Simplifying Manageability, Scalability and Host Mobility in Large-Scale Enterprise Networks using VEIL-click

Sourabh Jain, Zhi-Li Zhang
University of Minnesota-Twin Cities
{sourj, zhzhang}@cs.umn.edu

ABSTRACT

The explosive growth in the network driven services and devices is causing existing networks to continually expand to accommodate the demands set by them. However, underlying network architecture can not sustain this continual expansion. As a result, several ad-hoc mechanisms are used as workarounds, which make the networks increasingly complicated and difficult to manage. In this paper, we present *veil-click*, which is aimed at simplifying the management of large-scale enterprise networks by requiring minimal manual configuration overheads. It makes it tremendously easy to plug-in a new routing-node or a host-device in the network without requiring any manual configuration. It builds on top of a highly scalable and robust routing substrate provided by VIRO, and supports many advanced features such as seamless mobility support, built-in multi-path routing and fast-failure re-routing in case of link/node failures. Our current prototype of *veil-click* is built using Click Modular Router framework, and is being deployed in our lab for the evaluation under the real traffic conditions.

1. INTRODUCTION

While Internet driven applications and devices are growing at a rapid pace, they are continuously stressing the underlying network architecture to meet the demands, which are limited by the scalability of the underlying network technologies. These trends are posing daunting challenges to network designers and operators, who often resort to complicated workarounds to expand their networks and accommodate the demand, which is usually achieved by paying a huge cost in terms of the additional manual overhead in managing and maintaining the infrastructure.

There are several reasons for this stress. On one hand, layer-2 network technologies such as Ethernet are largely "plug-&-play" and require minimal manual configuration. However, they can not scale to create large and dynamic (layer-2) networks such as, Metro Ethernets, huge enterprise networks, single ISP network, since they rely on "flooding" based packet forwarding and address resolutions. On the other hand, Internet Protocol (IP layer) which acts as a "glue" to interconnect several layer-2 networks, require careful and extensive network configurations including the IP address assignments to routers and end-host devices which complicates the network management tasks. They offer relatively poor support for mobility and have limited scalability for the underlying routing protocols due to the state-size explosion and so forth.

To address these challenges we propose a network architecture, which not only simplifies the management for large-scale enterprise networks, but also improves the scalability and host mobility of the network to incorporate a large number of "routing-nodes"(e.g. routers and switches) and "host" devices. We build our new networking architecture using a recently proposed VIRO routing framework [9], which is a *paradigm-shifting* approach to network routing and forwarding that is highly scalable, robust and efficient.

In this paper, we describe *veil-click*— a realization of VIRO routing framework, which is tailored towards creating large-scale advanced layer-2 networks. There are several mechanisms that we employ to achieve this. First, we introduce a centralized controller for the network, which collects the network topology and performs the *vid* assignment for the routing nodes in the network. Second, we choose 48-bit long *vid* for both *host-devices* and as well as the routing nodes, so that, we can use the existing Ethernet semantics, such as ARP protocol for name resolutions and so on. Third, we do not introduce any new additional networking header for the basic packet forwarding. This is achieved by re-using Ethernet address fields in the layer-2 headers.

In order to demonstrate the practical feasibility of our routing architecture, we implement the key components of our proposed networking architecture using an initial prototype based on Click Modular Router framework [12], which is not only *completely backward compatible* to work along with the legacy hardware (e.g., Ethernet based switches) and software (e.g., host networking protocols such as ARP, IP based addressing etc.), but also leverages the VIRO routing framework to provide built-in mechanisms for load-balancing, fast rerouting, seamless mobility support and other key features needed to support future networking and application needs. Our initial evaluation using the Click based prototype switch shows that *veil-click* can support seamless mobility support for the *host-devices*, and it does not interrupt the on-going TCP connections for the hosts during the mobility. In addition, our experiments showed that TCP connections for the *host-devices* take only around 2-5 seconds to recover during the *host-device* mobility.

The remainder of the paper is organized as follows. We provide an overview of VIRO, the key ideas behind *veil-click* and related work in §2. §3 provides the description of basic components, §4 presents the basic design of our Click based prototype and in §5 we describe the advanced features. Finally we conclude the paper in §6. (Due to the space limitation, we could not include our simulation based evaluation results in this paper, which can be found in the VIRO tech-report [10]. The evaluation using the Click based prototype is currently in progress, while the source code for the *veil-click* can be downloaded from the Google Code repository [3].)

2. OVERVIEW

2.1 Motivation

The design of *veil-click* is motivated by the following problems in the existing networking technologies.

• **Requirement of extensive network configuration & address management.** Current IP networks require careful (and often manual) configurations and management. The need for *address management* is particularly cumbersome and problematic: while an end host joining a network can dynamically obtain its IP address via DHCP, adding a router or subnet to expand an existing network often requires allocation of one or more new IP address blocks. This is because IP address management is *link-based*: each link – either via a point-to-point connection or wired/wireless broadcast media – (or each subnet) must be assigned a distinct IP address block. Such address blocks must then be configured into routers, and injected into intra-domain routing protocols. Similarly, the assignment and manual configuration of OSPF areas for IP based intra-domain routing adds to the complexity.

• **Limited ability to exploit the richness in the network topologies.** IP routing within a single network domain (i.e., intra-domain routing) also suffers several major problems. These problems have their root in the *shortest-path* based routing paradigm used in IP (intra-domain) routing. The use of shortest-paths limits the ability of IP networks to exploit path diversity inherent in the network topology to perform load-balancing and fast-rerouting of traffic under failures. To perform load-balancing and traffic engineering, one has to resort to sub-optimal work-arounds or fixes, e.g., via IGP weight optimization. Likewise, various IP-based fast rerouting mechanisms have been proposed, which partly circumvent the slow convergence problem plaguing the traditional *reactive* IP routing protocols (e.g., OSPF or IS-IS). Unfortunately most of these fast rerouting mechanisms are fairly complex, and as "add-ons" to existing routing protocols require additional configurations, which further complicates the operations of IP networks.

• **Poor scalability of existing Ethernet based layer-2 networking protocols.** Unlike IP networks, layer-2 networks such as Ethernet are largely *plug-&-play*: hosts are equipped with persistent MAC addresses, and Ethernet switches automatically learn about host addresses and location, and perform packet forwarding seamlessly with minimal operator configuration and intervention. On the other hand, as it was originally developed for small, local area networks, it relies on *network-wide flooding* for packet forwarding and address resolution, which severely limits its scalability and efficiency.

• **No support for the host mobility.** Current networking protocols consider the IP address of a host as a proxy for the node's identity. While IP address is also used to route the packet to the destination, therefore, it acts as an address for the node as well. When a node moves from one subnet to another subnet in the network, its IP address has to be reconfigured either using the DHCP or by performing static manual configuration. This dual use of IP address as an identity and as well as an address, causes the reseting of the existing network connection between the hosts whenever their IP address is changed due to the mobility.

2.2 Related Work

To address the challenges faced by traditional layer-2 routing protocols, several solutions [1, 2, 11, 15] have been proposed, of which, SEATTLE [11] is closest in spirit to our work, in that, both utilize DHT (distributed hash table) techniques for scalable and efficient address look-up and resolution. However, SEATTLE employs the OSPF-style shortest path routing in layer 2. It therefore not only requires network-wide flooding in the control plane for building routing tables, but also suffers from the same scalability and robustness limitations plaguing shortest-path routing: for example, it is limited to the use of shortest paths only; load-balancing and fast rerouting can be complicated to implement. In contrast, VIRO routing framework used in *veil-click* avoids these inherent problems in shortest-path routing. It is far more scalable and robust (e.g., with $O(\log N)$ routing table sizes instead of $O(N)$ in OSPF). Similarly, to circumvent these problems in a data-center environment, several "customer-made" networking solutions have been proposed, see, e.g., [4, 7]. However, these solution are tied to a specific network topology, and therefore, can not work for any general network topology, which is mostly the case for large-scale enterprise networks. Our work is also substantially different from the "flat-id" based routing schemes such as UIP [6] and ROFL [5], which advocate a *flat* universal *id* space to replace the current global IP address space. These schemes employ a DHT-style randomly and consistently hashed *id* assignment–which produces an *id*-space completely independent of the underlying network topology–and perform routing based on logical distance to the *id* of the destination, incurring a stretch penalty (which is unbounded in the worst case). We circumvent these problems by introducing a *topology-aware* structured *vid* space. It incurs fairly small routing stretches, and effectively localizes the effect of failures. In addition, our previous works [8–10] provide a detailed qualitative and quantitative comparison with these related works.

In particular, VEIL addresses the same scalability and mobility related issues for large-scale layer-2 networks as addressed by other proposals such as SEATTLE [11], VL2 [7] and TRILL [2]. These proposals use link-state style routing for better data-plane scalability in layer-2 networks, however, control plane still suffers from the scalability concerns for a very large network. They also deploy Loc/ID split as proposed in LISP [14] to support the host mobility. However, they use different mechanisms to achieve this. E.g., SEATTLE uses a DHT based resolution mechanism, which is done by the switches in the network on behalf of host-devices. This adds additional overhead on the switches to perform the explicit resolutions. On the other hand, VL2 requires the hosts to perform the resolutions directly, by installing a special network driver on the host machines. In case of VEIL, it leverages the existing ARP protocol to perform the name resolutions without modifying anything on the *host-devices*, therefore, it provides more scalable and completely backward compatible solution.

2.3 VEIL-click: Design Overview

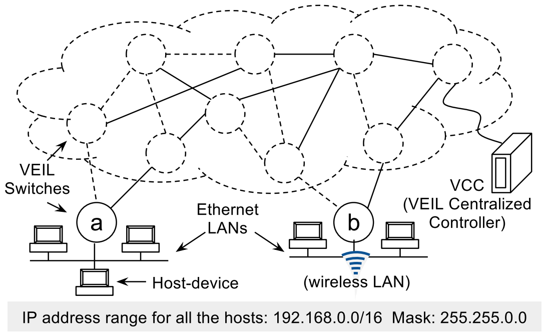


Figure 1: An example showing large-scale layer-2 network using VEIL-Click.

veil-click is a practical realization of VIRO using *Virtual Ethernet Id Layer* (in short, VEIL) prototyped using Click Modular Router framework. Figure 1 shows an example of a large-scale layer-2 network created using *veil-click* based switches. As seen in this figure, all the host-device that connect to this network can be assigned an IP address using a single IP address block. Therefore, it avoids the partitioning of the network into multiple subnets as performed in traditional layer-2/layer-3 networks. It enables the seamless mobility support for the *host-devices*, since they do not need to change their IP addresses, when they move within the network. As seen in Figure 1, the network consists of several *veil-switches* shown using circles, a VEIL centralized controller (in short *vcc*), and a large number of *host-devices* which can either directly connect to *veil-switches* or through Ethernet based Wired/Wireless LAN switches. The roles of each of these devices are as follows.

vcc. It bootstraps the network by performing the initial *vid*-assignment to *veil-switches*. In addition, it also assigns a *vid* to a new *veil-switch* that joins the network after the initial bootstrapping process. Although, VEIL uses a centralized *vid*-assignment for the *veil-switches*, it does not create a single point of failure. This is because once the *vid*-assignment to *veil-switches* is done, they do not need *vcc* for the packet forwarding and other related tasks. Moreover, it is possible to have additional *vcc* connected to the network for redundancy.

veil-switch. A *veil-switch* is the Click based prototype switch, which performs all the actions required to perform the routing and forwarding among the *veil-switches* using VIRO routing protocol. It also performs the *vid*-assignments for the *host-devices* connecting to it directly (or through Ethernet switches).

host-device. It represents any end-host device that connects to the network such as desktop/laptop computers, smartphones etc.

There are three key components of *veil-click*, namely, *vid*-assignment, routing and forwarding, and address resolution. *vid*-assignment to *veil-switches* is done by the *vcc*. To achieve this *veil-switches* send their neighbor information to the *vcc*, which computes the *vid* for the switch based upon its location in the topology. While, *host-devices* are assigned 48-bit long *vid* by the *veil-switch* they directly connect to. Therefore, no manual configuration is required to connect a *host-device* or a *veil-switch* to the network. The routing among the *veil-switches* is performed using the *vids* except at the edges where packet is forwarded to/from a *host-device*. Since we use a host-agnostic design, *veil-click* does not require any modifications to *host-devices* to communicate with each other using *veil-switches*, we intercept and use ARP packets sent by the hosts to resolve the IP addresses to *vids* instead of mapping them to the actual MAC addresses. The only exception to such handling of ARP packets is the case when two hosts are connected to the same *veil-switch* at the same interface, in which case, they directly use MAC addresses to communicate with each other and no intermediate *veil-switches* are involved in the communication. When a *host-device* moves in the network and connects to a different *host-switch*[1], its IP address may remain the same, and its updated *vid* is pushed to other hosts that it is talking to as an ARP reply packet. Therefore, the mobility does not interrupt the network connection for the host.

3. VEIL-CLICK: KEY COMPONENTS

veil-click consists of three essential components, namely, *vid*-management, routing and forwarding, and *host-device* namespace management. In this section we provide a brief description of each of these components.

• **vid Management.** In *veil-click* the *vid*-assignment for the *veil-switches* is performed by the *vcc*. To achieve this, an 'in-band' communication channel to communicate with *vcc* is required. In the following we first describe the protocol used by *veil-switches* to construct and maintain the spanning tree used to communicate with *vcc*, and then the algorithm used by *vcc* to perform the *vid*-management.

vcc Communication Protocol. This protocol consists of two key operations: i. broadcast of the current "best" path to reach *vcc* to directly connected physical neighbors, and ii. subscription to one of the physical neighbors which advertises the "best" path. If a *veil-switch* has an outgoing interface that connects to *vcc*, it announces this information to all its other neighbors by advertising itself as an immediate upstream node to reach *vcc* and its distance to the *vcc* (as number of hops) is 1. Similarly, *vcc* advertises the distance 0 to *veil-switches* directly connected to it. Whenever, a *veil-switch* receives this advertisement it compares the advertised distance with the distance advertised by its current upstream node to reach the *vcc*. If the advertised distance is smaller then it installs the advertising node as its new upstream node to reach *vcc*, and sends a "subscription" packet to the node indicating that it is the downstream node for the node. When a node receives a "subscription" packet from one of its neighbor nodes, it installs that node as one of the downstream nodes. A node uses the "upstream node" to forward the packet to the *vcc*, on the other hand downstream nodes are used to forward the packets coming from *vcc* to all the nodes in the network. We show this communication channel for the example network in Figure 1 using the continuous black lines.

vid-assignment. When a node learns its upstream node to reach *vcc*, it publishes its neighbor information to *vcc*. Upon receiving the neighbor information from a node, *vcc* stores it in its local database to construct the complete topology of the network, which is then used to perform the *vid*-assignment to *veil-switches* using the top-down graph partitioning based approach [10]. In this process each *veil-switch* is assigned a 32-bit long *vid*. When a new *veil-switch* joins the network after initial *vid*-assignment process, it first learns the upstream node to reach the *vcc* from its neighbors, and sends it neighbor information to *vcc*, which then assigns a *vid* to this node based upon its neighbor's *vid*. On the other hand when a *host-device* connects to a *veil-switch*, the *veil-switch* detects it by sniffing on the data packets sent by the device. Whenever it detects a new host IP address (*ip*) it assigns a unique 48-bit long *hostvid* to it by appending unique 16-bits to its 32-bit long switch-*vid* and pushes the mapping (*ip*, *hostvid*) to the *access-switch*[2] corresponding to IP address *ip*.

• **Routing & Forwarding.** After the initial *vid*-assignment is performed by the *vcc*, each node periodically runs the bottom-up routing table construction process as described in [9]. Since, *vids* are topology-aware, each node summarizes the routing entry to reach other nodes using 32 unique *vid* prefixes, and therefore only need to store a maximum of 32 routing entries[3]. In order to forward a packet to a given destination, a *veil-switch* uses the routing entry corresponding to the longest matching *vid* prefix. In addition, the implementation of VIRO in *veil-click*, also extends the basic routing protocol to enable the multi-path routing and as well as fast-failure re-routing by having multiple routing entries at each *vid* prefix level. However, as mentioned in VIRO [9], it needs to be done carefully. Otherwise, it may cause the forwarding loops in the data plane if different *veil-switches* on the path choose conflicting forwarding entries. To avoid these loops, *veil-switches*, also include a forwarding identifier on the packet in the form of a shim-layer between the layer-2 and layer-3 headers. It ensures that all the nodes on the path choose a consistent forwarding entry for a given data packet.

Whenever, a *host-device* sends a packet to the network, *host-switch* detects it by looking at the source address in the Ethernet header and overwrites the source MAC address by the *vid* for the host, before forwarding it to other *veil-switches*. Similarly, if a switch receives a packet which has the destination MAC address as one of its *host-devices' vid*, it overwrites the destination address in the Ethernet header by the actual MAC address of the host before forwarding the packet to the host.

[1]A *host-switch* for a *host-device* is the *veil-switch* that it is directly connected to.

[2]An access-switch for a host is the *veil-switch* whose *vid* is closest to the 32-bit long hash value of its IP address. Here, closeness is measured using the XOR distance.

[3]Number of routing entries in the routing table of a node in VIRO is same as the number of bits used to represent the *vids*. Since we use a 32-bit long *vid* strings in *veil-click* for the *veil-switches*, hence a maximum of 32 routing entries in the routing table.

• **Host-device Namespace Management.** *veil-switches* detect the *host-devices* directly connected to them by sniffing on the packets sent by them. Whenever, they discover a new *host-device*, they assign a unique *vid* to them, and push the (*vid*, *ip*) mapping to the *access-switch*. Also *host-switches* publish these mappings periodically to *access-switches* corresponding to the mappings. When a *host-device* sends an ARP request message to resolve an IP address, the *host-switch* extracts the IP address (*ip*) in the request, and if it is not one of its *host-devices*, it forwards the query to the *access-switch* corresponding to *ip* as an encapsulated ARP request message. Upon receiving the encapsulated ARP request *access-switch* looks up the requested IP address in the mappings stored by it and replies back with the *vid* if found, else it discards the request.

4. VEIL-CLICK: THE PROTOTYPE

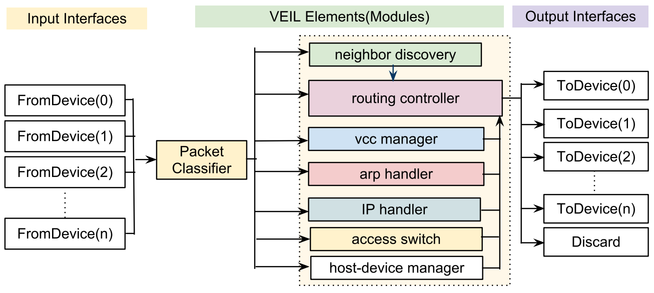


Figure 2: An overview of the design of Click based prototype.

veil-click is based on Click modular framework, where different functionalities are broken in to individual modules, which can be developed independently, and plugged-in together to compose a full fledged *veil-switch*. In case of *veil-click*, we build several elements (or modules) for various operations such as ARP handling, Routing Table management etc. as Click elements. These individual elements are connected together to form various components of *veil-click*, which are shown in Figure 2. In this figure rectangular boxes represent the Click elements (or a group of elements for simplified representation), while solid black lines represent the interconnections between them.

At the time of writing of this paper, we have finished the basic implementation of *veil-click*, which includes all the functionalities mentioned in the paper. We have tested the basic working of the current prototype using a small testbed in our lab. The initial testbed consists of 5 server machines with 5 Ethernet ports on each as *veil-switches*, 3 wireless access points which are directly connected to different *veil-switches* through wired Ethernet cables, and several laptop computers running Ubuntu/MAC OS X/Windows operating systems as test *host-devices*[4]. The complete test network is configured to use a single subnet prefix, and the *host-devices* are assigned static IP addresses.

Using this testbed we evaluated the support for host-mobility by connecting the *host-devices* to different wireless access points, while they are communicating with each other. For these experiments, we set up a TCP connection between two *host-devices*, where source *host-device* generate the traffic at constant bit rate (1600 kbps), and move the destination *host-device* between two different wireless access points. In Figure 3 we show how the mobility of the destination *host-device* affects its traffic receiving rate. In this figure, red dashed lines represent the transitions from one access point to another, and solid blue lines represent the average goodput every second at the destination *host-device*. In Figure 3(a) we show how rate changes when destination *host-device* for multiple such transitions. As seen in this figure, the mobility causes minimal disruption for the TCP connection. We zoomed in on these individual transitions, and show one such transition in Figure 3(b). Our results show that TCP connection stabilizes with in 2-5 seconds during all such network transitions.

In addition, we are currently in the process of rigorous evaluation of the prototype using larger network topologies. We have already performed the extensive evaluation of VIRO routing architecture using simulations on a variety of network topologies, which are presented in the VIRO technical report [10].

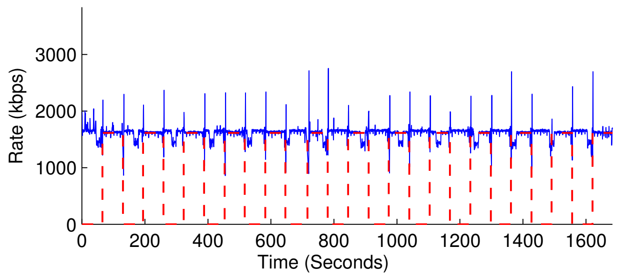
5. VEIL-CLICK: ADVANCED FEATURES

veil-click uses a unique modular design, which can be easily extended to incorporate several valuable features for large-scale layer-2 networks. These features not only simplify the design and management of large scale networks, but also provide many additional new features. In the following we describe some of these features.

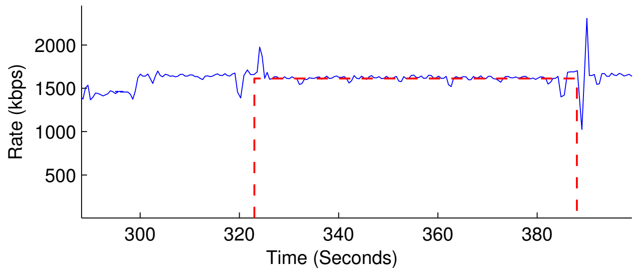
• **Policy Control using Custom Namespace Resolutions.** Unlike traditional Ethernet networks, where ARP request messages are broadcasted in the network, *veil-click* allows broadcast-free IP address resolutions using a DHT style look up and store service. It not only helps in significantly reducing the overhead of ARP, but also allows flexibility in restricting the "unwanted network" traffic for the hosts based on some pre-defined policies. This can be achieved by denying ARP resolutions based on the preset policies, which can take into account the identities of both source and destination hosts to make the resolution.

• **Robust Host Mobility Support.** *veil-click* allows hosts to keep the same IP address when they move from one *veil-switch* to another. In addition, it provides a smooth hand-off during the transition, such that it does not interrupt the network connection between the two

[4]We do not modify anything on the *host-devices* to connect them to the network, and no additional software is installed to allow them to communicate with the network



(a) Longer time window



(b) Shorter time window

Figure 3: Host-device mobility and the traffic bit rate.

hosts, if one of them changes their *host-switch*. It is achieved through "push" based notifications used by *veil-click*. When a *host-device* changes its *host-switch* by connecting to a different *veil-switch* it gets a new *vid* based upon the new switch's *vid*. This new *vid* is pushed using an unsolicited ARP reply packet to all the other hosts the host was talking to.

• **Fast Failure-Rerouting & Multi-path Routing.** Topologies for data-center networks and large-scale enterprise networks in general have rich path diversity. These topologies are designed to allow multiple paths to connect any pair of nodes for load-balancing or robustness. By utilizing multiple routing entries in the routing table at each bucket level, *veil-click* provides *built-in* support for multi-path routing, load-balancing and fast re-routing, with no additional complex mechanisms. This allows a node to choose different paths to reach a destination bucket, either for load-balancing or for fast-rerouting in case of link/node failures. Unlike ECMP, additional routing entries in case of *veil-click* are not limited to only shortest path routes, and therefore provide more flexibility in multi-path routing. (see [10] for details).

6. CONCLUSION

In this paper we presented a network architecture to create large-scale plug-&-play networks. It is designed with two broad sets of goals: i) to support – *with minimal manual configuration* – *large, dynamic* networks which connect tens or hundreds of thousands of diverse devices with *rich physical topologies* and reduce the management overhead; and ii) to meet the *high availability, robustness, mobility, manageability and security requirements* of these networks and the services running on top of them. These goals are motivated partly by the rise of huge data centers, emergence of cloud-computing and services, as well as the continued trends in large campus, enterprise and ISP (wired, wireless and cellular data) networks to use 1/10/100 Gigabit Ethernet as the core (layer-2) networking technology.

Toward these goals, we demonstrated our initial prototype *veil-click*, which is built using Click Modular Router framework. *veil-click* aims to significantly simplify the current management overhead for large-scale enterprise networks by automating most of the network configurations for both *host-devices* and as well as the routing nodes in the network. In addition, it provides built-in mechanisms for multi-path routing, fast failure re-routing, and seamless mobility support. Finally, the source-code for *veil-click* is publicly available, and can be downloaded from our Google Code project repository [3]. In addition, the development using OpenFlow [13] based design is currently underway.

Acknowledgement

The work is supported in part by the NSF grants CNS-0905037, CNS-1017647 and CNS-1017092 and the DTRA Grant HDTRA1-09-1-0050.

7. REFERENCES

[1] Cisco fabricpath. `http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9402/white_paper_c11-605488.pdf`.

[2] Ietf trill working group. `http://www.ietf.org/html.charters/trill-charter.html`.

[3] The source code repository for veil-click. `http://code.google.com/p/veil-viro-umn/source/browse/#svn/veil`.

[4] M. Al-Fares et al. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008.

[5] M. Caesar et al. Rofl: routing on flat labels. In *SIGCOMM*. ACM, 2006.

[6] B. Ford. Unmanaged internet protocol: taming the edge network management crisis. *SIGCOMM*, 2004.

[7] A. Greenberg et al. VL2: A scalable and flexible data center network. *SIGCOMM*, 2009.

[8] S. Jain et al. VEIL: A Plug-&-Play Virtual (Ethernet) Id Layer for Below IP Networking. In *BIPN workshop in conjunction with GLOBECOM'09*.

[9] S. Jain et al. Viro: A plug & play, scalable, robust and namespace independent virtual id routing for future networks. In *INFOCOM'11*.

[10] S. Jain et al. Viro: A plug & play, scalable, robust and namespace independent virtual id routing for future networks. In *Tech report*. http://networking.cs.umn.edu/veil/viro.pdf.

[11] C. Kim et al. Floodless in seattle: a scalable ethernet architecture for large enterprises. *SIGCOMM*, 2008.

[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 2000.

[13] N. McKeown et al. Openflow: enabling innovation in campus networks. *SIGCOMM '08*.

[14] D. Meyer. LISP, The Internet Protocol Journal, Volume 11.

[15] T. L. Rodeheffer et al. Smartbridge: a scalable bridge architecture. In *SIGCOMM*, 2000.