Web Service Deployment and Management Using the X-bone

Oscar Ardaiz Villanueva, Polytechnic University of Catalonia
Joe Touch, University Southern California / Information Science Institute
oardaiz@ac.upc.es, touch@isi.edu

Abstract

In this paper we present a system for automatic deployment and management of web service. The problem we try to solve is how to introduce new distributed services in a network. Today deployment of a distributed service is a costly and time consuming process which requires obtaining network topology and a well thought placement and organization of servers, besides coordinated installation, configuration and execution of servers. Previous to this work, this problem has been addressed tangentially in a couple of works, which were mainly concerned about proposing a killer app of active technologies.
Extending the X-bone [1], a system for automatic overlay deployment and management, with mechanisms for code distribution, remote application installation, configuration and execution, makes it capable of deploying and managing distributed services. Currently it deploys web caching service and distributed web service.
Distributed service deployment and overlay deployment with the X-bone has one fundamental property: there is no need to modify networking code or application code therefore deployment can take place over current Internet infrastructure. This makes it useful for deployment of new services on today's production networks. Beside the system is also concerned with management of the service after it has been created, which is also essential in a production network.
Currently we have demonstrated our prototype in a small experimental network. We are looking for some large-scale real scenario where to demonstrate the validity of the system.

Keywords: web caching, distributed web servers, distributed service deployment, distributed service management

Oscar Ardaiz Villanueva is supported by the Spanish Ministry of Education through FPI Grant # AP97-33426821.
This work is partly supported by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-98-1-0200. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

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.

1 Introduction

There has been much attention to the problem of introduction of new services into the network for two main reasons as commented in [2]. In first place, there is a users demand for new network services which are already well tested as multicast, IPv6, web caching, etc., but are not widely available because their introduction requires costly and time consuming manual deployment by network operators. In second place there is a technological push from recent advancements in active technologies: active networks, which permit to inject securely programs into the network thereby allowing for faster service innovation. This problem has also being address from a telecommunication approach by the Open signaling community which argue that by "opening up" network nodes, new services can be realized [3].

Introduction of new application layer services is not less a lengthy process than introduction of network layer services. Many distributed services are uncommon in ISP networks such as the NTP time service, LDAP directory service because it takes too long for administrator to deploy; other have been deployed in a tedious process by network administrator such as web caching services, and others will never be available for end application since nobody will care to deploy them, Fe. maybe event services.
The work presented in this paper concentrates on the deployment of web services: web caching service and distributed web service. Caching and replication have been known for long to be the solution to applications performance degradation. As such they were proposed extensively to be applied to the Web (and to FTP) as soon as they become fairly popular [4][5][6][7][8]. However neither solution has been widely adopted.
The main reason has been the fact that both solutions depend very much on network characteristics, require a well thought placement and organization of servers and a coordinated installation and execution, thereby it takes a long time to set it up. In fact many misconfigured caching systems and distributed webs worsens user perceived web performance. Consequently caching and distributed solution gained the fame of not being useful.
Our goal is to create a system for automatic deployment and management of web services, which will increase the rate of installation of such distributed service, and ultimately improve performance of the Web and relieve network congestion.
We have extended the X-bone [1], a system for automatic network overlay deployment currently being developed at USC/ISI, to deploy web caching services and distributed web services.

2 Problem Statement

Our goal is to develop a system that automates deployment and management of caching services and distributed web services relieving network administrators of costly and time consuming deployment tasks. With this goal in mind, we need to analyze first how network administrators manually deploy them, to characterize which steps need to be automated.
Today deploying a caches hierarchy or mesh requires:

- Discovery of network topology and hosts,
- lay out a caching topology as in fig. 1 (= select servers placement and organization to reduce network traffic and decrease clients access time),
- coordinated installation, configuration and execution of proxy caching servers,
- configuration browsers or redirector service,
- monitoring service performance and reconfigure if necessary.

And deploying a distributed web requires:

- discovery of network topology and hosts
- lay out a content delivery topology as in fig. 1. (= select servers placement and organization to decrease clients access time, balance servers load and reduce network traffic),
- coordinated installation, configuration and start-up of distributed web servers,
- configuration redirector service,
- monitoring of service performance and reconfigure if necessary.

A deployment system for both web services will automate all these tasks, it will be responsible for their management, and it will monitor that it works properly. If things go wrong or the environment changes it will reconfigure or shut down and redeploy the service.

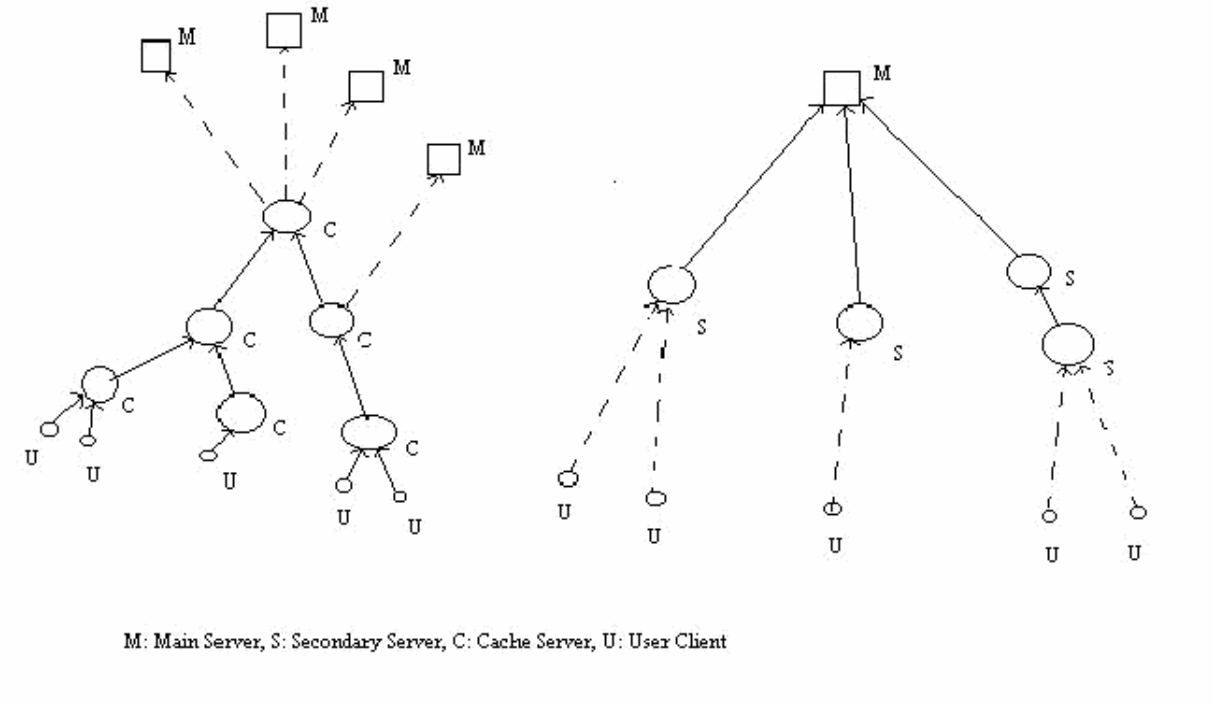

Fig. 1: Web caching service topology, distributed web service content delivery topology.

3 Related Work

Architectures for deployment of distributed services have been proposed by Amir [9]: the Active Service Framework v.1, and by Govindan et al. [10], which propose an Active Distributed Service Framework and discuss which are their issues. Both proposals try to benefit from active technologies, which allow safe code distribution, installation and execution, to create a framework where distributed services can be easily introduce into a network. However, unlike our solutions, it requires application recompilation and new host environments to be deployed.
Amir develops a MeGa service within this framework which allows client multimedia applications to deploy one single multimedia proxy gateways on the nearby network, which translate from one multimedia stream format to another. It uses three mechanisms for deployment: announce listen communications, soft state at hosts and multicast damping to solve multicast implosion, which yield a

robust implementation with efficient resource sharing. The main drawback of this system is that it only allows for one single client deploying one single proxy. It has not demonstrated how a service creator would deploy coordinately a large scale service with several server.

Another project that automatically deployed web services is WEB-OS in its Rent-A-Server application [11]. WEBOS is a project aimed at creating a set of basic operating system services to support for geographically distributed, highly available, incrementally scalable, and dynamically reconfigurable applications. Rent-A-Server is a WEB-OS application to dynamically deploy web servers based on users demand. Its main drawback is that it is built on a new OS which requires adding a layer on top of existing OSes to intercept new system calls.

Management of distributed web services is the most essential function of content distribution networks companies as Akamai [12] and as such we presume that they will have implemented a good management system, however proprietary. Also projects like RaDar [13] & WebWave [14] have proposed architectures for a global information hosting service based on dynamic object replication and migration. These architectures support several mechanisms for distributed web services management as distributed load monitoring, dynamic content delivery topology reconfiguration, etc.

Organization of a group of cooperating cache servers has been researched in LSAM project at ISI [15], which uses multicast to discover and distribute cache configuration, plus auto configuration agents which automatically create a hierarchy of caches, or the Adaptive Web Cache project at UCLA [16] which aims at creating a self organizing virtual topology of multicasting group caches with a totally distributed request routing protocol.
State of art in network management systems for cooperating caching servers currently spins around Squid SNMP support for managing cooperative caches [17]. However we do not know of any application developed to manage a caching system using that Squid SNMP data.

Finally network storage infrastructures such as Internet 2 Distributed Storage Infrastructure [18] and the Intelligent Storage Market Infrastructure [19] are suppose to be the substrate where caching and distribution services would be provided on demand. However little work has been done showing how caching and distributed services would be deployed on either of these infrastructure and how they will be managed.

4 Discussion

Creation of a service (or an overlay) which is composed of several cooperating processes running on different network nodes presents several important issues. Which are the fundamental components of a deployment service? , how is nodes environment where services are deployed?, how are resources provisioned and managed to build with them a service or an overlay?, which are the mechanisms that will provide the deployment service?, and will it have these desirable properties: efficiency (and scalability), security, fault tolerance, ease of use, manageability of created services, etc. ?.

4.1 Deployment service main components

4.1.1 Group Creation

The main abstraction of the deployment service. Traditionally in group communication systems ,as Horus [20], a group was created as soon as one individual requested to join the group, subsequently other individuals joined the group. However in a deployment service a group creator explicitly requests a number of individuals to be the group at its creation. Afterwards this group creator can easily assume the role of group manager. One of the main concerns is operation atomicity since individuals might not be willing to join the group, or not being ready. Group creation operation can have different variants depending on how you define the group: by group cardinality, by group aggregated properties, etc.
In the X-bone an overlay manager is responsible for making multicast request, selecting group members among those nodes that respond and controlling that they all join the group atomically.

4.1.2 Group Organization

Group members are organized to provide its service in an efficient manner. Group organizations are represented by topologies. Most common topologies (organizations) are hierarchies and meshes, others being common are rings, trees, etc.
A deployment service needs to layout the group topology. It can be selected by the user or it can be derived from its requirements by the requirements mapping function. Afterwards it has to configure each node according to this topology. Nodes can be sent their position in their topology or they can look it up in a topology server.

4.1.3 Service API

The deployment service is invoked by the service creator through an API. This API main function shall be Create_() in which service or overlay specification will be passed; it will return the reference name by which this service can be accessible or the deployment service could publish the service itself in the DNS.

4.1.4 Requirement mapping

Service requirements corresponding to services that need lower-layer services are translated into lower-layer service requirements, fe. application service requirements (transaction download time, ...) are translated into networking service requirements (e2e delay, e2e bandwidth,..). In the X-bone requirements are currently expressed in its simplest form: number of hosts and routers, link BW, etc. and are not mapped. Chang describes in [18] several equations on how to map requirements for a distributed service such as minimum access latency, maximum storage cost or minimum number of storage nodes to a set of storage servers connected by a networking service for a distributed storage infrastructure, but does not show any prototype.

4.1.6 Deployment at multiple layers

Different layer services have different deployment requirements. To create a generic API we will have to investigate which are the differences among them, f.e. an app layer service requires persistent storage resources, it has its own topology and routing (servers organization) which needs not to be the network topology and routing, its performance is characterized with different parameters from network layer

performance and from link layer performance (transaction latency, transactions rate vs. e2e delay, e2e BW, loss rate vs. hop delay, hop BW, hop loss rate), and its service depends upon the networking service which calls for coordinated multilayer service deployment.

4.2 Nodes environment

Services or overlays are deployed on a set of existing nodes which have to provide an environment so that processes which comprise the service or overlay can be remotely executed securely and effectively. This has been the primary concern of active networks research which using latest advances in secure programming environments have developed network nodes which allow up to in-packet code distribution and per-packet code execution networks. However this flexibility comes with a performance penalty very difficult to ellude. As well management of this active nodes is being point out as being very difficult.
The X-bone does not use an active approach, rather it uses standard Unix machines as node environment, which have proven to be very flexible and secure enough, for a long time in the Internet community. It will have as a positive side effect that we will not have to care about learning and managing a new environment.

4.3 Resource provisioning and management

Sufficient resources have to be provisioned so that a service or overlay is deployed according to the specifications requested by its creator. It implies several functions that have to carried on on every network node:

- Resources mapping
  It is the translation from high level service specifications to low-level resource requirements. In QoS network architecture [21] we can find many examples of which are the difficulties of mapping those high level requirements to CPU, bus, memory, I/O resources.
- Resource discovery
  Enough resources to meet those requirements have to be discovered. Resource and topology discovery is an active area of research since it is being more common for applications to request special network characteristics. Multicast hop limited queries is an elegant solutions which the X-bone uses, however limited availability of multicast ask for different approaches. Unicast solutions are valid, however they have limited scalability, [22] proposes a unicast solution which is scalable, however it implements it with active networks technology, which can be as useless as the multicast approach due to limited deployment.
- Resource reservation
  Some of those resource need to be reserved either for long term comsuption, or to allow for atomic deployment in systems with several concurrent deployment entities. Currently resource reservation in only handled within RSVP integrated services architecture. Currently the X-bone does not enforce any resource reservation, but we will have to evaluate which strategics are appropriate.
- Resource management
  Resources in every node will be shared among different services and overlays from different owners, which calls for effective resource management policies and mechanisms on every node. In the X-bone a resource daemon in every node keeps track of resource usage, maps resource to request, announces resource availability, and enforces predetermined policies. Currently only bin policies are implemented. This is the local agent whereby deployment takes place on every node.

4.4 Deployment mechanisms

Once resources are allocated for a service or overlay, service processes must be started on every node. It can require that service code be distributed to every node. Processes also need to be configured before execution and they must export their management interface so that they can be monitor and configured subsequently. There are several ways these mechanisms can be implemented:

- Code distribution
  Deployment manager uploads server code to hosts or hosts could download application code. If there were several host to deploy the application, a reliable multicast protocol could be used to distribute it.
  Next this code has to be installed in the system, before the application can be executed, which can involve uncompressing it, compiling it, storing it locally and registering it, operations which the local resource daemon will perform.
- Configuration and execution
  Servers have to be configured before execution. The deployment manager could configure the application or the application could configure itself. Beside application specific parameters, there are some generic parameters like topology and routing configuration which the deployer manager could assign better because it has a global knowledge of network topology.
  Deployment manager commands all hosts to start the application server. The service is up as soon as all servers are up. This is the most dangerous operation to be performed for service deployment. Active technologies solve this security hole with a hosting environment that only allows programs to execute safe operation. This way programs can auto load and execute themselves without requiring permission from anybody else. The X-bone solves this issue by allowing this risky operation only to certified agents.
- Management interface registration
  So that a service node process can be managed after it is running it must export its management interface so that any management entity can talk to it for monitoring and configuration operations.

4.5 Deployment service properties

The deployment service has to have several properties:

- Ease of use
  So that it is widely used the deployment system needs a simple, ease-to-use GUI. And it is already a must that it is fully decentralized, being accessible from anywhere. Web based front end GUIs are emerging as a good alternative to access the management system from any location in the network in an intuitive well known interface. The X-bone uses the web GUI.
- Manageability of created services
  Today management of a group of cooperating processes is a challenging task. Its main cause is the fact that networking management is only provided at the individual element. A deployment service has the opportunity to make things change, since the deployment entity is the most suited process to find out how to manage this group service since it was able to create them. This idea has been already proposed at [23].
- Fault tolerance
  Any error on the deployment process should halt deployment, the system should roll back deployment stopping any process started and locking out any resource.
  On networks where there can be several deployment managers, it could be the case that it is found

a host that has enough resources to execute a server, however before the server is installed and executed another entity could have deployed its server, leaving not enough resources to the first manager. Reserving resources will avoid this case.

- Security
  A system for deployment of services has many security concerns: from deployment of services by non authorized parties to deployment of services that eat up too many resources. There are extreme solutions, like creating a secure hosting environment that allow any third party program unrestricted access to resources like in some AN environments. But the most obvious solutions is to certified every command that could incur in some danger and to certify any executable to be distributed which is the option taken in the X-bone.
- Performance
  In such a complex process there can be several task which delay deployment and make it an unscalable service. We shall find out which task can be with a first operational prototype.

5 Web Service Deployment using the X-bone

The X-bone is a system for automatic deployment of network overlays. Network overlays are virtual networks, used as experimental networks where to run new network services like multicast IP, or as corporate virtual private network, etc. Deploying a network overlay requires discovering remote network nodes, laying out overlay topology, assigning addresses, determining tunnels locations, ensuring the overlay network has enough resources and configuring tunnels and routing tables. Automatic deployment of network overlays is achieved by the X-bone with a number of mechanisms: an overlay manager that configures and manages overlays, resource daemons that manages resources usage at nodes, and a multicast control protocol for efficient resource discovery and management communication.
X-bone mechanisms for overlay deployment can be reused to deploy web services.
We have implemented a web service deployment system reusing X-bone node resource manager functions, resource provisioning functions, deployment mechanisms and service API. We have extended the X-bone to deploy web services over overlay networks. An overlay network is deployed, and the web service is deployed on it. Configuration of any distributed service is simpler since the overlay hides to the application the complexity of the underlying network.
Characteristics of the X-bone that are useful for deployment of web services are:

- an API for service deployment and management, implementing functions as service creation, service monitoring, etc,
- an atomic overlay creation operation which can be the foundations for a group creation operation,
- resource provisioning mechanisms: multicast resource discovery and resource managers for local resource mapping and allocation,
- a topology abstraction object which we can reused for configuring easily web service organization,
- security mechanisms for service deployment using 3rd party certification,
- and finally, it works with unmodified code and host systems, unlike active technologies, which makes it possible to deploy on existing Internet networks.

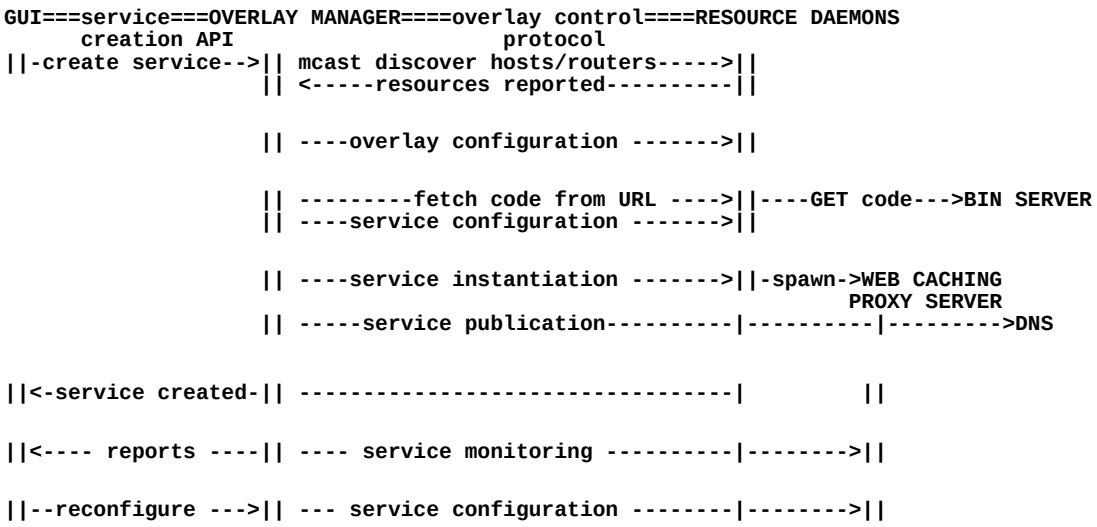

Fig. 2. Web Service Deployment Using the X-bone

5.1 Web caching service deployment

The deployment of a web caching service using the X-bone begins by discovering hosts capable of running caching servers. Discovery is being implemented using multicast hop-limited queries. Increasing TTL query packets are sent on a well-known port, expanding search distance as TTL increases. Nodes respond in a unicast packet to the overlay manager attaching information on node resources.

Overlay manager selects nodes that meet user requirements to deploy caching proxy servers. It creates an IP overlay with the selected topology setting up tunnels, configuring routes, etc. It send those nodes commands to obtain server executable binaries from a URL location. Host resource daemons fetch and install those binaries if they were not installed already. Overlay manager sends resource daemons commands on how to configure those proxy caching servers: IP address to which they have to bind, address of parent or sibling proxy caching server to forward requests to, and other application specific configuration. Finally every resource daemon spawns a caching server and responds whether everything was ok. If there is some negative response, the overlay manager rolls back deployment, else it considers deployment satisfactory and starts monitoring the service. Currently the only monitoring function implemented is a heartbeat ping sent by managers that refreshes resource daemons state.

Deployed services can be shut down by the overlay manager which commands resource daemons to stop and de-install servers.

Security is implemented using X.509 certificates. The overlay manager and each resources daemon certify each other on every request that modifies the status of hosts. Discovery requests are not secure. Server binaries are downloaded from a location indicated by the overlay manager, which should care that code staying there is secure, resources daemons do not certified those locations or the software they download yet.

Resource management is done by resource managers at every participating node who take care that a server is not installed beyond a limit on the number of servers allowed on that node.

Currently, services deployed are selected through a web GUI front-end where service type and overlay topology requirements are selected. This deployment request is sent to the overlay manager by an X-bone API interface. The overlay manager tries to map that overlay selection on available resources. Supported configurations are "ring", "mesh" and "star" topologies. If a selected overlay can be created, a

web caching servers will be deployed on it. In a ring topology a parent cache will be place on the center node, in a mesh every cache server will be sibling of the others, and in a ring topology, every cache is sibling of its neighbors. More complex topologies will be create by combination and recursion of the above.

5.2 Distributed Web service deployment

A system for the deployment of a web caching service can be easily converted into a system for the deployment of a distributed web service. Only minor changes on the configuration of servers and resource daemons will be required if it is based on surrogate proxy caching servers (a proxy caching server that responds solely to requests for the main web server), as Akamai [11] and others are doing. A distributed web service using surrogate proxy caching servers is a good option for not very dynamic web content. It relieves us from having to manage inconsistencies among replicas.
Deployment of a distributed web service involves the same steps as to deploy a web caching service plus a content delivery channel must be set up. Through the content delivery channel content is distributed to secondary web server either pushed from the main web server or pulled by secondary web servers. In addition, all server addresses have to be published as locations where that web content can be accessed. If distributed servers are surrogate proxy caches the content delivery channel is set up by configuring all higher level proxy caches to have the main web server as their parent.
The DNS server for that web domain name has to be configured with all addresses of surrogate proxy caching servers that provide distributed web service to support server selection. This DNS server will redirect request to the right server using some topology aware and load balancing algorithm.
Currently deployment of a distributed web service on a star topology configures the center proxy cache to be child of the main web servers and all other caches to be child of this cache. In other topologies all caches are children of the main web server. And the DNS server only uses a round robin server selection algorithm.

6 Conclusions & Work in Progress

We have deployed a web caching service and a distributed web service in a system comprised of a handful of hosting nodes. This set up has convinced us of the usefulness of some X-bone mechanism for web service deployment.
The X-bone fulfills some of the properties we request a deployment service: fault tolerance is achieved with a commit or rollback mechanisms which provides atomic group creation operation; security with certification is a well known solutions, which is as secure as good is our certification management system; plus it has a web GUI which makes it very easy for operators to learn how to used the service being as well accesible from anywhere.
Important properties not yet experimented are how manageable are services once created and which is the efficiency and scalability of this system.
Useful mechanisms of the X-bone for web service deployment are:

- the service creation API for overlay deployment can be used with little modifications for web service deployment,
- usage of standard operating system such as Unix as hosting environment has facilitated the development of a functional system. There was no need to recompile web servers in fact servers code is downloaded from its development web site in its compiled version.

However there are still some pieces to be redesign or completed to create a fully functional web service deployment system:

- deployment of a web service using the X-bone is tightly coupled to the deployment of an IP overlay, which implies that:
  - a network overlay has to be deployed prior deploying a web service; it has an advantage, the underlying network topology does not have to be discovered by the deployment manager since it will use the overlay topology which is known. Disadvantages are that since network overlays are created using tunnels, hosts must be able to set up tunnels and deployment managers have to manage overlay addresses. Also traffic throughput is reduced because tunnels require encapsulation and de-encapsulation of network packets.
  - the service topology has to be the same as the overlay topology; which need not to be, a service can have a mesh topology being on top a ring overlay.
- As it has been commented the deployment service does not make deployed service manageable.

Lastly we feel group creation and group organization mechanism needs further investigation. F.e one of the main drawbacks of the X-bone is its use of multicast for resource discovery, which although is a very elegant solutions its availability is very limited. As well only very basic topologies are supported and scalability of topology configuration mechanisms needs to be elaborated.
Work in progress includes:

- Web service deployment is to be decoupled from overlay service deployment, which will require a network topology discovery function and a mapping function which maps service topology (and performance) requirements into networking requirements, including persistent storage reservation.
- Generalize the service creation API to differentiate from application layer service creation to network layer service creation.
- Implementing mechanisms to allow created services to export its management interfaces.
- Investigate further group creation and group organization.

7 Acknowledgments

This work was done during a summer stay at USC/ISI.
I would like to thank Joe Touch for the opportunity to visit ISI, and all the people at the X-bone project for their help and support.

8 References

[1] J. Touch, S. Hotz "X-bone: a System for Automatic Network Overlay Deployment" Third Global Internet Mini Conference in conjunction with Globecom´98, Nov. 1998. http://www.isi.edu/x-bone.
[2] David L. Tennenhouse, David J. Wetherall Network Systems group, MIT "Towards an active networks architecture" Proceedings, Multimedia Computing and Networking, San Jose CA, 1996.
[3] Open Signalling Working Group, http://comet.columbia.edu/opensig/.
[4] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. "A hierarchical Internet object cache" In Proceedings of the USENIX 1996 Annual Technical Conference, January 1996.
[5] Wang, Z., Crowcroft, J., "Cachemesh: A Distributed Cache System For World Wide Web," NLANR Web Cache Wk., Boulder, Jun. 1997.
[6] Mark E. Crovella and Robert L. Carter, "Dynamic Server Selection in the Internet", in Proceedings

of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95), August 1995.
[7] Bestavros, Azer. "Demand-based dissemination for Distribute Multimedia Application". Proceedings pf the ACM/ISMM/IASTED International Conference on Distributed Multimedia Systems and Applications. Stanford, CA. August 1995 .
[8]Peter B. Danzig, Richard S. Hall, Michael F. Schawrtz "A Case for Caching File Objects Inside Internetworks" Colorado University Technical Report CU-CS-642-93 1993.
[9] E. Amir, S. McCanne, R. Katz "An Active Service Framework and its Application to Real-time Multimedia Transcoding" Proc. SIGCOMM'98 Sept. 1998.
[10] R.Govindan, C. Alaettinoglu, D. Estrin, "A framework for active distributed Services" ISI/USC Technical Report 98-669 1998.
[11] E. Belani, D. Culler, P. Eastham, C. Yoshikawa, M. Dahlin, T. Anderson, A. Vahdat, "WebOS: Operating System Services For Wide Area Applications" Proc. of IEEE Sym. on High Performance Distributed Systems, July 1998.
[12] Akamai Inc., "Freeflow", http://www.akamai.com/ , Dic. 1999.
[13] M. Rabinovich, A. Aggarwal, "RaDaR: A Scalable Architecture for a Global Web Hosting Service" WWW8 Conference, Toronto May 1999.
[14] A. Heddaya, S. Mirdad, "WebWave: Globally Load Balanced Fully Distributed Caching of Hot Published Documents" Proc. 17th IEEE Intl. Conf. on Distributed Computing Systems, Baltimore, Maryland, May 27-29, 1997.
[15] J. Touch,"The LSAM Proxy Cache - a Multicast Distributed Virtual Cache", 3rd International WWW Caching Workshop, Manchester, England, June 15-17, 1998.
[16] Lixia Zhang, Scott Michel, Khoi Nguyen, Adam Rosenstein, "Adaptative Web Caching" 3rd International WWW Caching Workshop, Manchester, England, June 15-17, 1998.
[17] Ircache "SNMP Management of Proxy/Caches",.http://www.ircache.net/Cache/cache-snmp/ 1999.
[18] J.C. Chuang, "Resource Allocation for stor-serv: Network Storage Services with QoS Guarantees", NetStorage'99, Network Storage Symposium Dec. 1999.
[19] Micah Beck, Terry Moore "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels" 3rd International WWW Caching Workshop, Manchester, England, June 15-17, 1998.
[20] R. Renesse, K. P. Birman and S. Maffeis, "Horus, a flexible Group Communication System", Communications of the ACM, April 1996.
[21] Aurrecoechea, C., Campbell, A.T. and Hauw, L., "A Survey of QoS Architectures", ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol.6 No.3, pg. 138-151, May 1998.
[22] Y.Chae, S. Megur, E. Zegura, S. Bhattacharjee, " Exposing the network: Support for topology-sensitive applications", 3rd IEEE Conference on Open Architectures and Network Programming, Tel-aviv,Israel, March 2000.
[23]Aurrecoechea, C., Lazar, A.A. and Stadler, R., "Towards Building Manageable Multimedia Network Services", Proceedings of the IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS'97), Montreal, Canada, July 1997.