VEIL: A "Plug-&-Play" Virtual (Ethernet) Id Layer for Below IP Networking

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

***Abstract*—This paper proposes *VEIL*—a novel, "plug-&-play" *Virtual (Ethernet) Identifier (Id) Layer* for below IP networking. The objective is two-fold: i) VEIL directly addresses the scalability, efficiency and reliability challenges facing the traditional Ethernet, while retaining its "plug-&-play" feature; ii) but perhaps more importantly, VEIL provides a uniform (below IP) *convergence* layer to support a large, dynamic *and heterogeneous* (layer-2) network that is capable of connecting hundreds of thousands or more *diverse* physical devices. The key idea in our design is to introduce a *topology-aware, structured* virtual id (vid) space onto which both physical identifiers as well as higher layer addresses/names are mapped. VEIL completely eliminates *network-wide flooding* in both the *data* and *control* planes, and thus is highly scalable and robust.**

I. Introduction

The explosive growth of the Internet has enabled a wide range of diverse devices to be interconnected and communicate with each other through a variety of disparate technologies. While serving as the universal "glue" that pieces together various heterogeneous physical networks, the Internet Protocol (IP) suffers certain well-known shortcomings, e.g., in terms of need for careful and extensive network configurations, relatively poor support for mobility, and so forth. In contrast, layer-2 technologies such as Ethernet are largely "plug-&-play" in that hosts are equipped with persistent MAC addresses, and Ethernet switches automatically learn about host addresses and location, adapt to changes in network topology as well as host mobility, perform packet forwarding seamlessly with minimal operator configuration and intervention. Because of this simple "plug-&-play" semantics, today's *switched* Ethernet technology (where the collision domain is no longer a size-limiting factor) has been rapidly expanded to *large, dynamic* networks, such as large data centers and Metro Ethernet, with up to tens of thousands switches and millions of hosts.

On the other hand, the unprecedented scale as well as the demanding efficiency and robustness requirements of these new large, dynamic (layer-2) networks also pose revolutionary challenges on the Ethernet technology that was originally developed for small, local area networks. For instance, the *network-wide flooding*–often resorted by Ethernet switches to locate end hosts and forward packets whose locations are yet to be learned–not only significantly reduce the network capacity. The spanning tree algorithm used to avoid forwarding loops not only results in sub-optimal forwarding paths, but also is slow to adapt to changes in the network topology. To address these challenges, several solutions [7], [11]–[14] have been proposed, of which SEATTLE [7] is closely 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 routing in layer 2. It therefore not only requires network-wide flooding in the control plane for building routing tables, but also suffers the same scalability and robustness limitations plaguing shortest-path routing. (We refer the reader to Sec.II for further discussion of these and other related works.)

In this paper we propose *VEIL*—a novel, "plug-&-play" *Virtual (Ethernet) Identifier (Id) Layer* for below IP networking. The objective is two-fold: i) like SEATTLE, VEIL directly addresses the challenges facing the traditional Ethernet, while retaining its "plug-&-play" feature; ii) but perhaps more importantly, VEIL provides a uniform *convergence* layer (or "logical link layer" using the ISO OSI parlance) to support a large, dynamic *and heterogeneous* (layer-2) network that is capable of connecting hundreds of thousands or more *diverse* physical devices–not only Ethernet-equipped devices, but also non-Ethernet devices such as 802.16-based sensors, blue-tooth devices–in a *scalable* and *robust* fashion. The proposed VEIL architecture is a shim layer that operates under the (traditional) network layer (e.g., IPv4/IPv6) and above the ("native") link layer/physical layer such as Ethernet, 802.11 Wireless LANs, etc. The key idea in our design is to introduce a *topology-aware, structured* virtual id (vid) space onto which both physical identifiers (e.g., Ethernet MAC addresses), pid's in short, as well as higher layer addresses/names (e.g., IPv4/IPv6 addresses) are mapped. Using such a topology-aware, structured vid space as the basis for efficient and scalable (DHT-style) object look-up/address resolution, routing and forwarding, VEIL completely eliminates *network-wide flooding* in both the *data* and *control* planes. As a result, VEIL is highly scalable and robust while at the same time offers better support for multi-homing and mobility. In Sec. II we provide an overview of the proposed VEIL architecture, and in Sec. III we describe the key mechanisms of VEIL. Initial evaluation results are presented in Sec. IV, and the paper is concluded in Sec. V.

II. Overview and Related Work

A. Overview of VEIL

The proposed *VEIL* architecture is a shim layer that operates under the (traditional) network layer (e.g., IPv4/IPv6) and above the ("native") link layer/physical layer such as Ethernet, 802.11 Wireless LANs, etc. For simplicity of exposition, here we assume that each physical device has a 48-bit Ethernet MAC address, although this is not necessary. In fact, as will be clear later, the proposed VEIL architecture allows *heterogeneous* physical end devices with other types of MAC addresses

(e.g., smart phones with "hard-coded" phone numbers, bluetooth devices, 802.16-based sensors, etc.) to be plugged into the same layer-2 network. In lieu of Ethernet switches or wireless access points (APs), we have *VEIL switches* to which end devices (either through wired or wireless channels) are connected: they "speak" the *native* MAC/physical protocols to deliver data to/from these connected physical end devices; among themselves, they communicate using the VEIL protocol and perform VEIL operations such as vid assignment, mapping, routing (see below) to provide scalable and robust end-to-end connectivity and data delivery within a VEIL network.

The key idea in our design is to introduce a *topology-aware, structured* virtual id (vid) space onto which both physical identifiers (e.g., Ethernet MAC addresses), pid's in short, as well as higher layer addresses/names (e.g., IPv4/IPv6 addresses) are mapped (see Fig. 1a). By *topology-aware*, we mean that the physical network topology (formed by the connections among VEIL switches) is *embedded* into a *structured* space (e.g., a Kademlia-like virtual tree [9], a hypercube, a d-dimensional Euclidean space) in such a manner that *physical proximity among VEIL switches are approximately preserved.* More precisely, VEIL switches are assigned vid's in a manner such that if they are *logically* close in the vid space (based on a logical distance function, e.g., longest prefix, Hamming or Euclidean distance) are also *physically* close (e.g., in terms of hop counts) to each other. Using a (binary) Kademlia virtual tree as an example, Fig. 1b shows such an embedding: the leaf nodes correspond to VEIL switches (*not* physical devices connecting to them!), the vid of a VEIL switch is the binary strings along the path from the root to the corresponding leaf node. (We note here that the tree is *virtual* in the sense that any intermediate node (or the root) in the tree does not correspond to any physical switch/device, but the collection of switches/devices share the same prefix[1].) The logical distance between a pair of vids in this vid space is defined as the difference of number of bits used to represent a vid and the length of the longest common prefix for the pair. Hence it is clear that all VEIL switches within the same sub-tree (thus with logically closer vid's) are also physically closer. In Sec. III we will briefly discuss how the vid assignment can be performed in either a centralized or distributed fashion. End devices connecting to a VEIL switch inherit an *extended* vid consisting of the (32-bit) vid of the switch plus a (randomly assigned) 16-bit local vid (see Sec. III for detail).

Throughout the paper, we will use a binary Kademlia (virtual) tree to illustrate how the proposed VEIL works. (More generally, a K-ary tree may be used.) Taking advantage of this topology-aware, structured vid space, VEIL switches run a vid routing protocol (referred to as *VIRO*) to collaboratively build routing tables, maintain network-wide connectivity[2], and perform end-to-end data delivery across a VEIL network. In VIRO, routing tables are constructed piece-meal based on the vid logical distance instead of physical distance (e.g., hop counts), and packets from a source VEIL switch are forwarded towards their destination along a logical path with decreasing logical distance (to the vid of the destination VEIL switch). In Sec. III we briefly outline the basic operations of VIRO, and describe how packets are forwarded by VEIL switches using VIRO. The detailed description of VIRO and proof of its correctness can be found in [6] (see also [8] for an outline of VIRO in the context of a mobility architecture).

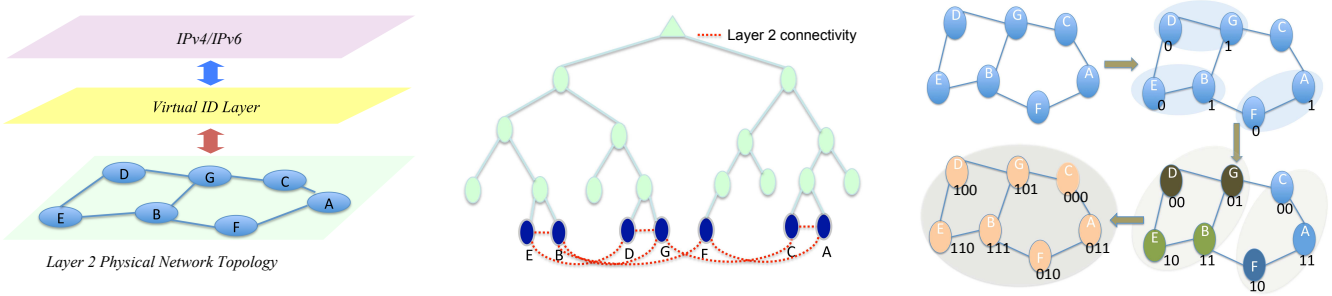
While at the expense of incurring additional routing stretches–albeit fairly small in general thanks to the topology-aware construction of the vid space (see Sec. IV)–when compared to the shortest path (using the physical distance) routing, the logical distance-based VIRO routing affords several important advantages. i) *Scalability.* By taking the *structured* vid space and constructing routing tables in a piecemeal, bottom-up fashion, VIRO completely eliminates *network-wide flooding* in both the data plane (unlike Ethernet switching algorithm) and *control plane* (unlike OSPF and other shortest path routing algorithms). Furthermore, because of the natural *hierarchical* structure of the vid space, routing information regarding far-away part of the network is automatically aggregated using the vid prefixes. As a result, the routing table size is in the order of $O(\log N)$, where N is the number of VEIL switches in the network, as opposed to $O(N)$ (as in the case of OSPF). ii) *Robustness.* Unlike OSPF, no *network-wide* full topology needs to be maintained by any switch, thanks to the structured vid space, and hence changes in network topology do not need to be flooded globally. Due to the aggregate routing information maintained by switches, failure of a link or switch node can be *localized*, without affecting nodes in far-away parts of the network. Furthermore, path and topology diversity can be easily exploited in VIRO by using multiple forwarders; hence failure of one forwarder does not affect network-wide reachability. iii) *Multi-Homing and Mobility.* VIRO also provides seamless and better support for multi-homing and mobility. It allows an end device to be connected with multiple VEIL switches without causing loops and other complexities; mobility of an end device can be easily supported through scalable and efficient pid (or higher layer names/addresses) to vid mapping and lookup.

B. Related Work

There are several proposals, e.g., RBriges [12], CMU-Ethernet [11], Viking [14], SmartBrigdes [13], that attempt to address the scalability limitations in scaling Ethernet to *large, dynamic* networks (see Sec. 1.1 of [7] for more detailed discussion on these proposals). Our work is closely related

[1]As such, it does not make sense to talk about the failure of an intermediate node! Hence unlike a physical tree (e.g., as used in Ethernet spanning tree), failure of any VEIL switch only affects a leaf node in the (virtual) tree. Namely, there is no single (*intermediate*) point of failure.

[2]Unlike the peer-to-peer (P2P) Kademlia routing protocol which operates on top of the IP network layer and thus the end-to-end connectivity is assumed (and maintained by the underlying network layer), VIRO has to build *network-wide connectivity* in a "bottom-up" manner based on the (native) link layer connections.



(a) Overview of VEIL (b) VEIL based topology (c) Construction of VEIL

Fig. 1. Design of Virtual (Ethernet) ID Layer

to SEATTLE [7], with similar goals but two *key differences*. As pointed out earlier, while SEATTLE eliminates data plane flooding, it employs OSPF-like shortest path routing, which requires *network-wide flooding* of link state advertisements (LSAs) in maintaining network topology and tracking its changes. SEATTLE thus suffers the same limitations plaguing OSPF-based IP routing. For example, Node and link failures therefore require network-wide flooding of LSAs and re-computation of routing tables at all nodes. For scalability, hierarchical, area-based routing must be introduced, which in turn introduces additional management complexity as well as routing stretch penalty. In contrast, with the introduction of vid's and a structured vid space, VEIL can support a large, dynamic (below IP) network with heterogeneous devices, not simply Ethernet-enabled devices, in a more self-organizing and scalable fashion (e.g., with $O(\log N)$ routing table sizes instead of $O(N)$).

Our work is also substantially different from the identifier-based routing schemes such as VRR [3], UIP [5] and ROFL [4], which advocates 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 a 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). In addition, link and node failures and dynamics (joining, leaving or moving around in the network) often induce a network-wide effect, as two logically close nodes may be far away in the underlying physical network In contrast, VEIL introduces a *topology-aware*, structured vid space, and as a result, the VIRO routing protocol incurs fairly small and *provably bounded* routing stretches, and effectively localizes the effect of failures. Lastly, in an earlier work of ours [8], we proposed the use of the vid space as the basis for a new mobility architecture–where each end device is assigned a fixed flat universal id (uid) and a dynamic vid that depends on its current location in the network for routing purpose–to provide efficient, scalable and flexible support of end-host mobility.

III. The VEIL Layer

In this section we outline the key mechanisms of VEIL: vid assignment, the VIRO routing protocol, vid lookup/address resolution, and end-to-end packet forwarding. We will use the notation $v(x)$ or v_x to denote the virtual id of a host x (or VEIL switch), and $hash_k(val)$ represents a k-bit hash value for val using a consistent hash function. We will refer to *end-host devices* attached to switches as *hosts*, and the terms *node* and *switch* are used interchangeably to denote a VEIL switch.

A. VEIL Id Assignment

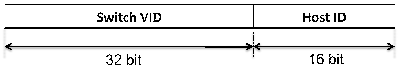


Fig. 2. vid field structure.

To be backward compatible with Ethernet MAC addresses, VEIL assigns a 48-bit long vids (VEIL Id) to each switch/host in a VEIL network. As shown in Fig. 2, a vid comprises of two parts: The first part, the $switch\text{-}vid$ field, is a 32-bit identifier that uniquely identifies a VEIL switch in the network. The second part, the $host\text{-}id$ field, is a 16-bit identifier that uniquely identifies a host attached (through either a wired or wireless link) to a VEIL switch. We refer to the switch that a host is currently attached to as the *host-switch* of the host. The vid of each host is assigned as follows: the first field is set to be the same $switch\text{-}vid$ as its host-switch, and the second field is randomly assigned, e.g., by using a 16-bit hash of its MAC address (pid), $host\text{-}vid := hash_{16}(pid)$, and is locally unique with respect to the host switch. For a switch, its $host\text{-}vid$ part of its vid is assigned in a similar fashion. In the following we will discuss how the $switch\text{-}vid$ part of a switch vid is assigned. For conciseness, unless otherwise stated, the term vid simply refers to the $switch\text{-}vid$ part of switch vid's, as they are the ones that form a *topology-aware*, *structured* vid space and are used for routing and forwarding.

In setting up a VEIL network (and before running the VEIL vid assignment algorithm), a global parameter L ($1 \leq L \leq 32$) is selected, which is the (maximum) height of a (binary) Kademlia virtual tree of the vid space. *By default, L is set to 24.* The logical distance between two vids v_x and v_y is then given by

$$\delta(v_x, v_y) = L - lcp(v_x, v_y)$$

where $lcp(v_x, v_y)$ is the length of the longest common prefix between (the first L-bits of) v_x and v_y. Thus, the largest logical distance between any two switches is 24. In addition, each leaf node (i.e., a *logical* level-0 node) in the Kademlia virtual tree

corresponds to a collection of switch nodes with the same L-bit prefix (i.e., of logical distance 0). Given this logical distance function, the vid assignment must ensure the following two properties: i) The *closeness* property: if two nodes are close in the vid space, then they are also close in the physical topology; in particular, if two nodes are *logical* neighbors (i.e., sharing the same L-bit prefix), they are directly connected. ii) The *connectivity* property: any two *adjacent* logical sub-trees (i.e., sharing a common prefix) must be physically connected, i.e., there must be at least one physical (wired/wireless) link connecting one node of each subtree.

We have designed two modes of vid assignment for bootstrapping a VEIL network using either a *distributed* or *centralized* algorithm which guarantees these two properties. For example, in the distributed mode, nodes run a bottom-up, recursive clustering algorithm based on the proximity: they first randomly generate $(32 - L)$ bits as the lowest $32 - L$ bits of their vid's; they then exchange information with their neighbors, and iteratively merge with their neighboring nodes to form a larger super-node, and assign 0 or 1 to the next bit in their vid's (see [6] for details). An example of the distributed vid assignment is schematically shown in Fig. 1(c). Initially, all nodes are level-0 nodes. In the first round of the vid assignment, nodes merge with their neighboring nodes to form (level-1) super-nodes (DG, BE, AF, C) by assigning a 1-bit (0 or 1) vid to distinguish themselves in the super-node. In the second round, level-1 nodes are clustered, as (DGBE, AFC), and additional 1-bit is prepended to their vid's. This process continues until all nodes are assigned a unique L-bit vid's. In the centralized mode, one node is picked as the master node and collects the network topology. It then runs a top-down graph partition algorithm and assigns vid's to nodes accordingly. Alternatively, a few (central) nodes may be first selected and manually assigned a unique prefix, and afterward a top-down clustering algorithm is used to expand the prefixes and assign vid's to all nodes. This method, for example, is especially useful and efficient in data center or other networks where the network topology has a natural hierarchical structure.

After the network bootstrap process, when a new node is added to the network, it simply talks to its direct neighbors to learn their vid's. It auto-assigns its own vid by setting the first L bits to be the same as one of its neighbors, and generating a unique bit-string for its remaining $32 - L$ bits[3]. Lastly, we remark that the end host in fact does not need to know its vid at all; as will be clear in Sec. III-C, the vid of an end host is used only by its host switch to look up and translate between its vid and pid (MAC address), thus not used by the end host itself at all.

[3]Depending on the type of networks to be constructed, L can be judiciously configured initially to account for future network expansions, and thus it makes additional vid assignment possible. For instance, using the default value $L = 24$, we allow 256 nodes residing with the same leaf node.

TABLE I
ROUTING TABLE OF NODE C

Bucket	Prefix	Nexthop	Gateway
1	001	Nil	Nil
2	$01*$	$A(011)$	$C(000)$
3	$1**$	$G(101)$	$C(000)$

B. Virtual Id Routing (VIRO) Protocol

We briefly outline the key ideas behind the VIRO routing protocol (see [6] for detail). For each node x, we define its level-k *bucket*, denoted by B_k^x, as the group of nodes which are k logical distance away from x, and a level-k *sub-tree*, denoted by $S_k(x)$, as the set of all the nodes at logical distance no more than k from x. Formally, we have

$$B_k^x = \{y : \delta(x, y) = k\} \text{ and } S_k^x = \{y : \delta(x, y) \leq k\}.$$

For $k = 0, \ldots, L-1$, we say a node $p_k \in B_k^x$ is a *gateway* for a level-$(k+1)$, B_{k+1}^x, if $\exists p_{k+1} \in B_{k+1}^x \wedge \exists p_k \in S_k^x \wedge (p_k, p_{k+1}) \in E$, where E is the edge set for direct (physical) connections among nodes. In VIRO, a node only needs to maintain one routing entry to reach any node in bucket B_k^x, $1 \leq k \leq L$. For (directly connected) nodes in the level-0 bucket, B_0^x, one entry per node is maintained. Each routing entry contains the following fields: the bucket-level and its corresponding prefix, a list of gateways, and a list of next-hops for reaching the gateways. (A next-hop is a node directly connected to node x.) Table I shows an example of the routing table for node C in the network topology shown in Fig. 1.

The construction of the routing table uses a bottom up round-by-round protocol, where after the kth round, each node x has built a routing entry to reach nodes in its bucket B_k^x. During round 0, each node x discovers its directly connected neighbors, and all nodes in B_0^x. During round k, $1 \leq k < L$, x uses a publish/query based mechanism to i) either publish itself as a gateway to B_{k+1}^x if it has a direct connection to a node in B_{k+1}^x, or ii) query to discover such a gateway so as to install a routing entry for bucket B_{k+1}^x. Because of the closeness and connectivity properties of the vid assignment, this process is guaranteed to converge and generate the correct routing table. The details of the routing algorithm and its correct proof can be found in [6].

The forwarding process using VIRO is simple. In order to forward a packet from switch s with vid v_s to a destination d with vid v_d, s first computes the logical distance $k = \delta(v_s, v_d)$. If $k > 0$, s forwards the packet to a nexthop corresponding to Bucket B_k^s in the routing table. If $k = 0$ and v_d is not a host directly connected to s, s forwards it to d's host switch, which then delivers to d. If v_d is a host directly connected to s, then s directly delivers it to d (see the next subsection for how this is done).

C. Id Lookup/Address Resolution, and End-to-End Packet Delivery

Like SEATTLE [7], we employ a one-hop DHT for id lookup and address resolution. Recall that each host-switch assigns vids to all the hosts connected to it. A host switch discovers the MAC/IP addresses (denoted by pid and IP respectively) of an end host that is directly connected to it by

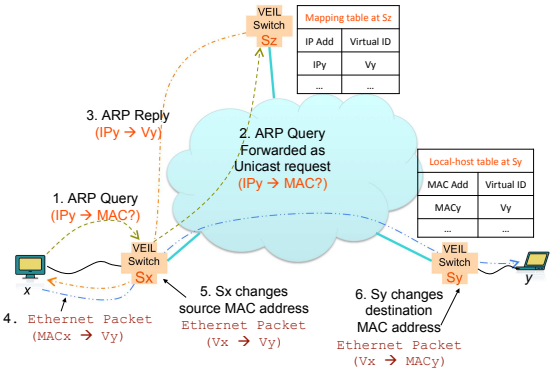


Fig. 3. *vid* lookup and address resolution process.

listening to (or "snooping") all traffic on the link[4]. It assigns a vid to each host, and store the $\langle pid, vid\rangle$ mapping as well as the $\langle IP, vid\rangle$ into a local cache. Furthermore, the host switch periodically publishes the mapping $\langle IP, vid\rangle$ (as long as the end host is still connected to it) to a specific switch, called the *access switch* (of the said IP address), the $switch\text{-}vid$ of which is closest to the 32-bit hash, $hash_{32}(IP)$. The access switch stores this $\langle IP, vid\rangle$ mapping into its cache, and is responsible to answering queries for the said IP address from other nodes.

Fig. 3 depicts the address resolution, id lookup and packet delivery processes. When host x wants to send a packet to host y with IP address IP_y, it first sends an ARP query to resolve IP_y to MAC address of y (as in the standard IP operation). This ARP packet is intercepted by the host-switch S_x. If host y is not directly connected to S_x (otherwise, everything operates as the usual Ethernet), the host switch S_x sends a unicast request to the access switch S_z whose $switch\text{-}vid$ is closest to $hash_{32}(IP_y)$. Upon receiving the request, S_z replies with the corresponding $\langle IP, v_y\rangle$ mapping. S_x caches this mapping in its local cache for future packet forwarding, and sends an ARP reply to x with v_y as the MAC address. Host x encapsulates the IP packet in an Ethernet packet with v_y as the destination MAC address. When switch S_x receives this packet, it replaces the source MAC address (host x) by its vid, and forwards the packet toward the host switch, S_y, of the destination host y, using the VIRO packet forwarding described earlier. Upon receiving the packet, the host switch S_y uses v_y to look up the actual MAC address of host y in its local cache, rewrite the destination MAC address field, and forward the packet to host y.

D. Support for Mobility and Legacy Protocols

Host mobility with a VEIL network can be easily supported by updating the IP to vid (or MAC/ pid to vid) mappings. The host-switches are responsible for updating these mappings and publishing them to the corresponding access-switches, whenever changes take place. VEIL is also completely compatible with the current Ethernet semantics such as ARP/RARP, DHCP and VLAN. As explained in Sec. III-C, the ARP protocol is supported with a flooding-free lookup/query mechanism. Similarly, RARP and DHCP can also be supported: a host-switch simply intercepts the appropriate packets, and then forwards them via unicast to appropriate destinations. Moreover, VLAN can also be accommodated by, say, setting aside the first 8-bit of the $switch\text{-}vid$ as the VLAN number. For each VLAN, there will be one corresponding (virtual) VEIL network, each with its own logical topology and vid sub-space. In this way, VEIL can inherit all the advantages of VLANs, while at the same time improving scalability, security, and network management.

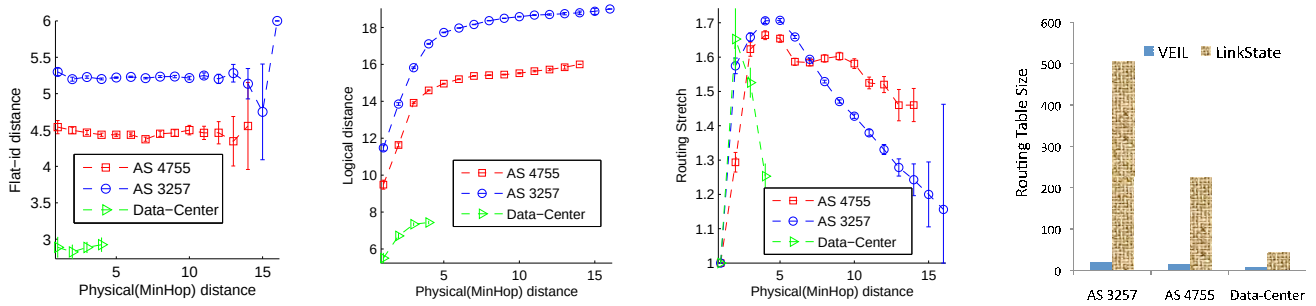
[4]For an end host that is *VEIL-enabled*, it may run an explicit *association* protocol to associate with its host switch (similar to the host/AP association in 802.11 wireless LAN), reports its MAC/IP address, learns its vid assignment, and set its interface to receive packets with the destination MAC address equal to its vid. For such hosts, no pid/vid translation is needed at the host switch.

IV. Initial Evaluation

To evaluate the performance of the proposed VEIL architecture, we have implemented it in a simulation framework using JAVA and Omnet++ [1]. Since there is no publicly available large campus network topologies, we tested VEIL using some of the AS topologies available from RocketFuel [15] , and several data-center topologies. We present our results for the following three topologies: *i) AS 3257:* This topology consists of 506 nodes and 750 edges. *ii) AS 4755:* This topology consists of 226 nodes and 285 edges. *iii) Data center topology:* We use a fat-tree like topology to represent a typical Data-Center network [2]. It consists of 9 core switches, 18 aggregation switches and 18 intermediate switches in the topology.

Evaluating the vid Assignments. Using the three topologies, we evaluate the efficacy of the vid assignment in generating a topology-aware vid space, using the distributed bottom-up clustering algorithm mentioned in the paper. We first assign each node in the network with a randomly assigned *flat-id* (as a proxy for the MAC address (pid) of the node). The *flat-id distance* between a pair of nodes is defined as the difference of number of bits used to represent a vid and the length of the longest common prefix of their *flat-id*s. (This would be the logical distance between two nodes if one were to directly use these flat-id pid's for routing, as in [3], [4]). Using the random *flat-id* assignment, Fig. 4(a) shows the distribution of flat-id distances of all node-pairs, as a function of their physical distances (measured in terms of hop counts). In contrast, Fig. 4(b) shows the same distribution of logical distances using our distributed VEIL id assignment algorithm. We see that using the VEIL id assignment, the logical distance increases linearly with increasing physical distance. This shows that the VEIL id assignment indeed embeds the physical topology into its structured vid space.

Routing Stretch. The routing stretch is defined as the ratio of the path length taken by a packet from a source to a destination using the VIRO routing protocol to that of the shortest (min-hop) path between the two nodes. As shown in the Fig. 4(c), the routing sketch stays between 1-1.5 for most of the pairs. Hence the VEIL VIRO routing protocol incurs a fairly small routing sketch, while affording more scalability and resiliency.



(a) Flat-id Distance (b) Logical Distance (c) Routing Stretch (d) Routing Table Size

Fig. 4. Evaluation results for VEIL

Routing Table Size. In Fig. 4(d), we compare the size of per node routing tables for VEIL (using VIRO routing protocol) and link-state-based shortest-path (as used in SEATTLE) routing protocols. The huge difference in the routing table sizes comes from the fact that the number of routing table entries in the shortest path routing algorithm scales with the number of nodes in the network; in VEIL, it scales logarithmically.

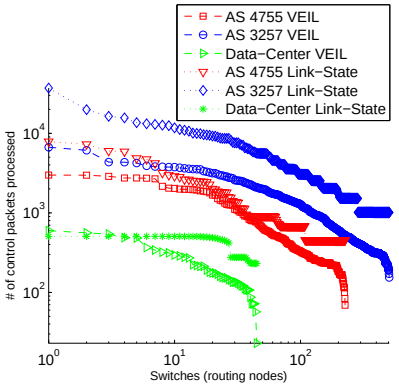


Fig. 5. Control overhead comparison for VEIL and Link State routing protocol.

Control-Overhead. In Fig. 5, we compare the control overhead for both VEIL and link-state-based routing protocols, in terms of the number of control packets processed by each node for a single round of routing table construction. We see that compared to the link-state routing protocol, VEIL significantly reduces the control overhead. This reduction in control overhead is achieved by VEIL, as it eliminates network-wide flooding, while the link state routing protocol requires network-wide flooding of link state packets.

Prototyping VEIL using OpenFlow switches. OpenFlow [10] is a recently developed platform to allow the programmability of switches in the enterprise/campus networks. We are currently working on the prototype of VEIL based forwarding using OpenFlow switches. We also plan to implement VEIL using the Click software router platform.

V. Conclusion & On-going work

In this paper we have presented *VEIL*—a novel, "plug-&-play" *Virtual (Ethernet) Identifier (Id) Layer* for below IP networking. The key idea in our design is to introduce a *topology-aware, structured* virtual id (vid) space onto which both physical identifiers as well as higher layer addresses/names are mapped. VEIL completely eliminates *network-wide flooding* in both the *data* and *control* planes, and thus is highly scalable and robust. Our initial simulation-based evaluation has demonstrated the immense scalability and robustness of VEIL. We are currently conducting extensive simulations to further evaluate the performance of VEIL and finalize the protocol and other mechanism development. As part of the on-going work, we will plan to implement VEIL using the OpenFlow switches and the Click software platform.

Acknowledgement. This work is supported in part by the National Science Foundation grants CNS-0626808 and CRI 0709048.

References

[1] Omnet++: A discrete event simulation environment. http://www.omnetpp.org/.
[2] M. Arregoces and M. Portolani. *Data center fundamentals*. Cisco Press, 2004.
[3] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by dhts. *SIGCOMM Comput. Commun. Rev.*, 2006.
[4] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. Rofl: routing on flat labels. In *SIGCOMM*, 2006.
[5] B. Ford. Unmanaged internet protocol: taming the edge network management crisis. *SIGCOMM Comput. Commun. Rev.*, 2004.
[6] S. Jain, Y. Chen, Z.-L. Zhang, and S. Jain. Viro: Scalable, robust and naming-proof plug & play routing. In *Tech report*. http://www.cs.umn.edu/~sourj/techreports/viro.pdf.
[7] C. Kim, M. Caesar, and J. Rexford. Floodless in seattle: a scalable ethernet architecture for large enterprises. *SIGCOMM*, 2008.
[8] G.-H. Lu, S. Jain, S. Chen, and Z.-L. Zhang. Virtual id routing: a scalable routing framework with support for mobility and routing efficiency. In *MobiArch Workshop*, 2008.
[9] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS02*, 2002.
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM*, 2008.
[11] A. Myers, E. Ng, and H. Zhang. Rethinking the service model: Scaling Ethernet to a million nodes. In *HotNets*, 2004.
[12] R. Perlman, J. Touch, and A. Yegin. RBridges: transparent routing. In *IEEE INFOCOM*, 2004.
[13] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. Smartbridge: a scalable bridge architecture. In *SIGCOMM*, 2000.
[14] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: A multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks. In *IEEE INFOCOM*, volume 4, 2004.
[15] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 2004.