A Virtual Internet Architecture[1]

Joseph D. Touch, Yu-Shun Wang, Lars Eggert, Gregory G. Finn
{touch,yushunwa,larse,finn}@isi.edu
USC/Information Sciences Institute
March 24, 2003

Abstract – *A Virtual Internet (VI) is an IP network composed of tunneled links among a set of virtual routers and virtual hosts. The architecture, like its virtual memory counterpart, provides an abstraction that hides the complexity of the underlying network and provides isolation-based protection that encourages resource sharing. A VI completely decouples its component hosts and routers from the underlying network to support both sibling and recursive VIs and to allow a node to participate multiple times in a single overlay, known as revisitation. The VI architecture provides a consistent multihoming paradigm, including end-to-end overlays, naming and addressing, virtual host requirements, and virtual gateway requirements. Consequences of the architecture are presented, including basic implications on the underlying network and host operating system, as well as additional requirements and mechanisms needed to support recursion and revisitation. Several implementations based on this architecture are discussed that explore the capabilities of VIs, including automated deployment and management, recursion for fault tolerance, geographic delivery, and support for peer-to-peer systems.*

I. Introduction

A Virtual Internet (VI) is a virtual version of the Internet in which virtual hosts and routers are connected by IP-encapsulation tunneled links over the existing Internet (Figure 1). A VI is an overlay network that rides on top of an IP network, and all the capabilities of the Internet in the overlay. VIs can provide security and isolation like VPNs, for whole virtualized networks rather than just remote hosts or subnets, unlike VPNs.

VIs generalize the tunnel backbones that helped deploy multicast (M-Bone), IPv6 (6-Bone), and Active Networks (A-Bone) [1][4][8]. Those backbones enabled new protocols to be tested and incrementally deployed on existing infrastructure. Unlike those interim solutions, VIs are intended as a more permanent capability, further enabling incremental tests, as well as to support persistent partial deployments of new capabilities where desired.

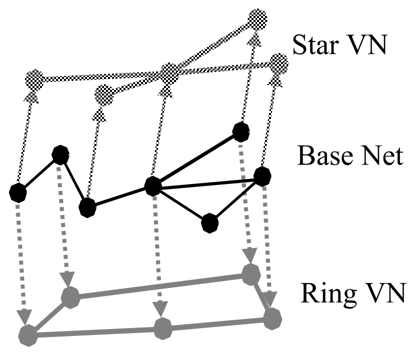

Figure 1 Multiple virtual Internets

Like their virtual memory (VM) counterpart, VIs manage concurrent sharing of resources, provide protection, and present an abstraction of the underlying service. Each process on a host may participate in a different VI (Figure 2). Address spaces of VIs are managed to avoid overlap except where they explicitly share components (gateways). Applications can interact with simplified topologies, e.g., trees for DNS or rings for web caches, without requiring application participation in creating or managing those structures – identical to that provided by the Internet, in which the name resolution and forwarding are provided services.

[1] Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreements number F30602-98-1-0200 entitled "X-Bone" and number F30602-01-2-0529 entitled "DynaBone". The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

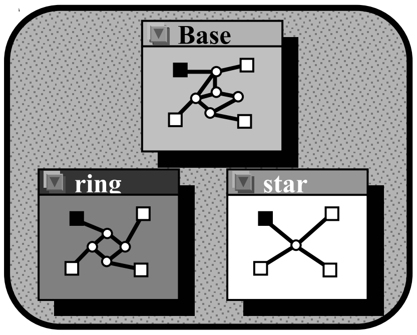

Figure 2 User's view of VIs

The VI architecture provides new capabilities for virtual networks, including recursion and revistation. Recursion allows VIs to be stacked, providing new opportunities for fault tolerance and path diversity. Revisitation allows a single base network node to emulate multiple VI nodes, allowing a VI to emulate larger networks than the base on which they are deployed.

The need to be virtual…

VM provides a paradigm that decouples the process memory from physical memory. Processes view memory as a single linear address space, starting at zero. Programmers need not focus on page layout in memory frames, address translation, paging, or even swapping to disk to emulate larger memory spaces. Processes are protected from each other's space, and space is allocated and managed by the operating system.

Current networking, by contrast, is closer to pre-VM programming. Distributed systems are exposed to the underlying network topology, and programmers or applications manage tunnels directly. Address spaces can collide, and there is no effective way to emulate a larger topology. Overlays are manually configured, with no automation to clean up abandoned overlays.

The VI architecture provides VM-style abstraction to networking, decoupling the virtual topology from the physical, and enabling automated coordination and management that avoids these pitfalls. Recall the star and ring overlays shown in Figure 1, and how they map onto the underlying network (shown in Figure 3 and Figure 4).

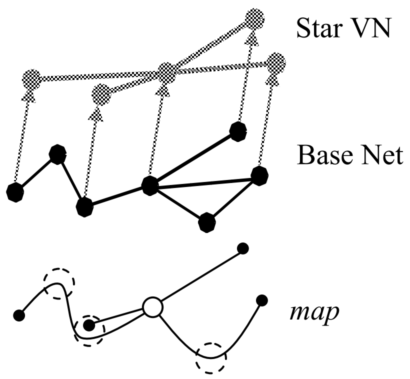

Figure 3 Map of star onto base

Underlying nodes in the base network may act as virtual hosts (solid dots), virtual routers (solid rings), tunnel transits (dashed rings), and some do not participate at all (dotted rings). Note that some nodes in the base network participate in multiple ways, e.g., all nodes in the ring are both virtual hosts and virtual routers (circle around dot), and some are also tunnel transits.

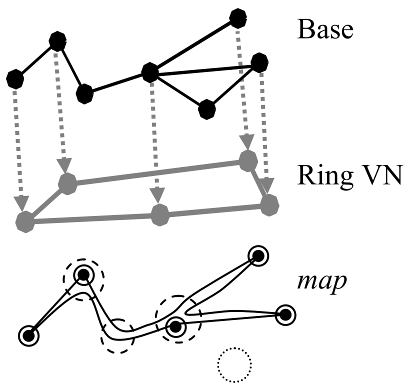

Figure 4 Map of ring onto base

There are a number of overlay systems that address portions of the VI architecture, and others that justify the need for one. Some of the VI architecture is already represented by M-Bone-style overlays and VPNs [8][19]. The broader end-to-end networking of VIs complements the backbone focus of these systems, and builds on the work of the X-Bone, VNS, and RONs [2][12][20]. The abstract nature of the VI and the presence of a fall-back base network enable automated VI management.

The VI also supports recursion and revisitation and resolves the impact of overlays and these extensions on the host and router requirements of the current Internet. Some systems recapitulate these requirements at the application layer (e.g., peer-to-peer networks), while others are limited to simplified deployments to avoid those issues (typical VPNs) [14]. Finally, few of these systems are interoperable or are advocated as permanent extensions to the Internet architecture.

This paper presents the VI architecture and discusses its virtual host, virtual router, and virtual link components, and the implications of virtualization on host and router requirements. New mechanisms that support new capabilities are presented, including a system that enables recursion. Finally, the architecture is applied to develop new services such as geographic delivery and recursion for fault tolerance, as well as to enable automated network management.

II. VI ARCHITECTURE

The VI extends the Internet model of networking, composed of data sources and sinks called hosts, data transits called routers (originally called gateways), and links between them [3][5]. In the Internet architecture, links are essentially static; link creation and tear-down occurs during network deployment, whereas links are configured up or down on shorter timescales. Forwarding is the process of deciding the next-hop destination of packets traversing a transit, and routing is the system of information exchange and computation that determines forwarding tables.

A VI virtualizes all these components. A single network node (host or router) may participate as virtual host, virtual router, or multiples of each simultaneously. Virtual links are part of VI deployment, and whether a link is available or not can affect VI routing. Note that VI links are not deployed as a result of routing computations; the act of provisioning a VI network and the routing within are decoupled.

The specifics of the VI model begin with its requirements and assumptions. The architecture is then presented, including details of its components and their characteristics. The implications of these decisions are addressed in Section III. The following describes the requirements and architecture of a system which has been implemented in FreeBSD and Linux. Details of the implementation are discussed throughout, and summarized in Section V.

II.1. Requirements

First and foremost, a VI is a network, a complete topology consisting of virtual hosts and virtual routers. This is distinct from typical VPNs, which tether individual remote hosts to an existing private network, or tie together two separate pre-existing networks. Other systems focusing on backbones include VNS, PPVPN, and many of the original overlay backbones (M-Bone, 6-Bone, A-Bone). It is critical to include hosts in the architecture; otherwise, key issues in addressing and source address selection are obviated.

VI further inherits all the conventional aspects of the current Internet, including the previous definitions of hosts, routers, and links, as well as the assumption of conventional multihop paths and unique (at least within an overlay) endpoint addresses. These assumptions are the result virtualizing the underlying network – the Internet. A VI runs over the Internet, and provides an IP overlay with all the capabilities of the existing Internet.

The overall goal of the VI is to provide an environment that applications and routing cannot distinguish from the existing underlying Internet. The resulting system thus avoids the need to reinvent network, transport, and application protocols, or to reimplement routing and forwarding algorithms, as with application-layer peer-to-peer systems [14]. It also enjoys the scalability of hop-by-hop forwarding using local decisions that link-layer emulation does not.

The VI includes additional assumptions which are not strictly required, but desired for a consistent and generic architecture. VI links use IP in IP encapsulation to avoid the need for new, non-ubiquitous protocol support [15]. Security – the P in VPN – is orthogonal, a property of links that can be enabled or disabled without effect on the rest of the network. The purpose of VI security is to preserve the contents of links, to protect the network layer, including routing exchanges. Additional end-to-end application security is provided by existing Internet transport or application layer protocols.

Finally, a VI assumes concurrence, recursion, and revisitation. Individual components can be members of multiple VIs concurrently. A VI can be deployed on top of an existing VI, just as on a base Internet, where the upper VI is a subset of the nodes of the lower network. Individual components can participate multiple times in a single VI overlay. All these properties ensure consistency and attempt to create the most flexible and capable generic VI architecture.

Summary of VI Architecture

The VI architecture can be summarized by a few basic principles:

1. VIs are composed of VRs and VHs connected by IP encapsulated tunnel links, emulating the Internet architecture
    a. Virtual hosts are data sources and sinks; only VHs increase or decrease the number of headers on a packet (i.e., encapsulate or decapsulate)
    b. Virtual routers are data transits; only VRs transit packets without changing the number of headers
2. VIs are completely virtual, decoupled from the base network on which they are deployed
    a. VIs support concurrence
    b. VIs support revisitation
3. VIs recurse by emulating a VI network as a VR in the base network
    a. In control recursion the inner network has unbound upper VR interfaces which *are* the interfaces of the emulated VR in the base network
    b. In network recursion the inner network has phantom VHs which *are* the interfaces of the emulated VR in the base network; these phantom VHs add and remove the headers of the recursive VI

The details of the architecture follow from these basic principles. Notably, even some of these principles (the a,b items) follow from the main axioms (1,2,3).

II.2. Components

The VI architecture is composed of virtual hosts, routers, and links, using a variant of unique endpoint addresses. VI hosts and routers develop and extend the Internet's currently limited multihoming capability to support concurrent VI participation. VI links incorporate two layers of tunneling to support revisitation, representing separate virtual link and network layers. Recursion is supported, both of configuration and deployment, as well as true network-on-network recursion.

Nodes of a VI overlay use unique endpoint addresses for interfaces on both hosts and routers, as with the conventional Internet. Addresses within each overlay are unique, but addresses may be reused in another overlay provided the two overlays share no nodes in common in the underlying network. This is known as "ships in the night" (SITN) addressing, after the term used to describe protocols that coexist but do not intermingle [10]. These addresses are managed by a separate automated system that assists in the deployment of overlays.

Other systems use separate VPN identifiers to distinguish IP address spaces in separate overlays [9]. This assumes that the VPN ID is itself unique, or at least SITN-unique, where no two overlays sharing a node use the same VPN ID. The VI architecture assumes the use (and reuse) of RFC-1918 private address space, because the effort of maintaining SITN-unique VPN IDs is identical to that of maintaining SITN-unique addresses, and the latter can use existing IP encapsulation mechanisms [15][17]. Using IP encapsulation further allows VIs to use existing implementations of application, transport, and network protocols, and existing implementations of forwarding and routing algorithms.

II.2.1. Virtual Hosts

Virtual hosts (VH) are sources and sinks of traffic on a VI overlay, and virtual routers (VR) are transits on a VI overlay. Being a VH or VR is a property of a node of the underlying network, whether host or router. Those nodes may participate as multiple VHs and/or VRs in a single overlay, or may participate as VHs and/or VRs in multiple concurrent overlays. Multihoming is thus a critical component of the overall architecture, as it relates to both hosts and routers [5].

Multihoming in the VI architecture extends the concept of an embedded virtual router, developed earlier to support hosts being on both experimental and production networks [21]. In this earlier model, all host traffic is directed through a phantom internal router, allowing end-to-end traffic to be forwarded through alternate outgoing host interfaces while using a single source address for traffic (Figure 5). For the VI, this implies that all VHs include VR capabilities and are full participants in the routing protocol of the overlay. This challenges certain implementations of routing protocols on base network hosts, such as *gated*, *mrtd*, and *zebra*.

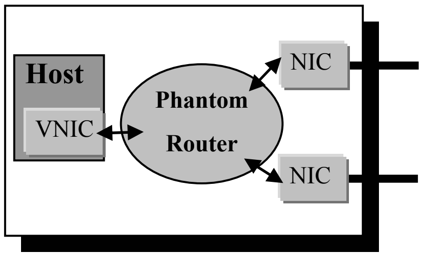

Figure 5 Host as router & internal host

Multihoming is required for all components of the VI, because even a base host with a single VH is necessarily a member of at least two networks – the Internet and the VI overlay. As a result, hosts need routing capabilities, and both VHs and VRs need to partition the interfaces into sets. This latter capability is required for routing protocols to exchange reachability information among the (virtual) interfaces of a single VI overlay, but to also keep the routing of different VIs separate.

Partitioning supports both concurrency and revisitation. Sets of interfaces associated to each overlay enforce per-overlay routing protocol exchanges. This restores the meaning of the ubiquitous universal *inaddr_any* on multihomed hosts, where it must now be associated to an overlay, but need not specify individual addresses therein. The effects of multihoming and partitioning extend throughout the VH and VR, and are also discussed in Section III.

The use of multihoming requires revisiting the strong vs. weak end system model [5]. The model determines whether addresses refer to interfaces or to the host to which they attach. In the strong model, addresses refer to the interface, and incoming packets are checked to see if they match the address of the interface on which they arrive (Figure 6, left). If they do not, they are discarded. This is seldom used in the Internet at the network layer, though it pervades the link layer, e.g., Ethernet packets are accepted only if they are addressed to the arriving interface. The weak model allows arriving packets to match any interface on the host (Figure 6, right).

Although Internet hosts tend to have weak network layer addressing, they also tend to have strong link layer addressing. The VI architecture recognizes this duality, and includes both a virtual link layer and a virtual network layer. The former uses the strong model and the latter the weak. The strong model is required for revisitation.

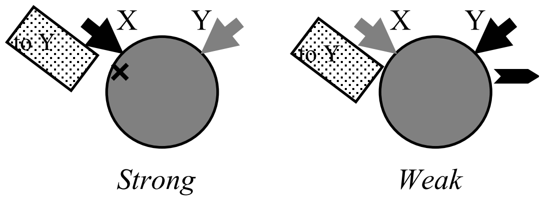

Figure 6 Strong vs. weak model

Other VPN protocols (GRE, PPP, PPTP, etc.) encode the virtual link and virtual network information inconsistently, both in the IP and transport layer headers. VI encodes them separately in distinct IP headers, allowing different layers to enforce the appropriate mechanism as necessary. The VH uses forwarding rules to implement the two-layer encapsulation by weaving packets among tunneling interfaces. The current VI implementation of strong host packet processing uses firewall rules to enforce strong host behavior on link-layer tunneling interfaces.

II.2.2. Virtual Routers

Virtual Routers (VRs) are similar to their base network counterparts, where different overlays are similar to non-interacting autonomous system domains (ASs). Routing information is exchanged among the interfaces of a VI overlay, but not between overlays or with the base network. Forwarding is similarly partitioned, where packets arriving on the interfaces of one overlay go out interfaces of that overlay or are dropped.

Given SITN IP addressing, forwarding partitioning is provided by avoiding default routes, or by using forwarding mechanisms that permit multiple independent defaults (i.e., that explicitly support grouping interfaces into sets). It is further useful if multiple routing daemons (one per VPN) can coexist, or if routing configuration supports explicit grouping of interfaces.

II.2.3. Virtual Links

Virtual links encapsulate packets in additional IP headers. The VI architecture uses two layers of encapsulation for each virtual link, to support revisitation. Consider a VI packet, which consists of data inside an overlay endpoint header. The innermost header indicates the source and sink addresses on the overlay, e.g., A→D in Figure 7. In that figure, base addresses X,Y, Z are used for the end hosts and router, and addresses Q, R, S, T are used for overlay links (OL), and A, B, C, D are used for overlay network addresses (ON).

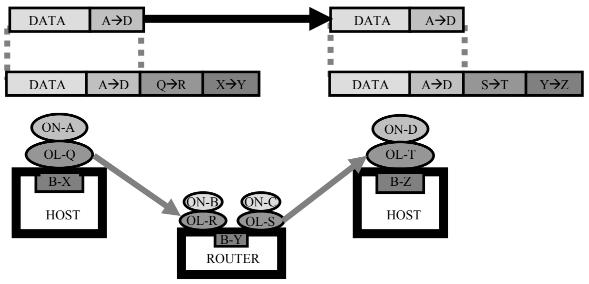

Figure 7 VI hop-by-hop header rewriting

The packet must be wrapped in an outermost header that indicates the source and sinks of the tunnel using the base network addresses, e.g., X→Y, then Y→Z. Note that this outer header is rewritten on each hop, like a link layer.

In the above example, the link layer addresses (Q, R, S, T) are not strictly needed. However, when revisitation is considered, these link addresses are required to distinguish between the Nth and N+1st visit to a node. Such revisitation is show later in Figure 11. Support for revisitation requires that Internet hosts (as noted before) and routers support forwarding based on the incoming interface, as well as the packet header.

Having a separate virtual link and virtual network layer further allows IPsec to be deployed on the virtual link layer independent of application IPsec at the virtual network layer or base network IPsec in the outermost base packet header.

II.2.4. Advanced Issues

There are additional considerations to developing a VI architecture. There are several details involving recursion and revisitation and complexities with IPsec interactions. A system for automating overlay deployment and a language for describing VIs are also discussed. Finally, there are issues with scalability, notably affecting the performance of recursive VIs that can be effectively stacked.

There are two distinct forms of recursion that that the VI architecture supports: control recursion and network recursion (Figure 8). Both are called recursion rather than stacking because there is no strict limit on layering; architecturally, both are recursive structures.

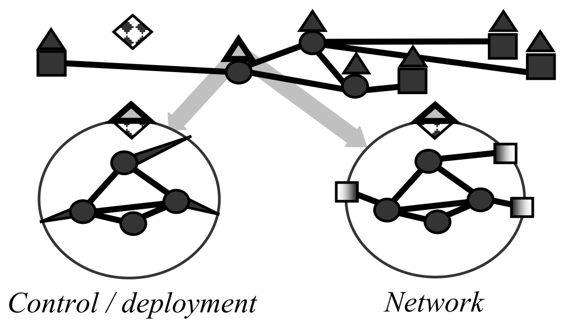

Figure 8 Two types of recursion

Control recursion is akin to compile-time recursion of the VI description language. It allows a compact symbolic representation to be expanded during deployment, allowing divide-and-conquer network management. Network recursion is true stacking of a VI on another VI, where packets on the uppermost VI have additional layers of header encapsulation when traversing inside the upper VI. Network recursion allows intermediate VIs to support advanced network capabilities and provide them as a service to the upper VI, e.g., dynamic routing can be used to support fault tolerance, even when the uppermost VI considers its overlay statically routed.

Figure 8 depicts both kinds of recursion. In this figure, hosts are squares and routers are circles. Each component has a network management daemon, shown as a triangle. One example VI architecture uses a controller daemon per VI, shown as a checkered diamond.

The figure shows how the VI architecture represents recursive embedded VIs as a VR in the upper-layer VI. In control recursion, the management daemon of that router is really the top-half of the controller daemon of the recursed VI (left circle). The interfaces of that router (edge of the circle) are unbound interfaces of routers inside the recursed VI. The result is divide-and-conquer deployment of a flat network structure where the internal recursion topology is visible throughout the upper VI. Packets traversing a control recursion VI have the same number of headers on links inside the recursion as outside the recursion.

In network recursion the recursed VI (right circle) is a VI layered on top of the base VI also modeled as a VR. It has a similar controller daemon, where the top-half of the recursive VI controller daemon plays the role of a router management daemon in the base network. However, packets inside a network recursion VI have additional headers that represent the hops inside the recursion; these hops are not visible in the base network. To the base network, the recursed VI looks exactly like a single virtual router.

Note that the VI models recursion of both kinds as a router, in the former case as a router with unbound interfaces, and in the latter case, as a router in the base network whose interfaces are phantom VHs in the recursive VI (shaded boxes in right circle). The network recursive VI represents its edges as hosts because hosts are the only component of a VI where a packet has a different number of headers when arriving (e.g., from an application) vs. leaving (e.g., out to the network). The hosts are phantom because they do not exist in the inner network; they exist solely to represent the interfaces of the VR in the lower VI.

As noted earlier, VI packets include two overlay headers – one for endpoint network addresses, and one for hop-by-hop link addresses (Figure 9). The application sets the overlay endpoint (typically destination; the source is added automatically), and the VI configuration sets the overlay link and base Internet addresses. When VIs are stacked, at first it would appear that two additional layers of headers would be required for each layer or recursion. However, the overlay link of the upper VI also serves as the overlay network of the lower VI, as shown in Figure 10. This reuse is possible because virtual interfaces use exactly one address, and addresses are never reused where overlays overlap (SITN addressing). Reuse of the base network address in similar fashion is not possible because base interfaces are not necessarily exclusive to a single base address.

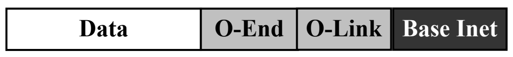

Figure 9 Basic VI header

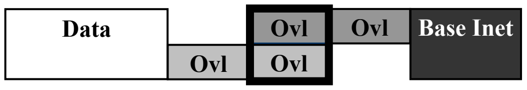

Figure 10 VI on VI header – overlapping addresses

Revisitation allows a VI to be completely decoupled from the components of the base network. Virtual memory already provides this capability, one which has proven invaluable in supporting virtual machines, as well as emulating large-scale systems in software. Similarly, a VI aims to allow overlay networking to be as flexible.

In revisitation, a single node participates multiple times or ways in a single overlay. Figure 11 shows a VI (top) and how it is mapped onto an underlying network. In this case, VH A is mapped onto base host X, and VRs C and E are mapped onto base router Z. Base router Y participates three times – once as VH F, and twice as VRs B and D. Packets sent from A to F will enter Y three times; Y needs a way to distinguish the various visits. The overlay endpoint addresses do not change (A to F), and for some paths even the hop-by-hop base addresses do not change (Z to Y). Virtual link addresses allow Y to distinguish between visits from, e.g., hops C' to D' from E' to F' (/'/ indicating virtual link).

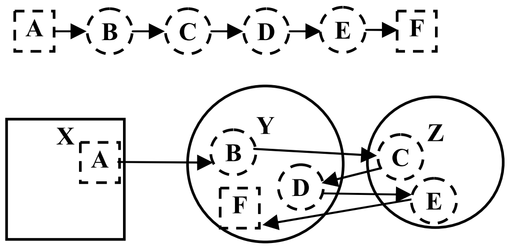

Figure 11 Revistation (VI above, as deployed including revistation below)

A language is under development that describes VIs and enables their automated deployment. It builds on the work of the X-Bone and VNS, both of which included languages for describing overlays [12][20]. The VI language has additional capabilities for both forms of recursion as well as revisitation.

There are no particular limits to the scalability of the VI architecture. Control recursion allows divide-and-conquer recursive deployment of large-scale flat topologies, and network recursion allows similarly scalable deployment of layered networks. The first VI layer adds two IP headers each of which may be compressed down to a single byte for frequent traffic [11]. Each header decreases the effective MTU of the underlying network, but with fragmentation this can be ignored – except for its effect in lossy environments. When upper layer VI packets are fragmented to fit into lower layer MTUs, and fragments are lost, the VI becomes very inefficient and may be effectively disabled. Tests to determine the limit of effective recursion are underway, and have indicated that there is little effect for the first two layers of stacking, even though the base MTU has decreased by over 10% (by 60 bytes). The focus of the VI is capability over performance, verified by experience with a variety of users.

III. IMPLICATION/CONSEQUENCES

The VI augments the Internet architecture with support for virtualization. The implications of this augmentation are discussed, including the effect on network architecture, component architecture, on protocols, as well as the opportunities it affords for automated network management.

Network nodes require a system for associating sets of network interfaces of respective VIs with each other. Routers need to contain routing protocols within each component virtual router, and virtual hosts need to map *inaddr_any* to meaningful subsets of addresses.

There are other issues of what is called "network reentrancy" – aspects of networked applications that require special attention to support concurrency. Most can be provided by judicious programming; they include:

1. avoid use of *inaddr_any*; bind to explicit lists of addresses instead
2. avoid use of directory names or login names that cannot be overridden, e.g., log files, configuration files, user IDs/group IDs, etc.

3. bind only to the most specific addresses and ports possible

Providing subsets of addresses also may require providing authorization to modify only one of the subset, i.e., fine-grained configuration control.

As previously noted, revisitation requires two layers of encapsulation in which one layer is strong and one weak. Forwarding must enforce the strong layer constraints, which typically requires policy routing, where the incoming interface is context for the subsequent forwarding decision. In current host operating systems, policy routing can be emulated using firewall rules, but this should be replaced with an integral policy routing and forwarding framework.

Network recursion requires a new protocol, one which combines some of the aspects of BGP and others of ARP. VI recursion models the upper VI as a virtual router in the lower VI. When packets arrive at that router in the lower VI, they expect to exit on the appropriate interface; this function is normally internal to the router, and performed internally by a mechanism akin to ARP.

Inside the recursive VI, packets are arriving at a VH on one side of the network and must be delivered to a particular VH on the other side – the exit-VH. Determining the proper exit-VH in a recursive VI is similar to determining the exit router of an AS using BGP. As a result, network recursion requires a merged variant of BGP and ARP. Preliminary analysis indicates that this is less of a hack than a generic mechanism that can be used to describe a variety of 'successive recursive resolution' systems, including Google, the DNS, IP forwarding, BGP routing, and ARP.

Finally, the VI enables extensive automation, and benefits greatly from its use. VIs are built on base IP networks, so they can assume underlying network connectivity for management and control. Coordinated systems for configuring VIs ensure SITN addressing to enable effective use (and reuse) of RFC1918 addresses, allowing virtualization without requiring support for new protocols [10][17]. As a result, VIs support existing applications, protocols, and operating systems with modifications needed only to support multi-pass encapsulation and forwarding.

IV. Prior and related work

Virtual Internets are a general virtual extension to the basic Internet architecture [3][5]. They virtualize all components of the Internet, providing the same functionality inside each virtual layer as exists in the base Internet.

VPNs extend portions of existing networks to remote sites, usually incrementally [19]. They do not deploy a complete, virtualized network, but rather focus on attaching hosts to existing, private networks over the public Internet. VPNs may use tunnels, or may use other means, e.g., tags, to separate private traffic from public, and lack support for virtual routing. PPVPNs are the complement of a VPN [6]; they consist of a virtual core with translation boxes at the periphery. The core supports virtual routing, but hosts are not part of the virtual network, and cannot participate in multiple PPVPNs concurrently.

PPVPNs also include VLANs, using a link layer to develop a virtual infrastructure. Link layers are notoriously non-scalable, imposing limits on response time and often requiring broadcast emulation for address discovery. By contrast, network layer virtual networks avoid those issues by using hop-by-hop forwarding, well-developed routing algorithms, and distributed address translation mechanisms (e.g., DNS).

Peer networks, as was noted earlier, recapitulate network architectures at the application layer [14]. They use application layer tunnels, and provide virtual routing at the application layer. Because each peer network is based on a separate architecture, it is difficult for a single application to participate in multiple peer networks. By contrast, a virtual Internet uses the same API at all layers, allowing a single application to use whatever layer is needed.

Virtual Internets are generalizations of the static, manually-deployed m-Bone, A-Bone, and 6-bone tunneled backbones [1][4][8]. VIs emphasize protection and abstraction, rather than optimization, as with RONs and Detour [2][18]. VI optimization can be achieved by replacing a portion of the general purpose architecture, just as a realtime OS can be achieved by replacing the scheduler in a conventional OS.

VIs enable automated support for multiple, concurrent virtual networks. They have already been used to support shared testbeds, automated deployment of applications, and management of overlapping address spaces, some of the goals of PlanetLab [16].

V. Status and Future Work

The VI architecture has been implemented in the X-Bone system and is under internal evaluation [22]. Concurrency and multilayer tunneling, as well as basic VI deployment and coordination are provided by a distributed network management system. The basic aspects of network recursion and dynamic routing, including the unified BGP/ARP mechanism, as well as revisitation have been implemented and tested, but not yet automated. Control recursion and extensions to the VI language to support recursion and revisitation are currently under development.

Many of the implications of the VI architecture have resulted in related projects: using recursion to support fault tolerant dynamic routing on hidden VI layers (DynaBone), using overlays to provide alternate geographic delivery (GeoNet), and developing OS extensions to support fine-grained per-overlay configuration control (NetFS), as well as forming the basis of the X-Bone overlay deployment and management system [20][22].

There are a number of design decisions that simplified the development of the VI. First, VIs are based on IP, assuming IP and providing IP, avoiding equivalent but less ubiquitous alternatives of GRE, PPP, and the VPN-ID, all under consideration for PPVPNs [6]. Using IP results in using RFC1918-style addresses with SITN avoidance, where addresses are not reused where overlays share mutual resources [10][17].

The VI architecture avoids issues of optimization, e.g., to use overlays to provide alternate routing services, as done with RONs and Detour [2][18]. This is a consequence of being truly virtual, where there can be no assurance that the path taken by the overlay is any better than the path of the underlying base network.

VI. References

[1] 6-Bone URL – www.6bone.net

[2] Anderson, D., Balakrishnan, H., Kaashoek, M. F., Morris, R., “Resilient Overlay Networks,” Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[3] Baker, F., “Requirements for IP Version 4 Routers,” RFC 1812, June 1995.

[4] A-Bone URL – www.isi.edu/abone

[5] Braden, R., ed. “Requirements for Internet Hosts – Communication Layers,” Internet RFC 1122, IETF, Oct. 1989.

[6] Callon, R., et al., “A Framework for Provider-Provisioned Virtual Private Networks,” (work in progress).

[7] DynaBone web pages – www.isi.edu/dynabone

[8] Eriksson, H., “MBone: The Multicast Backbone,” Communications of the ACM, Aug. 1994, pp.54-60.

[9] Fox, B, Gleeson, B., “Virtual Private Networks Identifier,” RFC-2685, September 1999.

[10] Gross, P., (ed), “Choosing a "Common IGP" for the IP Internet,” RFC-1371, October 1992.

[11] Jacobson, V., “Compressing TCP/IP Headers,” RFC-144, January 1990.

[12] Lim, L., Gao, J., Ng, T., Chandra, P., Steenkiste, P., Zhang, H., “Customizable Virtual Private Network Service with QoS,” Computer Networks, July 2001, pp. 137-152.

[13] NetFS web pages – www.isi.edu/netfs

[14] Oram, A. (ed.), Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology, O'Reilly, Sebastapol CA, 2001.

[15] Perkins, C., "IP Encapsulation within IP," RFC-2003, Oct. 1996.

[16] PlanetLab – www.planetlab.org

[17] Rekhter, Y., et al., "Address Allocation for Private Internets," RFC-1918, Feb. 1996.

[18] Savage, S., et al., "Detour: a Case for Informed Internet Routing and Transport," IEEE Micro, pp. 50-59, v 19, n 1, Jan. 1999.

[19] Scott, C., Wolfe, P., Erwin, M., Virtual Private Networks, O'Reilly & Assoc., Sebastapol, CA, 1998.

[20] Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," Computer Networks, July 2001, pp. 117-135.

[21] Touch, J., Faber, T., "Dynamic Host Routing for Production Use of Developmental Networks," Proc. ICNP '97, Atlanta, Oct. 1997, pp. 285-292.

[22] X-Bone web pages – www.isi.edu/xbone