A Global X-Bone for Network Experiments

Joseph D. Touch, Yu-Shun Wang, Venkata Pingali, Lars Eggert*, Runfang Zhou, Gregory G. Finn
*USC/ISI and NEC Labs**
{touch,yushunwa,pingali,rzhou,finn}@isi.edu and lars.eggert@netlab.nec.de

Abstract

A global Internet overlay testbed is being deployed to support the distributed, shared use of resources for network research. The Global X-Bone (GX-Bone) augments the X-Bone software system, enhancing its coordination mechanisms to support deployment of local overlays to world-wide, shared infrastructure. The GX-Bone is based on the X-Bone's Virtual Internet Architecture which extends the Internet for both concurrent, parallel and recursive overlays, and provides decentralized, automated deployment and management. GX-Bone supports host virtualization through the NetFS file system, granting individual users compartmentalized access and control of host and router configuration, and the DataRouter extension to IP loose source routing that supports application control of network-layer forwarding. GX-Bone can be installed on user-modified kernels, uniquely supporting both conventional kernel-level protocol development and coordinated global infrastructure sharing.

1. Introduction

X-Bone is a system for deploying and managing Internet overlays [17][20]. It coordinates the configuration and management of virtual networks, enabling shared use of network resources (Figure 1). The Global X-Bone extends the X-Bone implementation from a stand-alone software system for local experiments to a global infrastructure for wide-scale network research.

Overlays can be used for isolation, concurrency, and abstraction. They protect traffic, allowing new protocols to be tested, and were originally used to protect multicast IP addresses from leaking onto the conventional Internet. Overlays isolate different protocols, such as was used to deploy IPv6 incrementally over the current IPv4 Internet. Overlays also allow network components to be shared, supporting network concurrency akin to multiprocessing. This allows concurrent network experiments to share core infrastructure, as the 6-Bone and M-Bone currently do. Finally, overlays provide abstraction of topology, allowing experiments with routing protocols on rings, where, e.g., the physical topology is a star, a ring, or of arbitrary design. This allows the network topology to reflect forwarding decisions, as is done with peer-to-peer architectures, e.g., based on hypercubes or Plaxton trees [16].

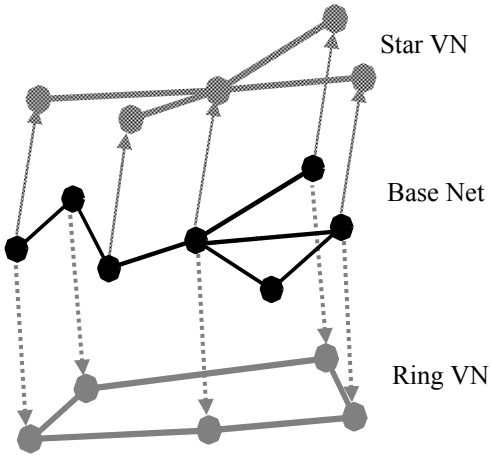

Figure 1 Multiple virtual Internets

The X-Bone applies a general architecture for network virtualization to the Internet [20][22]. This architecture supports *concurrence*, *recursion* and *revisitation*. Concurrence allows deployment of multiple, parallel concurrent overlays. Recursion enables deployment of overlays inside other overlays. Revisitation enables reuse of the same node in a single overlay more than once. The core set of capabilities has been implemented as a software system that allows programmatic deployment of overlays through an XML interface; a web interface for user-directed deployment is also available.

The X-Bone code has been available since 2000 as both a FreeBSD port and a Linux RPM. It has been used in numerous individual deployments to support overlay and application experiments and the development of advanced virtual networking architectures. Although the X-Bone architecture already supports global resource discovery, each of these installations has operated largely independently. This remains a key feature of the X-Bone system – each installation may remain completely decentralized in both management and operation. No global coordination is required.

An X-Bone overlay contains both *named* and *unnamed* resources. The current X-Bone uses an expanding ring multicast search to discover unnamed resources. Named resources must be specified by IP address or DNS name.

Although this simple mechanism is sufficient for local testbeds or pre-coordinated global experiments, it does not easily support a global testbed. On a global scale, it is inefficient to locate resources using multicast alone. The existing implementation hence did not support the needs of a global community of X-Bone users that share their resources to create large-scale, distributed testbeds spanning numerous administrative domains.

The *Global X-Bone,* which we call *GX-Bone*, is an effort to support this worldwide X-Bone user community in resource discovery and sharing. It augments the current X-Bone software with a centralized registry, where members publish information about resources they are willing to share. Access control is an integral part of this registry; members may indicate lists of users with whom they wish to share each resource.

The remainder of this document presents a brief overview of the Virtual Internet architecture that is the basis of the X-Bone, and the X-Bone system itself. It then describes the extensions that form the Global X-Bone, and the advanced networking capabilities that are being added to X-Bone nodes to address deficiencies of other current overlay systems.

2. A Virtual Internet

A *Virtual Internet* (VI) is a virtual version of the Internet. Just as the Internet is a graph of hosts and routers connected through links, in a Virtual Internet, *virtual* hosts and routers are connected by IP-encapsulation tunneled links over the existing Internet.

A VI generalizes the tunneled backbones that helped deploy multicast (M-Bone), IPv6 (6-Bone), and Active Networks (A-Bone.) These testbeds supported wide-scale networking research by enabling the testing and incremental deployment of new protocols on existing infrastructure [1][4][9]. Unlike those interim solutions, VIs support persistent partial deployments of new capabilities.

The *VI Architecture* (VIA) extends the Internet model composed of data sources and sinks (hosts), data transits (routers or gateways), and links between them [22]. The VI virtualizes each of those components, resulting in *virtual hosts* (VHs), *virtual routers* (VRs), and *virtual links.* The latter are typically encapsulation tunnels. The VI architecture has three basic tenets:

TENET 1. Internet-like: VIs are composed of VRs and VHs connected by encapsulated tunnel links, emulating the Internet

TENET 2. All-Virtual: VIs are completely virtual, decoupled from their base network

TENET 3. Recursion-as-router: A VI recurses when some of its VRs are VI networks (the inner VI is modeled as a VR)

A number of corollaries follow from these tenets:

Corollary 1-A: Virtual hosts increase or decrease the number of headers on a packet

Corollary 1-B: Virtual routers do not change the number of headers on a packet

Corollary 2-A: VIs support concurrence

Corollary 2-B: VIs support revisitation

These tenets and corollaries are the basis of VIA. They evolved out of analogies between the VIA and multiprocessing and virtual memory (VM). Like both multiprocessing and VM, VIA allows multiple parties concurrent shared, protected use of a single resource in virtual ways. Like VM, VIA can use a small number of physical resources (memory page frames, network interfaces, respectively) to emulate a larger number (memory pages, virtual interfaces) by revisitation (swapping, known as revisitation in VIA). All can be layered recursively, and all provide a simpler, uniform abstract programming interface.

3. The X-Bone

The X-Bone is a system for the dynamic deployment and management of Internet overlay networks [17][20][27]. Overlay networks are used to deploy infrastructure on top of existing networks, to isolate tests of new protocols, partition capacity, or present an environment with a simplified topology. Current overlay systems include commercial virtual private networks (VPNs), and IP tunneled networks (M-Bone, 6-Bone) [1][9]. The X-Bone system provides a high-level interface where users or applications request DWIM (do what I mean) deployment, e.g.: *create an overlay of 6 routers in a ring, each with 2 hosts*. The X-Bone automatically discovers available components, configures, and monitors them.

The X-Bone system allows different applications on the same end host or router to be associated with different overlay networks. For example, a single generic network mapping utility on one host might have different views of the network depending on whether it was attached to the base network or one of the overlays (Figure 2).

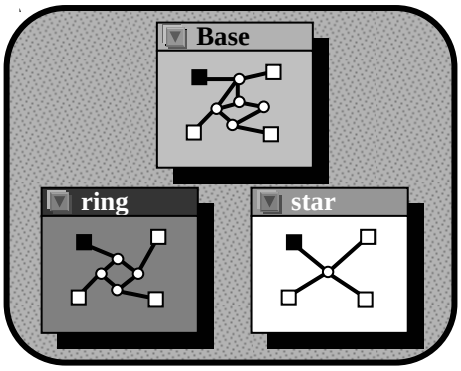

Figure 2 User's view of overlays

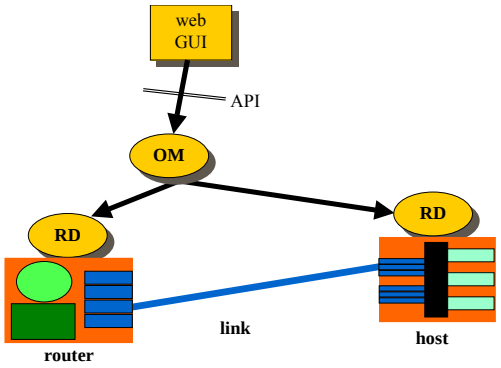

Figure 3 X-Bone architectural components

Some overlay systems require OS and/or application modifications, restrict the number of overlays a router or host can participate in, or require manual component configuration. The X-Bone requires no specific OS or application modifications and only basic IP in IP encapsulation, and uses existing implementations of IP services such as dynamic routing, name service, and other infrastructure. Finally, the X-Bone virtualizes the current Internet architecture to support overlays, and supports stacking (recursion) of overlays for fault tolerance and capacity sub-provisioning for experiments.

Within each overlay, the X-Bone provides a completely standard networking interface that includes, for example, network interfaces, routing tables, and firewalls. Applications continue to interact with the virtualized versions of these mechanisms that are part of the overlay abstraction just as they interact with the regular, physical interfaces. This capability enables experimentation with advanced networking applications within a global, virtual network. Furthermore, the X-Bone supports experimentation with kernel-level networking modifications.

The X-Bone uses a two-layer tunnel mechanism, rather than the single layer used in conventional overlays. It is this two-layer scheme which supports stacked overlays, as well as permitting use of unmodified applications and network services inside a deployed overlay. It also permits network resources (hosts, routers) to participate multiple times in a single overlay, and is the only known overlay system that integrates both IPsec support and dynamic routing [18].

3.1. X-Bone Review

The X-Bone is a distributed system composed of *Resource Daemons* (RDs) and *Overlay Managers* (OMs), with a graphical user interface (GUI) and a more direct API. These components are shown in Figure 3. The functions of the RD and OM have been incorporated into a single daemon, but operationally they can be discussed as distinct units.

OMs deploy overlays; a user creates an overlay by sending a request to an OM, either via a web-based GUI (Figure 4) or by sending an XML message directly to the OM API. Each overlay is coordinated by a single OM. Large overlays can be created by divide-and-conquer, where a single OM will fork sub-overlay requests to other OMs. Fault tolerance can be achieved by replicating state in multiple backup OMs. Both of these latter capabilities (recursion, fault tolerance) are supported in the X-Bone architecture, though not yet implemented in current releases.

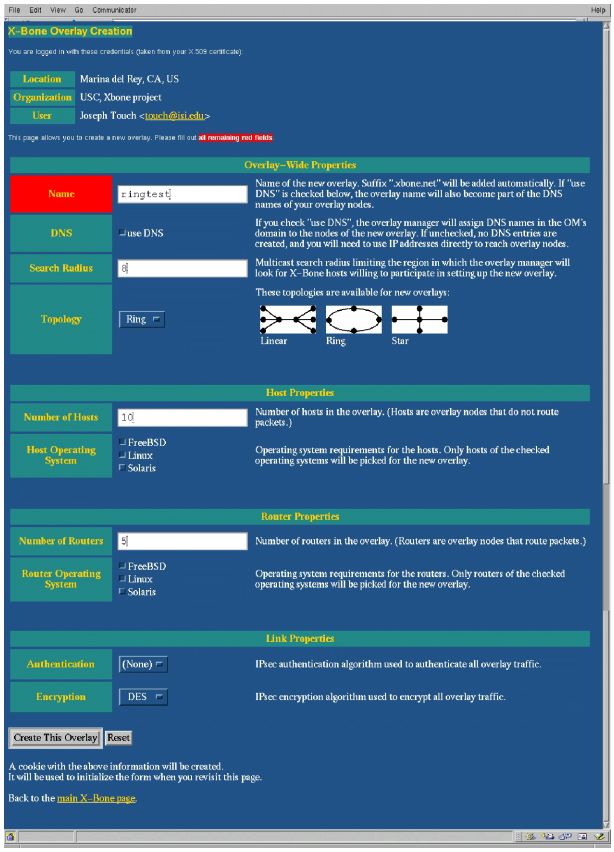

Figure 4 X-Bone graphical user interface

An OM creates an overlay in phases, using various mechanisms to locate the specific resources required for its construction. These discovery mechanisms include the existing multicast ring-search (shown in Figure 5), or the new registry that is part of GX-Bone. The overlay request is translated to an invitation, and the invitation is transmitted using UDP. An invitation indicates a set of simple conditions, e.g., a specific set of host operating systems, bandwidth requirements, etc. Invitations currently fit in a single UDP packet; where they do not, IP's automatic fragmentation and reassembly is utilized.

RDs are daemons that configure and monitor the resources of routers and hosts. RDs listen for UDP invitations, and respond when their capabilities, available resources and permissions match. The RDs respond with UDP messages, indicating their willingness to participate in an overlay, and their capabilities (protocol version, OS type, etc.). The OM arbitrarily selects a suitable subset from among the responding RDs and opens TCP/SSL connections to each chosen RD. The OM determines configuration information, such as tunnel endpoint addresses and routing table entries, and sends specific configuration information to each RD. Subsequent overlay actions initiated by the OM include keep-alive pings, liveliness and status requests, and modifying or dismantling configurations.

TCP/SSL secures the reliable configuration channel only. The X-Bone supports S/MIME authentication to secure invitations; responses need not be secured, because they will be validated by subsequent TCP/SSL connections to the invited nodes, which will confirm their availability at that time anyway.

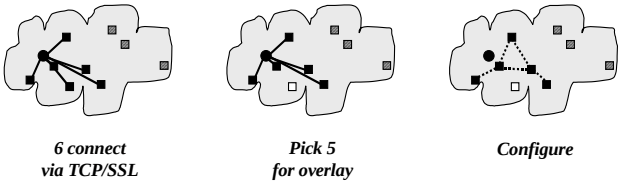

Figure 5 Responding to invites, selecting, and configuring the overlay

3.2. Distributed System Implementation

The X-Bone is a distributed system, where each overlay is coordinated by one OM, and each OM instructs RDs and possibly other OMs how to deploy and coordinate an overlay. A single overlay may be replicated at one or more other additional OMs, but at any given time, exactly one OM is managing each given overlay. A single OM usually manages many overlays.

Users decide which OM manages a given overlay when the overlay is deployed, by which OM receives the initial request. Delegated sub-overlays are each managed at exactly one other OM, and these sub-OMs report back to the primary OM. RDs execute all local commands.

The X-Bone is implemented in Perl, where the RDs run as *root* on local resources (hosts, routers) or on buddy-hosts from which they issue privileged configuration commands (e.g., for Cisco routers).

3.3. GUI/API for Configuration Control

The X-Bone includes two separate APIs, XB_API for overlay configuration requests and replies (TCP reserved port 2165), and XB_CTL for issuing the generic resource configuration commands (UDP and TCP reserved privileged port 265). XB_API uses XML for a web-based front-end, to enable human-readable overlay requests. This control interface includes support for user-specified netlists, i.e., network connectivity lists, to indicate specific topologies. It also supports a small set of preprogrammed topologies: star, ring, and line.

XB_CTL issues standardized configuration requests, which are translated by the RD into local versions, e.g., specific to the operating system and other node properties.

3.4. System for Application Deployment

The X-Bone includes a system for scripted application deployment (Figure 6) [24][25]. A single script is submitted during a request to deploy an overlay, with parameters instantiated on a per-overlay and per-node basis. The scripts accept a set of generic commands (***config, start, stop, status***) issued by the OM. These scripts have been used to deploy a set of web (squid proxy) caches, a set of FreeBSD *jails*, and to configure kernel modules for fault-tolerant layered overlays (DynaBone) [11][19]. The script deployment system is designed to deploy application-layer code, in the context of the addresses and DNS names of a particular overlay, on each node of an overlay in a coordinated fashion, as part of the automatic deployment system.

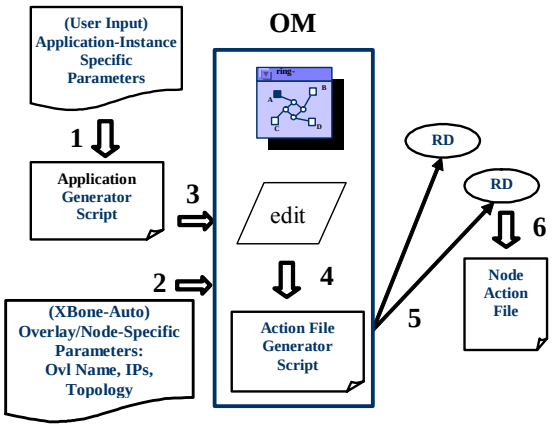

Figure 6 Application deployment system

3.5. Safety & Security

The X-Bone design and implementation focus on security. The RDs need to run as *root* to configure interfaces, setup tunnels, and install IPsec keys; as such, they present a substantial potential security challenge. Further, distributed overlay deployment relies on selecting a number of nodes based on their capabilities; knowing these capabilities can also present a security hole. Finally, application deployment needs to be carefully controlled.

The X-Bone uses an X.509 certificate hierarchy together with TCP/SSL to authenticate and encrypt all communication [10]. Each RD has its own access control list (ACL), which provides limits for each resource based on user's names (via patterns). Such resources include number of overlays, number of tunnels, queue limits, bandwidth limits, etc. Further, there are global resource limits for each node, as well as hard-coded constraints, which limit overlays to using RFC1918 address space, e.g.

The X-Bone sends S/MIME authenticated UDP invitations, and each node decides whether it has the resources indicated, as well as whether the user named in the request should be allocated those resources. Nodes respond to invitations only when they want to participate in the indicated overlay; otherwise, they can remain silent and anonymous. Some of this model needs to be relaxed in the GX-Bone, because it is not globally feasible to send invitations everywhere.

All commands issued by the OM are authenticated based on the signed key of the OM, and recorded in the RD. Each RD has its own rollback and recovery mechanism, which allows RDs to re-install state on reboot, but also dismantles overlay configuration unless a heartbeat is received from the appropriate OM that created the corresponding overlay.

Application deployment is controlled by limiting which functions are run as *root*. Although the RD uses *root* privileges for most configuration operations, application scripts are run as *nobody*, or as some preconfigured username. It is further possible to limit which scripts can be run on which nodes in the ACL, as well as to limit scripts to a predefined, preconfigured, signed set of defaults. Further compartmentalization of access is provided by NetFS [23].

4. A Global X-Bone Testbed

As noted before, the X-Bone to date has been deployed as stand-alone software, to avoid the need for any centralized coordination. The X-Bone has already been deployed in a number of stand-alone testbeds. The first was a combined USC/ISI-UCL (Univ. College London) testbed in 2000, which was used for Active Nets demos involving nodes in Marina del Rey, CA and London, U.K. Other X-Bone systems were deployed in Ottowa, Canada, Univ. Catalonia, Spain, and Univ. Kentucky. Note that we do not have a list of all sites where the X-Bone has been deployed because the system does not require any central coordination. Although this allows testbeds to be autonomous and not rely on ongoing USC/ISI support, it also fails to leverage shared resources when such sharing is desired.

As a result of our experience with independent deployments of the X-Bone, we are now preparing a global X-Bone deployment based on modified software. The new GX-Bone release includes an option, defaulted to "off", to join the global X-Bone testbed. Joining this infrastructure involves several steps, each discussed in detail below:

- advertising a node in the GX-Bone registry
- incorporating a filtered copy of the GX-Bone ACL
- incorporating a filtered copy of the GX-Bone certificate authority list

The basic purpose of these three components is, respectively, to assist with global resource discovery, to enable global use of shared resources by a known set of users, and to support distributed authentication at low cost.

There are other capabilities, developed in conjunction with the X-Bone to support overlay research, which are expected to be part of the GX-Bone. These include NetFS, a system for partitioning *root* permission requirements, and DataRouter, a string-rewriting routing system which supports application forwarding at the network layer.

4.1. The GX-Bone Registry

In order to find nodes that may be anywhere on the globe, an OM needs to direct its invitations more intelligently than by multicast advertisement. Although multicast can be used, it is inefficient when most of the nodes lack the capabilities or locations desired. The simplest alternative is to have individual nodes register with a central database, indicating their willingness to participate in the GX-Bone. This registration may include any of a node's properties, but should include enough information to limit the number of global invitations the node would receive.

Once registered, there are a number of alternatives that can be employed to assist OMs in locating desired shareable resources. The simplest is to have the OM download the database periodically, or have it pushed when sufficient changes accumulate, and have each OM scan its local copy when searching for resources. Note

that it is not necessary that the database be up-to-date, because OMs still send invitations to RDs, which are need to confirm their willingness to participate anyway. At worst, errors in the database will result in wasted invitations sent or available resources not found.

Distributing a copy of the registry has other implications, however. One of the reasons the X-Bone used an "invite-reply" protocol is to keep the resources of the local nodes private. Nodes respond only when they have the desired resources which can be allocated for the indicated user. Nodes that want to remain truly private need never make their presence known except to a limited set of parties. The central registry violates this principle, because it may contain sensitive information, such as a node's OS, patches, etc.

There are two ways to combat this privacy issue. First, information in the registry need not be complete; the less specific a node's description, the more likely it will receive invitations from other GX-Bone OMs. Those invitations are not likely to be a source of denial-of-service themselves, and so can simply be ignored if desired. Second, the registry need not be downloaded to the OMs; instead, invitations can be forwarded to the registry as part of a 'scan' process. The registry can respond with a list of potential invitees based not only on the invitation parameters (OS=Linux, Testbed=CAIRN, etc.), but also based on the context of the issuing OM. This variant allows individual nodes to place some of their configuration information in the trust of the central registry.

As with any resource location system, the GX-Bone registry has issues with scalability and performance. Like any database, it can be implemented in a hierarchical fashion with delegation along any parameter desired, e.g., by geographic region, by DNS suffix, by IP prefix, by propagation latency, or by capability (e.g., IPsec, IPv6, etc.). For the purposes of a global network testbed, a single central database suffices for overlay deployment timescales (seconds) and for the numbers of nodes expected in the near term (tens of thousands).

4.2. The GX-Bone ACL

The use of a global registry enables resource discovery, but is insufficient to enable users to utilize nodes that were unaware of their need. In addition, there needs to be a global ACL, to indicate what kind of resources are to be shared to the general public or subsets thereof. The current X-Bone ACL can already express this sort of default, in the degenerate case, e.g., where name=".*"; this is sufficient to allow users who are not otherwise listed to have resource permissions indicated.

In a global testbed, however, a single, global default is not always the best. It is safer for users to advertise their general needs, e.g., 5 virtual interfaces per node, 2 overlays per node, etc., in advance. These general resource requests are listed in the GX-Bone ACL. Individual nodes can import any subset of entries in the GX-Bone ACL, e.g., via filters (import any where interfaces<5 and where name ends in ".edu"). Users managing nodes can review these requests, and enable or deny individual entries in their own ACL as desired.

4.3. The GX-Bone CA list

The X-Bone relies heavily on the X.509 certificate system, which presumes that identity is established based on certificate authorities (CAs) known a-priori. This is similar to web browsers, which have the CA certs of a known subset of commercial CAs loaded at compile-time. Browsers allow override of this list of CAs by two mechanisms: manual loading of new CA certs, and manual approval of certificates signed by unknown CAs (typically asking, "This certificate is signed by an unknown CA; would you like to enable it a)for this connection, b) for this browser session, or c) forever?")

Because the X-Bone RDs are completely automatic, and because the OM may be interacting with a program rather than by a browser GUI, such manual confirmation is not appropriate. Instead, the list of CA certs which are used by nodes joining the GX-Bone is listed in a central database, which OMs and RDs can load (or load subsets of) as desired.

One alternative would be to preload a set of fixed CAs, as done with current web browsers, and assume that most X-Bone deployments have X.509 certificates signed by these known CAs. Unfortunately, X.509 certificates from these commercial CAs are often prohibitively expensive for research testbed use.

5. Augmented Kernel Options

The X-Bone makes no specific requirements about the kernel on which it is deployed; provided the control software (PERL code) has been ported, any operating system will suffice. There are porting requirements, e.g., to support multilayer tunnels, to support IPsec transport mode on virtual interfaces, etc.; these generic capabilities are already provided in current versions of FreeBSD and Linux. The X-Bone allows nodes to be deployed on customized kernels with modified networking stacks as well.

There are two modifications developed at USC/ISI in conjunction with the X-Bone system which may be deployed at GX-Bone nodes, to further enable advanced distributed network testbed capabilities. DataRouter supports packet routing based on string matching and rewriting, and is used to support application-layer forwarding in a network-layer overlay. NetFS supports partitioned permissions to limit *root* access within each overlay.

Neither of these kernel modifications is required for a node to join the GX-Bone, but each will be indicated in the registry. These thus represent new capabilities which we hope will further support virtual network research.

5.1. DataRouter

The X-Bone was intended to support application-specific overlays, where the topology of an overlay would be uniquely matched to a particular application's requirements. The goal was to have a single, network-layer system which could leverage existing protocols while supporting per-application topologies. This avoids having each application determine how best to deploy its own overlay.

Peer-to-peer networks have become the counterexample to this goal. They implement application-specific overlays to forward requests at the application-layer, based on application-layer information. They cannot use typical network-layer overlays, because they require application control over forwarding rules, and require that the forwarding mechanism examine application data.

DataRouter augments existing IP loose source routing (LSR) with string-based routes [21]. Individual next-hops are still indicated, as with LSR, by IP addresses, but subsequent hops are indicated by strings in the LSR field (Figure 7). When reaching a DataRouter LSR router, the string is matched against a list of patterns, and the matching rewriting rule applied. The rule also indicates the next-hop IP address, as with a conventional routing table entry.

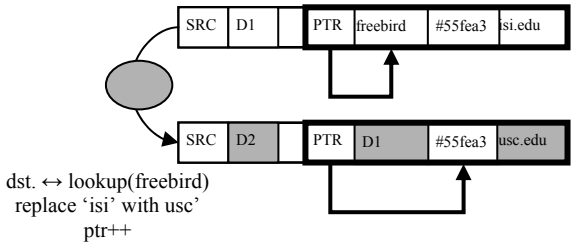

Figure 7 DataRouter LSR as string rewriting

DataRouter has already been applied to moving Chord peer-to-peer forwarding to the network layer, with a 30x increase in performance [16]. As importantly, DataRouter-Chord applications need not implement a new reliable transport protocol on top of the application layer; it can reuse TCP/IP, achieving not only higher performance, but also congestion control compatibility.

5.2. NetFS

NetFS enables applications inside an individual overlay to retain control over the configuration of a subset of interfaces, without running the application as *root*. It maps local network resources – interfaces, tunnels, routing tables, IPsec databases, etc., to a virtual file system much like */proc* maps processes to virtual files (Figure 8).

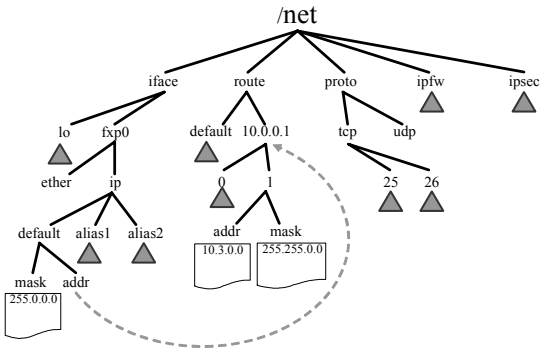

Figure 8 NetFS maps network configuration to a virtual file system

There are two advantages to such a mapping. First, *root* permissions can be partitioned, where applications deployed on one overlay can configure some network parameters but not others. Second, use of NetFS by applications simplifies portability; given multiple systems that support NetFS, a single application can manage network configuration on all systems.

6. Prior and Related Work

The X-Bone was developed to automate the effort of tunnel-based testbeds of the 1990s, notably the M-Bone and 6-Bone (the "X" in X-Bone is intended to denote a variable). It provides a more detailed architectural virtualization of the Internet, inspired by VPNs. It is designed to provide high-performance, shared network resources for application-specific use, rather than requiring recapitulation of effort as with peer networks. The GX-Bone variant is designed to provide a persistent network research infrastructure in the spirit of DartNet and CAIRN, but using primarily overlays rather than dedicated links. GX-Bone is a network version of distributed, global application testbeds such as the Grid and PlanetLab.

The X-Bone itself is inspired by the way in which testbeds evolved in the early 1990s, from dedicated routers and links in DartNet and CAIRN to tunnel-based overlays in the M-Bone, 6-Bone, and A-Bone [1][4][5][9]. Similar systems were envisioned earlier, notably MorphNet and SupraNet, in particular which described overlays at multiple layers in the protocol stack, or which explored recursion [2][6][7]. The X-Bone is the first deployment system to bring these architectural

components together in a single, proof-of-concept implementation.

The X-Bone's network-layer overlays complement link-layer testbeds, such as Emulab, and application-layer testbeds such as PlanetLab [13][26]. Both of these systems are persistent testbeds of shared components, PlanetLab being the larger-scale example. GX-Bone builds on PlanetLab's set of global resources by deploying network overlays, as well as by providing coordinated deployment of applications by the X-Bone's automated script capability.

GX-Bone is a global testbed that can deploy overlays on-demand. Other overlay testbeds have been deployed, notably the M-Bone and 6-Bone, as well as testbeds for specific research projects, notably the RON testbed [3]. RON represents a single, static set of overlay tunnels, in effect being an instance of a single GX-Bone overlay. Other overlay experiments, such as Detour, and VANs (virtual active networks), were deployed using tunnels, but did not have a persistent infrastructure beyond the individual experiment [12][15].

7. Benefits

The GX-Bone provides a new infrastructure for network research. It provides a simple, user-level interface to dynamic overlay deployment. It supports concurrent overlays with automated resource management, avoiding the need for 1960's-style pegboard reservation schedules. It supports very high performance research, and allows the reuse of existing application, transport, and network layer protocols, as well as existing applications. It also allows experiments based on the existing knowledge base of kernel-level modifications, allowing custom kernels to participate in a global, shared infrastructure.

7.1. Ease of use

The X-Bone provides a web-based, do-what-I-mean GUI for overlay deployment. Resources are used as available, where nodes support concurrent overlays where possible, and avoid them (via resource counts, e.g., num_overlays=1) where necessary. There is no need for a 1960's style pegboard reservation system, no need to reserve resources in advance. The X-Bone brings overlay deployment and use into the realm of multitasking, a kind of network multitasking operating system.

This contrasts to previous network testbeds, such as DartNet and CAIRN, where resources were reserved by calendar system [5]. It also contrasts to PlanetLab, where 'slices' – sets of vservers deployed on a set of nodes – are deployed a-priori and left in place [13]. Further, due to how vservers operate, it is not possible to configure the network interface of a vserver from within that vserver, and PlanetLab configures the vservers but does not create tunnels between them. The X-Bone has already demonstrated 'slice' deployment using our application deployment system, where a simple script –written in a few hours – was used to deploy a set of vservers connected by a set of IPsec-encrypted tunnels.

Further, the X-Bone, and GX-Bone as a result, can deploy overlays emulating different network characteristics, using *dummynet* [14]. A deployed overlay can be configured with additional link latency, link bandwidth limits, link queue limits, or emulated link losses. *Dummynet* is only one example; any system that can be configured from the command line can configured by the X-Bone's application deployment system or incorporated into the core system's configuration instructions (X-Bone runs as source), as desired.

7.2. Performance

The X-Bone has achieved performance of gigabits/second and hundreds of thousands of packets/sec on commodity PC platforms [19]. It forwards packets in the existing kernel code, at existing kernel rates. The X-Bone's RD code configures the host or router, but otherwise packets are processed by the conventional, optimized code. The X-Bone software has been demonstrated to support hundreds of concurrent overlays on a single, conventional PC, and recursive overlays up to 16 levels deep.

X-Bone packets are forwarded at IP forwarding rates at intermediate nodes. At overlay routers, there are additional encapsulation and decapsulation steps, as well as additional forwarding steps. With modern processors, the encapsulation and decapsulation steps are negligible, and the forwarding rate is limited only by the additional pass through the kernel that two-layer encapsulation requires. As a result, a single-level overlay (not recursive) forwards at roughly half the conventional IP forwarding rate.

This contrasts to the performance of peer-to-peer systems, which typically 'forward' messages at 1/30 the rate of IP forwarding (5,000 pkts/sec, vs. 150,000 pkts/sec for an overlay, vs. 300,000 pkts/sec. base IP forwarding on a 2.4Ghz Xeon PC).

7.3. Reuse of net and transport protocols

The X-Bone enables network level research without the need for applications to reinvent transport and other protocols. DataRouter extends conventional IP forwarding with string matching and rewriting loose source routing, allowing an overlay to support peer-to-peer forwarding with network layer performance. Messages need not be terminated at each hop, e.g., using hop-by-hop (HBH) RPCs, as is done in Chord; such HBH behavior at the

application layer requires additional application mechanisms to provide transport protocol properties, such as reliable, in-order delivery [16].

Further, network layer overlays can reuse existing network and transport protocol mechanisms, such as dynamic routing dynamic routing, congestion control, and windowing. This avoids the need for applications to reinvent – or rediscover – solutions to these well-known problems.

By using network solutions to these issues, the X-Bone allows existing applications to operate over deployed overlays. Existing implementations of FTP, ping, traceroute, web servers, mail servers, etc. all operate over X-Bone overlays. This leverages the installed base, and avoids additional effort and errors of recapitulation.

7.4. Allows custom kernels to participate

The X-Bone's design relies on conventional IP packet forwarding, with common extensions for IP encapsulation, where the entire X-Bone system is deployed at the application layer. RDs and OMs configure network interfaces, tunnels, and routing table entries; packets continue to be forwarded by existing kernel mechanisms. DataRouter is a good example of how conventional kernel-based network protocol experiments can be incorporated into an X-Bone overlay; the X-Bone system is deployed on a DataRouter kernel [21]. Similar modifications can be deployed anywhere in the GX-Bone, flagged by labels which can be used by experimenters to find appropriate nodes anywhere in the world (e.g., find "DataRouter" nodes).

8. Future Work

The GX-Bone, like the X-Bone on which it is based, represents an ongoing effort at USC/ISI to develop and explore the Virtual Internet Architecture. In addition to the ongoing development of the GX-Bone system, including its registries and discovery system, we are working to enable more complex, layered deployment of large-scale overlays. Two specific avenues of future work involve support for partitioned resources and support for inter-overlay gateways.

8.1. Partitioned resources

The X-Bone currently deploys network-layer overlays, assuming individual nodes can participate in multiple overlays without mutual interference at the network layer. NetFS is designed to avoid the interference of *root* permissions among these overlays, but there are other ways in which overlays compete at shared resources. In particular, it can be challenging to limit computational resources, disk volumes, etc. across overlay instances, as current hosts and routers do not expect to virtualize their resources.

Jail partitions some of these resources, notably disk volume access, and provides localized *root* permissions for some purposes. Unfortunately, *jail* and its variants (*vservers*, *VMware*, etc.) do not fully support our Virtual Internet Architecture's notion of network reentrancy, where wildcards like *INADDR_ANY* map only within each partition. These systems also do not support sets of associated interfaces within a jail, partitioned from other sets in other jails.

Clonable stacks are a step in the right direction [28]. Although intended to support alternate network stacks, e.g., TCP-Reno and TCP-Tahoe, it can be used to support the associations required for network reentrancy. In particular, interfaces within a single overlay at virtual router instance are associated with a weak-host model, and interfaces between such interfaces are associated by the strong-host model.

Clonable stacks also offer new opportunities to limit the amount of CPU load allocated to each stack, thus limiting the computational resource interference of multiple virtual hosts or virtual routers instantiated on a single node. There remains substantial work to explore and validate this capability, and to port it to other systems.

8.2. Inter-overlay Gateways

The current X-Bone deployment system assumes that individual overlays are strongly partitioned, avoiding inter-overlay interaction. This prohibits inter-overlay gateways, such as would be used to splice together disparate services running on individual overlays. It is not yet clear what best architectural extension would support this capability, because we prefer the Internet model (common communication framework) to that of concatenating disparate network layers. One possible option is to explore recursion for this capability, where each recursive component represents a unique network architecture, and the overarching "outerlay" (to borrow terminology from DynaBone) supports the gatewaying capability [19].

9. Acknowledgments

The authors thank the numerous contributors to the code and architecture of the X-Bone project, both within projects of USC/ISI (X-Bone, DynaBone, NetFS, DataRouter) as well as in collaboration. These include Steve Hotz, Amy Hughes, Josh Train, and Nimish Kasat and others at USC/ISI, Peter Kirstein, Panos Gevros, Manash Lad, Piers O'Hanlon, and others at UCL in the U.K., and Gregorio Martinez and Manuel Gil Perez and others at the Univ. Murcia in Spain.

This work was partly supported by the NSF STI-XTEND (ANI-0230789) and NETFS (ANI-0129689) projects. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

10. References

[1] 6-Bone URL – www.6bone.net

[2] Aiken, R., et. al., "Architecture of the Multi-Modal Organizational Research and Production Heterogeneous Network (MORPHnet)," ANL-97/1, Argonne National Lab, IL., Jan. 1997.

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

[4] A-Bone URL – www.isi.edu/abone

[5] CAIRN web pages – www.cairn.net

[6] Campbell, A., et al., "Spawning Networks," IEEE Network, July/Aug. 1999, pp. 16-29.

[7] Delgrossi, L., Ferrari, D., "A Virtual Network Service for Integrated-Services Internetworks," 7th Int'l Workshop on Netw. and OS Support for Digital Audio and Video, May 1997.

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

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

[10] Hickman, Kipp, "The SSL Protocol," Netscape Communications Corp., Feb. 1995.

[11] Kamp, P., Watson, R., "Jails: Confining the omnipotent root," Proc. 2nd International System Administration and Networking Conference (SANE), May 2000.

[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] PlanetLab – www.planetlab.org

[14] Rizzo, L., "Dummynet: A Simple Approach to the Evaluation of Network Protocols," ACM Computer Communications Review (CCR), v27 n1, Jan. 1997, pp. 31-41.

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

[16] Stoica, I., Morris, R., et. al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. Sigcomm 2001, Aug. 2001, pp. 149-160.

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

[18] Touch, J., Eggert, L., Wang, Y., "Use of IPsec Transport Mode for Dynamic Routing," RFC-3884, Sep. 2004.

[19] Touch, J., Finn, G., Wang, Y., Eggert, L., "DynaBone: Dynamic Defense Using Multi-layer Internet Overlays," Proc. 3rd DARPA Information Survivability Conf. and Exposition (DISCEX-III), April 22-24, 2003, Vol. 2, pp. 271-276.

[20] Touch, J., Hotz, S., "The X-Bone," in Proc. Third Global Internet Mini-Conference, Proc. Globecom '98, Sydney, Australia Nov. 1998.

[21] Touch, J., Pingali, V., "DataRouter: A Network-Layer Service for Application-Layer Forwarding," Proc. Int'l Wkshp. on Active Networks (IWAN), Springer-Verlag, Dec. 2003.

[22] Touch, J., Wang, Y., Eggert, L., Finn, G., "Virtual Internet Architecture," Future Developments of Network Architectures (FDNA) at Sigcomm, August 2003. (ISI-TR-2003-570).

[23] Train, J., Touch, J., Eggert, L., Wang, Y., "NetFS: Networking through the File System," ISI Technical Report ISI-TR-2003-579.

[24] Villanueva, O.A., Touch, J., "Web Service Deployment and Management Using the X-Bone," Spanish Symposium on Distributed Computing, SEID2000, Sept. 25-27, 2000.

[25] Wang, Y., Touch, J., "Application Deployment in Virtual Networks Using the X-Bone," Proc. DANCE: DARPA Active Networks Conference and Exposition, May 2002, pp. 484-493.

[26] White, B., Lepreau, et al., "An Integrated Experimental Environment for Distributed Systems and Networks," Proc. Fifth Symp. on OS Des. and Impl. (OSDI), 2002, pp. 255-270.

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

[28] Zec, M., "Implementing a Clonable Network Stack in the FreeBSD Kernel," Proc. USENIX 2003/FREENIX, pp. 137-150.