their policies to the resources represented in the delegated environment.

Efficient use of label spaces: by explicitly allowing for label translation as part of delegation, both as an available function within some of the delegated network elements and as a property of some of the delegated links, our framework allows for very efficient policies that allocate labels for different uses. This deals with the unavoidable fragmentation of label spaces owned by providers after multiple cycles of delegation requests and delegation expirations/returns.

Dynamic resource allocation: SDN resource delegations can be short- or long-lived, depending on the strategy of the owner. By setting delegation terms shorter, the resource owner may force more frequent reconsideration and reallocation of delegated resources, thus optimizing allocation for more short-lived traffic phenomena in its network.

Ability to combine multiple approaches in one architecture: our framework does not *require* that controllers in multiple domains communicate with each other to coordinate bandwidth allocation or consolidate into a single 'big-switch' to offer inter-domain services. Instead a multitude of solutions is possible: a single virtual provider may be created across multiple domains, or multiple controllers owned by the same domain can coexist and support different services within different delegations. The particular approach to coordination of RMs or controllers is left as a *deployment*, rather than an architectural choice.

Nested virtualization: our framework supports nested virtualizations by providing sets of constraints expressed as an ILP formulation. Each new delegation can be further subdivided subject to controls of the authorization framework.

Ability to introduce an economy: by defining explicit and termed delegations of resources, our approach permits the introduction of an economy into the network, where delegations may be exchanged for *consideration*, with the possibility of creating marketplaces and auctions, as described in [15], due to the detailed nature of the resource definitions and a clear separation between *resource ownership* and *resource use*.

Our future work focuses on defining the language to express delegation requests and integrating an authorization framework into the prototype.

7. ACKNOWLEDGMENTS

This work is supported by the US Department of Energy grant DE-SC0007425 and by the US National Science Foundation under grant No. CNS-1111256.

The authors would also like to thank Jeff Chase and Nicholas Bastin for valuable input and discussions of the subjects contained in the paper.

8. REFERENCES

[1] I. Baldine, Y. Xin, A. Mandal, P. Ruth, A. Yumerefendi, and J. Chase. ExoGENI: A Multi-Domain Infrastructure-as-a-Service Testbed. In *TridentCom: International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, June 2012.

[2] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori. VeRTIGO: Network Virtualization and Beyond. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 24–29. IEEE, 2012.

[3] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. *RFC 2748*, 2000.

[4] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 7–12. ACM, 2012.

[5] D. Irwin, J. S. Chase, L. Grit, A. Yumerefendi, D. d Becker, and K. G. Yocum. Sharing Networked Resources with Brokered Leases. In *Proceedings of the USENIX Technical Conference*, June 2006.

[6] T. Koponen, M. Casado, N. Gude, J. Stribling, L. e. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix : A Distributed Control Platform for Large-scale Production Networks. *USENIX OSDI*, 2010.

[7] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a Role-Based Trust-Management Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2002. IEEE Computer Society.

[8] M. S. Malik, M. Montanari, J. H. Huh, R. B. Bobba, and R. H. Campbell. Towards SDN enabled network control delegation in clouds. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–6. IEEE, 2013.

[9] N. Martinez-Bazan, S. Gomez-Villamor, and F. Escale-Claveras. Dex: A high-performance graph database management system. In *Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference on*, pages 124–127. IEEE, 2011.

[10] J. McCauley, A. Panda, M. Casado, T. Koponen, S. Shenker, and U. C. Berkeley. Extending SDN to Large-Scale Networks. *Open Networking Summit*, pages 1–2, 2013.

[11] K. Phemius, M. Bouet, and J. Leguay. DISCO: Distributed Multi-domain SDN Controllers. *CoRR*, abs/1308.6138, 2013.

[12] A. Sadasivarao, A. Lake, C. Liou, S. Syed, P. Pan, C. Guok, and I. Monga. Open Transport Switch - A Software Defined Networking Architecture for Transport Networks. *HotSDN'13*, pages 115–120, 2013.

[13] E. Salvadori, R. D. Corin, A. Broglio, and M. Gerola. Generalizing virtual network topologies in OpenFlow-based networks. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011.

[14] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[15] T. Wolf, J. Griffioen, K. L. Calvert, R. Dutta, G. N. Rouskas, I. Baldine, and A. Nagurney. Choice As a Principle in Network Architecture. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 105–106, New York, NY, USA, 2012. ACM.