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

The X-Bone is a system for deploying and managing Internet overlays [1][2]. It coordinates the configuration and management of virtual networks, enabling shared use of network resources (Figure 1).

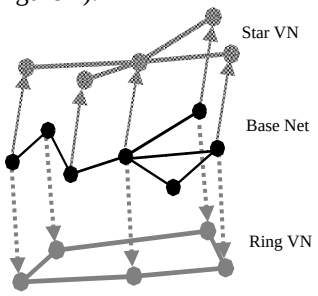

Figure 1 Multiple virtual Internets

The Global X-Bone (GX-Bone) extends the X-Bone implementation from a stand-alone software system for local experiments to a global infrastructure for wide-scale network research. The GX-Bone augments the X-Bone software, enhancing its coordination mechanisms to support deployment of local overlays to world-wide, shared infrastructure. GX-Bone also complements the network virtualization-related features of X-Bone overlays with fine-grained access control to network configuration through NetFS [1] and application-directed forwarding through support for DataRouter [3]. GX-Bone can also be installed on user-modified kernels, uniquely supporting both conventional kernel-level protocol development and coordinated global infrastructure sharing.

A global Internet overlay testbed based on GX-Bone is currently being deployed to support the distributed, shared use of resources for network research.

1. The X-Bone

In this section, we discuss the X-Bone system that is at the core of GX-Bone. The X-Bone is a system for the dynamic deployment and management of Internet overlay networks [1][2]. 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)[9][10][11]. 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.

X-Bone creates IP tunnel-based Internet overlays consistent with a general architecture for network virtualization of the Internet [4]. 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.

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. The X-Bone system allows different applications on the same end host or router to be associated with different overlay networks through its application deployment mechanism. The combination of virtualization and application deployment capability enables controlled experimentation with advanced networking applications within a global, virtual network.

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. X-Bone uses existing implementations of IP services such as multi-

layer tunnels, virtual interfaces, IPsec, dynamic routing, name service, and other infrastructure.

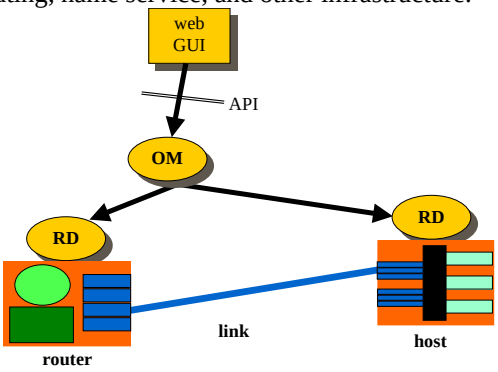

Figure 2 X-Bone Software Design

The X-Bone distributed system is composed of *Resource Daemons* (RDs) and *Overlay Managers* (OMs), with a graphical user interface (GUI) and an XML-based API. These components are shown in Figure 2. The functions of the RD and OM have been incorporated into a single daemon, but operationally they can be discussed as distinct units. All communication is secured using SSL and S/MIME. Further ACLs are used to limit the overlay creation and use.

OMs deploy overlays; a user creates an overlay by sending a request to an OM, either via a web-based GUI (Figure 2) 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.

2. 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. 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.

The new GX-Bone release of X-Bone software includes an option, defaulted to “off”, for nodes in a testbed 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 all at low cost.

The registry typically includes node’s properties or preferences that provide enough information to limit the number of global invitations the node would receive. During overlay deployment, the registry’s entries are searched for appropriate hosts. It is not necessary that the registry be accurate because each RD confirms access and resource control, as in the default system. The registry primarily helps to improve efficiency of overlay deployment through refinement of the set of nodes to be contacted.

GX-Bone ACL is a global X-Bone ACL that is used to indicate what kind of resources are to be shared to the general public or subsets thereof. Sites/nodes may selectively adopt from the GX-ACL. The current X-Bone ACL structure can already express a sort of catch-all default, in the degenerate case, e.g., where name=“.*”; this is sufficient to allow users who are not otherwise listed to have resource permissions attached to that entry. In a global testbed, however, a single, global default is not always appropriate. Individual nodes 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”).

The X-Bone relies on the X.509 certificate system, which presumes that identity is established based on certificate authorities (CAs) known a-priori. However, X-Bone does not specify the CA to be used. Different deployments may use different CAs. To allow automatic authentication in the presence of multiple CAs, GX-Bone maintains a central database of known CAs that is automatically distributed to all sites. This allows authenticated communication between any two nodes from sites that are part of the GX-Bone without manual intervention.

The GX-Bone support for global registry, ACL and CA database, and automatic distribution provide the administrator of a X-

Bone testbed global visibility for the testbed nodes without compromising on control. Further, other capabilities, developed in conjunction with the X-Bone to support overlay research, are expected to be part of the GX-Bone. These include NetFS, a system for partitioning *root* permission at fine granularity for network configuration, and DataRouter, a string-rewriting, late-binding, generalized loose source route-based system which supports application forwarding at the network layer. More information on these can be found elsewhere [1][3].

Both DataRouter and NetFS require modification of kernel. However, neither of these kernel modifications is required for a node to join the GX-Bone. Presence of such kernel extensions is indicated through the registry as node properties. Accordingly, the overlays and applications that run within can be customized to use the extensions. Similarly other new capabilities [7][8] can advertised and used for network research, especially over wide area networks.

3. Global X-Bone Testbed Benefits

The GX-Bone provides a new infrastructure for network research. It provides a simple, user-level, do-what-I-mean interface to dynamic overlay deployment with automated global resource management.

The system allows for safe, controlled and wide-area network experiments that can leverage existing knowledge and code base of kernel-level modifications. Since GX-Bone treats local and global nodes almost identically, experiment design is the same when done locally or globally. We expect that in most cases, the only difference will be a change in node selection criteria on the GUI.

GX-Bone supports very high performance research, and allows the reuse of existing application, transport, and network layer protocols, as well as existing applications. The computational and disk overhead of X-Bone is minimal and predicable[1]. Since the networking interface remains unchanged, most existing software can be used unmodified. Further DataRouter [3] can help simplify and improve the performance of application-level networks.

4. Status and Demo

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. Current version of X-Bone includes support for IPv6, dynamic routing (RIP/RIPng), Dummynet, Cisco routers, and DNSSEC. The API is XML-based and the distribution comes with a web interface.

Prototypes for NetFS and DataRouter have been completed though not released officially. Current implementation of Global X-Bone database uses LDAP. The implementation has been completed and will be released shortly.

The demonstration at INFOCOM 2005 will include the X-Bone functionality and GX-Bone components.

References

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

[2] Touch, J., Hotz, S., “The X-Bone,” in Proc. Third Global Internet Mini-Conference, Proc. Globecom ’98, Sydney, Australia Nov. 1998.

[3] Touch, J., Pingali, V., “DataRouter: A Network-Layer Service for Application-Layer Forwarding,” IWAN, Springer-Verlag, Dec. 2003.

[4] Touch, J., Wang, Y., Eggert, L., Finn, G., “Virtual Internet Architecture,” Workshop on FDNA at SIGCOMM, August 2003. (ISI-TR-2003-570).

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

[6] Wang, Y., Touch, J., “Application Deployment in Virtual Networks Using the X-Bone,” Proc. DANCE, May 2002, pp. 484-493.

[7] Dina Katabi, Mark Handley, and Charles Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments." SIGCOMM, August 2002.

[8] Kaur, H. T. et al, “BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet," Proceedings of ACM SIGCOMM Workshop on FDNA, Volume 33, Issue 4, Pages 277-288, Karlsruhe, Germany, August 2003.

[9] 6-Bone URL – www.6bone.net

[10] A-Bone URL – www.isi.edu/abone

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