Virtual Internets[1]

Joseph D. Touch, Yu-Shun Wang, Lars Eggert
USC/Information Sciences Institute
July 1, 2002

Abstract— **A Virtual Internet is a network of IP-tunneled links interconnecting virtual routers and virtual hosts, providing full Internet capabilities at a virtual layer. They support protection, concurrent sharing, and abstraction, just like their virtual memory counterparts. A VI is a superset of the current Internet architecture, and supports revisiting a host/router multiple times in a single overlay, as well as recursive (layered) overlays. The VI architecture provides a generic, unified mechanism that supports VPNs, peer to peer networks, and more specialized overlay systems.**

1. Introduction

A virtual Intenet (VI) is an Internet composed of IP encapsulation tunnels over an existing Internet (Figure 1). Like VPNs, it provides a layer of security and isolation over an existing network [15]. Unlike VPNs, a VI virtualizes an entire network, rather than a single link.

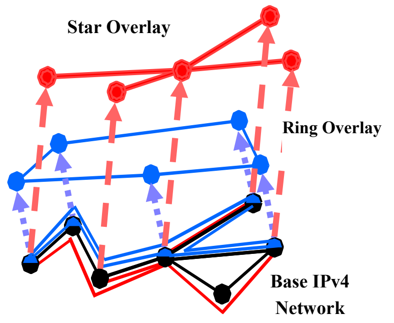


Figure 1 Multiple virtual Internets

VIs are a more generic version of the virtual backbones used to deploy new protocols, notably the m-Bone and 6-Bone, as well as more recently the A-Bone [1][4][10]. Like these backbones, a VI uses tunnels as virtual links. VIs extend the concept of these core backbones with links to hosts, creating a virtual topology completely isolated from the underlying Internet. This VI capability has been implemented in the X-Bone overlay management system [16][19].

The current state of distributed shared network testbeds emulates the state of operating systems in the 1960's [6]. Although pegboards have been replaced with web reservation systems, individual components are explicitly allocated in advance, along fixed time slots. Users request host-A, host-B, and host-D from 3pm-5pm PDT. Such exclusive reservation systems are inefficient and cumbersome, especially among groups spread across multiple time zones. Overlays are limited by the size that can be manually managed. Furthermore, there is no enforcement on the reservations; users can collide.

Peer networks are an application layer alternative to overlay networks [12]. They largely recapitulate the development of their network layer equivalents, most recently reinventing solutions to split-horizon routing and packet TTLs. At first they may appear to provide a convenient playground in which to develop new protocol capabilities, they are no easier to deploy, and often much more difficult to coordinate and control, than their network layer counterparts.

A virtual Internet provides to a network what a multitasking operating system and virtual memory provide to processes – a mechanism for concurrent sharing of resources, protection from overlap, and abstraction to a simple programming model[2]. Like VM, applications running on different VIs do not interact. Multiple applications on a single machine can concurrently participate in different VIs. The address space of different VIs avoids collision much the same was as in VM, by way of a coordination system. VIs also provide applications and users with a simple view of network topology, one that may be convenient to their architecture, easy to program to, or more direct to design protocols to interact with (Figure 2).

[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.

[2] Here VM focuses on indirection, memory management, etc. There is no VI equivalent of swapping.

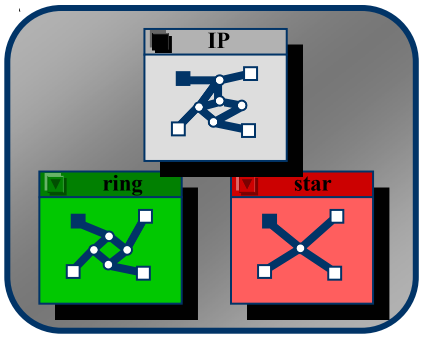


Figure 2 User's view of VIs

Virtual Internets also provide revisitation and recursion. Revisitation allows a single network component to emulate multiple virtual components, the VI equivalent of multiprocessing. Recursive operating systems and virtual memory allow one OS to be emulated inside another, e.g., VM-ware. Recursive virtual Internets allow testbeds inside of other testbeds, for example.

2. Virtual Internet Components

A Virtual Internet, like its real counterpart, is composed of links connecting routers and end hosts. Virtual links are provided by tunnels, the identifying aspect of virtual networks. Like tunnels as virtual links, virtual routers are emerging as an established component of the current Internet architecture. Virtual hosts are somewhat more complicated, requiring the resolution of long-standing issues of multihoming and network resource partitioning. Further, like its VM equivalent, a VI requires a management system to coordinate sharing and establish protection; this system, like virtual hosts, is new to virtual Internets.

2.1 Tunneled links

Tunnelling is the hallmark of a virtual or overlay network. Packets on tunneled links are encapsulated in the payload of other packets, and the outer packet header is used exclusively for routing between the tunnel endpoints. The protocol of this outer header determines at what layer the tunneling occurs, and its extent and interoperability. If the outer layer is Ethernet, the packet isn't going to get very far unless both endpoints are on the same LAN. As a result, most tunneling uses IP or an application protocol inside IP, effectively using two layers (application and IP) for encapsulation.

Common tunnels include IP, UDP, TCP, GRE, PPP, and PPoE. UDP and TCP are most useful for tunneling through NATs or deploying new protocols (e.g., peer nets) with application layer routing. For most other protocols, the primary encapsulation is the outermost IP header, and the additional header (GRE, PPP, PPoE) provides space for additional information.

Virtual Internet tunnels use two explicit layers of IP encapsulation, to provide virtual versions of the headers used in the real Internet [16][17]. Although most Internet processing uses the IP header, there are a substantial number of protocols that rely on a link layer header, notably ARP, and the rules that govern the behavior of routers and end hosts. VI's outermost IP encapsulation header provides a virtual physical layer, the next IP encapsulation header provides a virtual link layer, and its innermost (originating) header is the virtual network layer. Together, these three layers (original, and two additional via encapsulation) provide a virtual equivalent of full Internet capabilities.

The use of two layers is needed to support revisitation, where a single real node (host or router) participates multiple times in the same virtual Internet. This allows a set of 10 routers to emulate a ring of 200 routers, each router being visited 20 times in the same overlay.

Virtual Internets require some level of tunneling using encapsulation. Multipath routing and related techniques (e.g., MPLS, VLANs), exclusive of encapsulation, is not sufficient to support virtualization, largely because it is already a capability inside any given Internet, virtual or otherwise.

2.2 Virtual router

A virtual router forwards tunneled packets inside a virtual network. These packets are exchanged on some distinct subset of its virtual interfaces, namely those on the corresponding virtual network. A VR maintains internal routing tables computed from routing algorithms, just like a real router [3], except that each table and algorithm is distinct to a particular virtual network.

The concept of a virtual router is similar to that of border gateway routers, where multiple routing and forwarding engines exist inside a router. The difference is in the strict partitioning of the interfaces, tables, and algorithms. In border routers, information is shared across partitions, whereas in a virtual router, each set is distinct.

A real router can contain multiple virtual routers, given certain architectural support [17]. In particular, the most recent incoming interface of a packet (tunneled or not), must be maintained and used both for routing (as it conventionally already is) as well as for forwarding – e.g., to determine which virtual router does the forwarding. Current router architectures use the single, physical, incoming interface information for route computation, but not always for forwarding. Often, only the physical interface of an incoming packet is maintained; tunneled packets lose their tunnel identity. Virtual routing requires the entire chain of interfaces, one per decapsulation step as well as the physical incoming interface, or at least the most recent virtual interface visited. This, as well as table isolation and interface partitioning, support for virtual routers is emerging in a form compatible with a virtual Internet.

2.3 Virtual host

One of the key challenges of a virtual Internet is the host model. Many virtual network systems (PPVPNs, backbone overlays) ignore the end host, or treat it as a transient component in the architecture [2][7][11]. A true virtual Internet must extend to the end host as a first-class member.

The primary issue for virtual hosts is multihoming [5]. These hosts have at least one real interface to the Internet, and at least one virtual interface to a virtual Internet. Although routers have always been implicitly multihomed, the current Internet support for multihiming has been sporadic at best. Many emerging systems for multihoming focus on maintaining persistent transport layer connections by providing a layer of indirection and a coordination mechanism when connections move [8].

Instead, a true virtual Internet host contains environments which each appear as a single, independent host. Inside these environments, virtual network connections are established and coordinated just like on a real host. Hosts that connect to multiple virtual networks contain separate, independent environments that are connected to specific external virtual networks via an internal router (Figure 3) [18]. This allows existing dynamic routing among the virtual networks to support fault-tolerant, persistent connections without the need for explicit transport layer support.

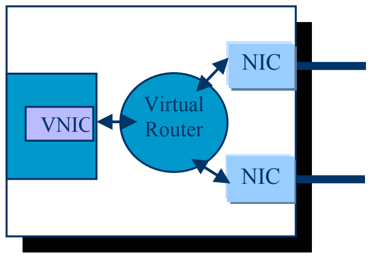


Figure 3 Host as router & virtual host

As a result, all hosts in a virtual Internet include some capabilities of a router. Further, virtual Internets always terminate at these virtual hosts; in this model, there are no ‘escalators’ and ‘elevators’, which translate packets from non-virtual hosts onto a virtual network (or if they are, they are considered remote aspects of those non-virtual hosts).

Support for virtual hosts requires revisiting the strong/weak end system model. The network layer of the current Internet uses the weak end system model, where packets arrive on a particular interface, and are accepted if they match the IP address of any interface (Figure 4, left). However, most current Internet link layers rely on a strong end system model, where packets arriving on an interface are accepted only if they match that incoming interface’s address; packets to other addresses are discarded (Figure 4, right).

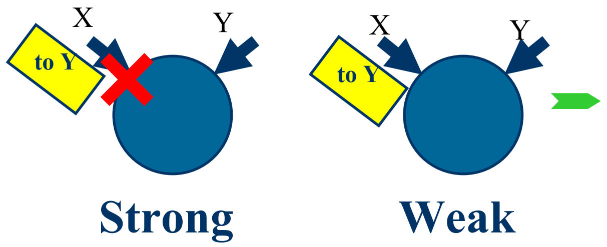


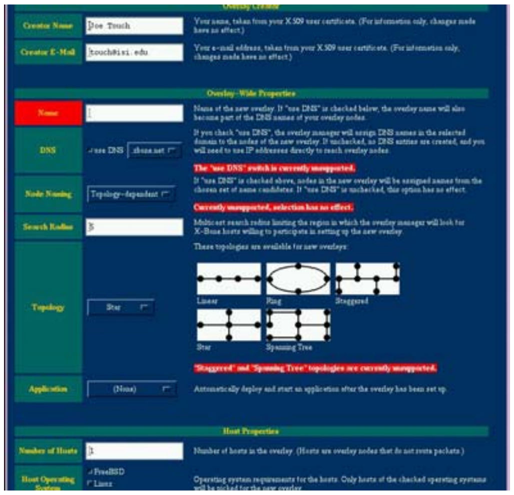
Figure 4 Strong vs. weak end system model

The VI model requires that the virtual link layer enforce the strong end system model, and that the virtual network layer allow the more generous 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.

2.4 Management system

Virtual memory is more than a level of addressing indirection; it relies on memory allocation, a paging algorithm, and a scheduler to support concurrent, independent use of memory. Similarly, a virtual Internet uses indirection (tunneling), together with overlay allocation, deployment, and management mechanisms.

The X-Bone represents one such system for the coordinated deployment and management of virtual Internets [16]. It provides an API for overlay operations, as well as a graphical user interface for user-level control (Figure 5). The goal is for applications to deploy overlays directly, as when processes are spawned by other processes; in the meantime, the GUI allows user-directed deployment, akin a Unix shcll or windowed desktop. In the GUI, users request basic topologies (line, ring, bus, star, etc.), and particular capabilies (IPsec, emulated delay, etc.).


Overlay Creator

Creator Name	Joe Touch	Your name, taken from your X.509 user certificate. (For information only, changes made have no effect.)
Creator E-Mail	touch@isi.edu	Your e-mail address, taken from your X.509 user certificate. (For information only, changes made have no effect.)

Overlay-Wide Properties

Name		Name of the new overlay. If "use DNS" is checked below, the overlay name will also become part of the DNS names of your overlay nodes.
DNS	use DNS xbone.net	If you check "use DNS", the overlay manager will assign DNS names in the selected domain to the nodes of the new overlay. If unchecked, no DNS entries are created, and you will need to use IP addresses directly to reach overlay nodes. The "use DNS" switch is currently unsupported.
Node Naming	Topology-dependent	If "use DNS" is checked above, nodes in the new overlay will be assigned names from the chosen set of name candidates. If "use DNS" is unchecked, this option has no effect. Currently unsupported, selection has no effect.
Search Radius	3	Multicast search radius limiting the region in which the overlay manager will look for X-Bone hosts willing to participate in setting up the new overlay.
Topology	Star	These topologies are available for new overlays: Linear, Ring, Staggered, Star, Spanning Tree. "Staggered" and "Spanning Tree" topologies are currently unsupported.
Application	(None)	Automatically deploy and start an application after the overlay has been set up.

Host Properties

Number of Hosts	3	Number of hosts in the overlay. (Hosts are overlay nodes that do not route packets.)
Host Operating System	FreeBSD, Linux	Operating system requirements for the hosts. Only hosts of the checked operating systems will be picked for the new overlay.

Figure 5 X-Bone web-based GUI

As with processes in a multitasking OS, applications in a virtual Internet are subject to certain constraints. Process memory space starts at zero, and the more generic processes use position independent code, and are reentrant to support stack-based recursion. Similarly, virtual Internet applications are said to be *network reentrant* if they attach only to explicit subsets of interfaces (avoiding INADDR_ANY), and use only relative usernames and filesystem paths (e.g., for logs, configuration files, etc.). In both cases, these constraints are driven by the desire for concurrent sharing by multiple copies of an application.

3. Recent results

The X-Bone is an implementation of this virtual Internet architecture [16]. It the implementation supports multiple, concurrent overlays on the same physical machines, and the architecture has been tested to demonstrate revisitation and recursion. The X-Bone uses only IP encapsulation, requiring no new protocols, no application recompilation or relinking, and no OS extensions. It supports the use of existing protocols and applications in deployed VIs, e.g., dynamic routing, multicast (including teleconferencing), IPsec, and network mapping and monitoring. Because it relies only on IP, it supports all current and emerging Internet standards in the virtual Internet as soon as they are supported in the base Internet.

The X-Bone architecture has been used to develop multilayer overlays for fault tolerant networks, called DynaBones (Figure 6) [9]. A DynaBone is just an application of a set of concurrent, parallel virtual Internets (called *innerlays*) providing alternate, secure transit paths for another, higher-layer virtual Internet (called an *outerlay*). The outerlay isolates the details if the innerlays and the decision as to which innerlay to use from the applications. The innerlays and outerlay are distinct virtual Internets that together provide new capability.

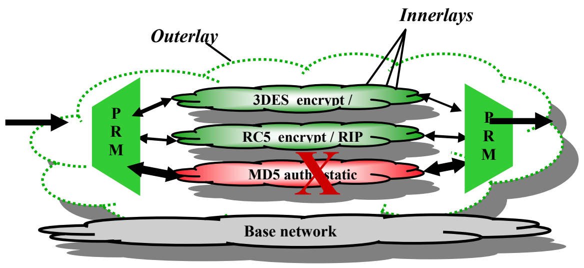


Figure 6 Layered VIs for security

4. 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 [15]. 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 is the complement of a VPN [7]; it consists 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.

Peer networks, as was noted earlier, recapitulate network architectures at the application layer [12]. 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][11]. VIs emphasize protection and abstraction, rather than optimization, as with RONs [2] and Detour [14]. 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 focus on the automated support for multiple, concurrent virtual networks. They have already been used, via the X-Bone, to support shared used of testbeds, automated deployment of applications, and management of overlapping address spaces, some of the goals of PlanetLab [13].

5. 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 – Comunication Layers,” Internet RFC 1122, IETF, Oct. 1989.

[6] CAIRN URLs – www.isi.edu/cairn

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

[8] Coene, L., “Stream Control Transmission Protocol Applicability Statement,” RFC-3257, April 2002.

[9] DynaBone URL – www.isi.edu/dynabone

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

[11] 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.

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

[13] PlanetLab – www.planetlab.org

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

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

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

[17] Touch, J., Eggert, L., Wang, Y., “An Architecture for Layer 3 Virtual Networks”, (work in progress).

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

[19] X-Bone URL – www.isi.edu/xbone