Decentralized Virtual Internet Deployment using DANSE

Venkata Pingali
USC/ISI
4676 Admiralty Way,
Marina del Rey, CA
pingali@isi.edu

Joseph D. Touch
USC/ISI
4676 Admiralty Way,
Marina del Rey, CA
touch@isi.edu

Yu-Shun Wang
USC/ISI
4676 Admiralty Way,
Marina del Rey, CA
yushunwa@isi.edu

ABSTRACT

DANSE (Dynamic Network Synthesizer) is a system and architecture to deploy virtual networks in a decentralized way using network-specific, platform-independent and reusable deployment strategies. DANSE combines network virtualization with decentralized control to provide a simple and generic control architecture in which sophisticated network control algorithms can be expressed. Further, DANSE provides the basic building blocks necessary to create evolvable IP networks. The latter has applications in the network management domain including automated network (re)configuration without flag days and self-organization.

1. INTRODUCTION

DANSE (Dynamic Network Synthesizer) extends Virtual Internets, discussed later, with decentralized control to provide a framework that is minimal and generic enough to program different algorithms, called control strategies to flexibly construct copies of IP networks, bootstrap services within the networks and garbage collect the old networks – all in a distributed way. Advanced network-level operations such as reconfiguration can be expressed as network-specific control strategies. These control strategies are do not disrupt existing/other networks and vary across networks allowing for diversity in the networks created. Virtualization, through its support for co-existence of multiple networks on the same set of hosts, is essential to making the control strategies, which create and manipulate networks, non-disruptive. The diversity in is achieved through a flexible two-tier control architecture consisting of a core service that exports a set of management primitives and a network-specific control service that combines link local information and operations with distributed algorithms.

Virtual Internet architecture and associated mechanisms support deployment of multiple concurrent networks with overlapping address and name spaces on a given set of hosts. VI architecture effectively makes networks first class objects that can be created, destroyed and modified algorithmically. X-Bone [21], a tool built to deploy VIs, allows link-layer technology-independent description of the network and uses hierarchical deployment strategy. The need for going beyond existing artifact, X-Bone, arises from the fact that greater flexibility is needed in terms of what is being deployed, how and when. Self-organizing networks, in which nodes can initiate a join, leave or change in topology, require that every node be able to initiate and coordinate the change. In X-Bone terms, every node is both an overlay manager and a resource daemon. Further, coordination-logic is specific to the VI. A set of hosts can support multiple VIs each of which has a different control strategy. The VI deployment mechanism as a result should be minimal, stable and generic enough to allow wide ranging approaches. We suggest that the deployment service should split into a generic VI-independent resource management service and a network-specific control. The management service handles virtual node and link creation and cross-network dependencies. The control, on the other hand, determines when and with whom the links must be established, and how to bootstrap higher-level services.

The rest of the paper is as follows. Section 2 describes the architecture and various issues of DANSE. Section 3 discusses the design and implementation of a DANSE artifact that is based on clonable stacks [29]. Section 4 discusses possible application scenarios. Section 5 identifies related work and Section 6 identifies open issues and future work.

2. ARCHITECTURE

This section describes the architecture of the DANSE. The first sub-section is a brief recapitulation of the principles of Virtual Internet (VI)[14]. Virtualization makes the network a first class object that deployed, moved and destroyed in a programmatic and non-disruptive way.

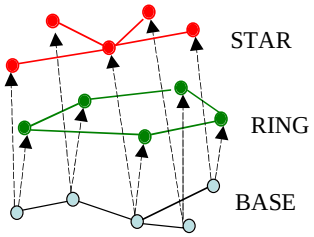

Figure 1 Multiple concurrent virtual internets can be deployed each with separate name space, address space and network properties such as topology.

2.1 Virtual Internets

A Virtual Internet (VI) is a virtual version of the Internet in which virtual hosts and routers are connected by IP-encapsulation tunneled links over the existing Internet (Figure 1,Figure 2). The idea is similar to virtualization of memory. Each VI is complete with addressing and routing, and applications running inside the VI are unaware of anything outside the VI. Some key principles underlying VIs include:

1. VIs are composed of VRs and VHs connected by IP encapsulated tunnel links, emulating the Internet architecture Virtual hosts are data sources and sinks; only VHs increase or decrease the number of headers on a packet (i.e., encapsulate or decapsulate). Virtual routers are data transits; only VRs transit packets without changing the number of headers

2. VIs are completely virtual, decoupled from the base network on which they are deployed. VIs support concurrence. VIs support revisitation

X-Bone[21] is a tool to deploy concurrent VIs in a network. It has a simple architecture consists of a central Overlay Manager (OM) that coordinates the deployment and a Resource Daemon (RD) on each host that configures the hosts to be part of the VI based on the instructions from the OM. X-Bone, and extensions support a variety of discovery mechanisms including multicast, an explicit host list and distributed registry[28] and alternative forwarding mechanisms[18]. The control models supported include centralized and peer-to-peer driven that is decentralized in some aspects such as link construction and topology management. VIs deployed use global addresses managed by a centralized address server and computed at deployment time.

VIs deployed by X-Bone use two-layer tunnels. The first and second layers are equivalent to a link layer and network layer respectively in the VI. The two layer tunneling achieves isolation between VIs and handles the subtle but difficult case of revisitation in which two virtual hosts belonging to the same VI reside on one physical host.

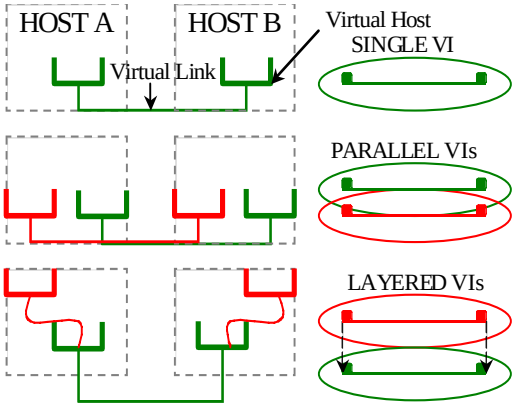

Figure 2 Shows virtual hosts and tunnels instantiated in three simple cases - single, parallel and layered two-node networks.The links indicate the flow of packets.

2.2 DANSE

DANSE combines virtualization and decentralization to enable flexible construction of VIs. DANSE provides a minimal control architecture in which a wide range a network deployment and management strategies can be expressed. These strategies combine resources, mechanisms and deployment rules that enable complex network services to be built. These control capabilities are distributed between a network-independent resource management module and network-specific control modules, as discussed later. Resource management module ensures consistency and interoperability between networks and the control modules coordinate network-specific operations across hosts.

We first identify the class of VIs that are deployed using DANSE, called Augmented VIs and then elaborate on DANSE itself.

2.2.1 Assumptions

Assumptions made here include: (1) the link layer is able to generate a unique link local address per physical interface, (2) host operating system has support for virtual network identified by the VI earlier

including virtual hosts and tunnels, and (3) backward compatibility is preferable but not necessary.

2.2.2 Augmented Virtual Internets (AVI)

An Augment Virtual Internet (AVI) is a Virtual Internet with subtle but important modifications related to tunneling and naming. All other properties such as two-level tunneling, use of a single addressing protocol (IP version) throughout the VI, hop-by-hop forwarding, and global addressing are inherited.

Tunneling. AVI tunnels use link local addresses within the VI, and may be in the base network as well. DANSE relaxes the assumptions regarding the global addressing within each VI for two reasons. First, end-to-end addressing service is VI-specific. That is address distribution is yet another application that is run within the VI. Second, existing VI deployment approach requires global coordination for topology control and addressing. The deployment process is simplified because link establishment can be performed using local information. Another difference is that the base network associated each link is link-specific and not network-specific. The latter affectively creates "hybrid" virtual hosts in which two links of the same virtual host may be overlaid on top of two separate base networks and hybrid VIs that cross base network boundaries. Existing artifact, X-Bone, supports only strict layering in which nodes of a VI have to exist to the base network. AVIs may use end-to-end or link-local addresses in the base network but always use link-local addressing within the VI. End-to-end addresses are computed by an AVI-specific addressing service and configured to be aliases on the network interfaces in the second-layer tunnel. Once in the network, the network-specific bootstrap services can be instantiated that compute the rest of the configuration.

Names. There is no explicit support for names in the VI architecture through the implementation incorporated it. AVI formalizes the support for naming to allow for interoperation between networks and implementations of the architecture. An Augmented Virtual Internet (AVI) has at least one network name, an opaque string that is known to a subset of nodes of the AVI. Different nodes may use different names to refer to a given AVI. A single network may have multiple names reflecting the administrative or other logical boundaries. However, at least one name must be shared between nodes that are one away from each other. The architecture does not specify ways of enforcing a single consistent name or resolving name conflicts across disjointed networks. Duplicate detection mechanisms may built using the primitives provided by the DANSE architecture. Node names are required for mapping across networks and for identifying specific virtual host when node revisitation is enabled. The architecture does not specify a form for this node name. Additional knowledge about the nature of the AVIs deployed could be exploited for this purpose. Each AVI also has a class name that identifies the nature of control. This is necessary to build control daemons that are network name-agnostic.

Default Base Network. The DANSE architecture makes the assumption that lowest level network is that includes all nodes in the world. The addressing in the default network may be link local or global. AVI layered on top of the default network are limited by the reachability properties of the default network. In case of global addressing and/or the special case of on-demand links such as dial up, mechanisms outside the scope of DANSE are expected to handle them.

2.2.3 Resources

DANSE identifies the following host-local and link-local resources for explicit management:

1. Virtual hosts
2. Virtual links
3. Address space for tunnels
4. Name spaces (node, network and class)

These resources have to be managed "outside" of AVIs because they all have built-in cross network dependencies and limits. Virtual host and link creation involves allocation of system resources including interfaces, security contexts, cpu time and disk space. In a given VI, the network-layer tunnel is located within a virtual host corresponding to that VI and the link-layer tunnel is located within a virtual host in the base network. Correct functioning requires that the base-network addresses of network-layer tunnel must correspond to the tunnel addresses set on the link-layer tunnel. The address dependency effectively crosses virtual network boundaries. Tunnel establishment involves negotiation between neighboring physical hosts. Network names and virtual host's node names are used to identify the end point of the virtual link accurately. Each network is assumed to belong to a particular class that broadly identifies the characteristics of the network. In practice this class name is used to identify appropriate control service.

DANSE does not specify important implementation details such as the specific address spaces used for the tunneling, the nature of control over these resources, the interface to drive allocation of these resources or the host-mechanisms to limit the capabilities of the management processes within each VI.

2.2.4 Mechanisms

DANSE requires that two mechanisms be supported by the network-independent part of any instantiation of the architecture. These are mechanisms that the network-specific control processes will use to manage the network. They include:

1. A way to discover/create/destroy virtual hosts and links, with link-local auto-configured addresses, on the local host
2. A minimal best-effort coordination mechanism within each AVI

Network-specific control processes can use these two mechanisms to bootstrap themselves and higher-level services.

2.2.5 Rules

DANSE-deployed virtual networks satisfy the following rules:

1. All networks deployed are augmented virtual internets.
2. All networks except a special default network is overlain on another network
3. Base network of a link is link-specific
4. A link that crosses VI boundaries requires inter-network routing layer.
5. Each network has at least one name
6. Node names are network-specific

Rules for deployment are as follows:

1. Deployment can be initiated from any node in the network that can reach atleast one other node in one hop.
2. Negotiation from link establishment in any given AVI occurs outside the AVI.

Link Establishment. Just like in the X-Bone system, DANSE requires that the negotiation for the link establishment for a particular VI occurs *outside* the virtual network, typically in the base network over which the tunneling occurs. This allows for immediate testing of node reachability. The base network is required to support end-to-end addressing if the link-peers are more than one hop away in the base network. Either end can initiate the link establishment process. The process might require multiple initiations and iterations until parameters such as tunnel addresses satisfy the required constraints at both ends. Upon successful negotiation, the interfaces are configured and state updated. A physical node may have multiple virtual nodes from a single network. The link establishment request not only has to specify the network name but also a node name within the network. Constraints on the underlying network can be expressed by in the design language. This could potentially cover network properties such as topology and security. This is a per-link decision and that could result in possible loops. Discovery of these loops and preventing such loops is part of the future work.

Cross-network overlays. DANSE supports cross-network overlay deployment (Figure 3), i.e., allows for nodes from different networks to be tunnel end points. A default implementation would use a single tunnel that crosses network boundaries. However, this adds to the complexity and need for global knowledge and network address translation. Instead DANSE deploys a network that serves as an intermediate routing layer on top of which the link is constructed. This network has three nodes and two hops with the middle node being a hybrid node. This hybrid node has presence in two networks and has one link each in each of the networks. This network requires end-to-end addressing. The link establishment, therefore, triggers a network deployment. The architecture does not specify any optimization approaches if there are multiple such links between the same host pair or some other host pair between the same networks.

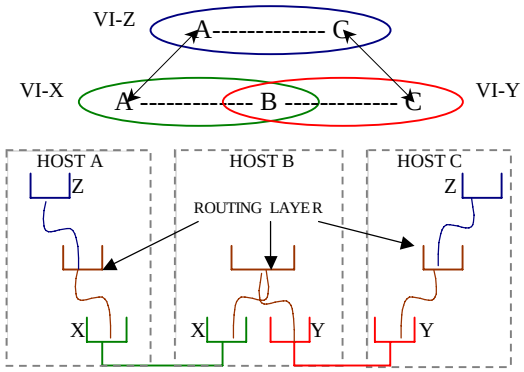

Figure 3 Shows a network Z overlaid upon two different networks X and Y. A-C link of network Z crosses network boundaries. This requires an inter-network layer, which is also a VI, to avoid translation.

2.3 Example: Emergency Network

Consider the deployment of an emergency network. The scenario is characterized by uncertainty in the number of hosts, topology and non-availability of expert administrators to configure the hosts and routers. The network is expected to be self-organizing, adaptive and inter-operable with the Internet.

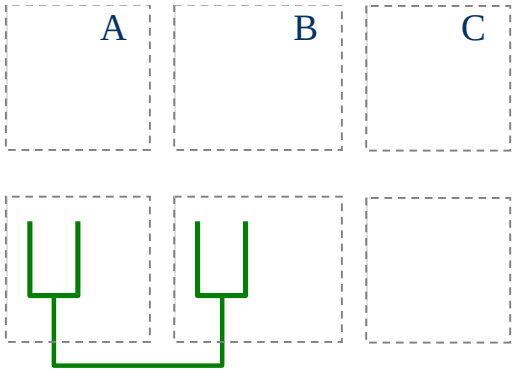

Figure 4 Hosts A and B, that do not have a link between them initially, discover each other and initiate a new trivial network

The configuration process starts at any given node upon achieving one-hop reachability through other mechanisms such a human physically connecting the cable to the host or the host registering with the wireless access point. IPv6 configures link-local network-layer fe80:: address that is used for tunneling and negotiation. Each host is assumed to run the DANSE resource management service and self-organizing network control service, henceforth referred to simply as the control service that directs the network self-organization process. The DANSE service broadcasts the host's presence on the link and learns about other nodes. When two nodes learn of each other's presence (

Figure 4), the control daemons running on both hosts request the DANSE service running on the local host to create a new virtual network with the neighboring host. The DANSE service instantiates a new virtual host, clonable network stack [29] in our artifact, negotiates addresses with the neighboring node, configures appropriate tunnels in the virtual host. The hosts initiate this network creation in an uncoordinated fashion. Depending on the implementation of the DANSE service and the timing, zero, one or two networks are created. This is an efficiency, and not correctness, issue because the creation of the network has no side effects on the rest of the nodes or networks. In case no network is created, either end can restart the process after a random delay. If two networks are created, one can be destroyed by control service.

At this point a trivial two-node network is ready to use. The network comes preconfigured with interfaces and link local addressing. This network does not have, and also does not require at this point, services such as global addressing or routing. Additional services have to be instantiated separately using the intra-network coordination service provided by the DANSE implementation within the new network.

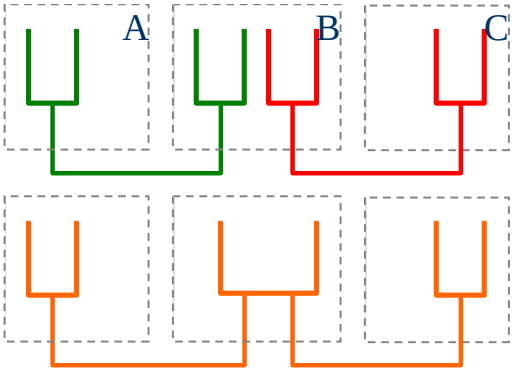

Figure 5 Hosts B and C also discover each other and form a second trivial network. The two networks are eventually merged in a systematic way to form a single three-node network.

If a new physical host is now detected (Figure 5), there are two possibilities. Either the new node can join the existing network or a new network can be formed with one of the nodes from the existing network. The control service decides the course. Over time, multiple networks may be formed. The control service coordinates the process of network merging where a single unified network is constructed from the members of two or more existing networks in a multi-step operation. The process is typically initiated on a node that participates in multiple networks. The implementation of the control service can vary from simple trial and error with no control over topology to sophisticated topology and plan-based algorithms. In all cases, the transition from one network to another is seamless and non-disruptive. The smooth migration from one generation of the network to the next is not possible in existing Internet because mutual exclusivity of the old and new configurations.

2.4 Discussion

The issues raised by DANSE can be broadly classified as those related to virtualization and those related decentralization of control. DANSE addresses the mismatch between the control capabilities that exist today and those required to express sophisticated network control strategies through a combination of virtualization and decentralization. DANSE trades complexity for simplicity, scalability and uncertainty. The state of the network must be discovered and control expressed through indirect operational rules.

2.4.1 Virtualization

Bootstrapping Services. AVIs have minimal configuration – a set of links pre-configured with link-local addresses. To enable bootstrapping of higher layer services, the architecture requires that at least one coordination mechanism be deployed. For example, a simple broadcast based on the distributed consensus algorithms with minimal complexity such as the Echo algorithms[23] can be instantiated at the time of virtual network deployment. This simple coordination mechanism can be used to bootstrap other services. For example, a broadcast-based DHCP service can be implemented in which the DHCP clients broadcast their DHCP request to all nodes. The DHCP server can then respond, using the same broadcast mechanism. Similarly a routing server may use the coordination mechanism to discover the topology and determine the routing parameters. Such service instantiation can be network-specific, programmable and reusable.

2.4.2 Decentralization

DANSE embraces decentralized knowledge and control and the uncertainties that arise in terms of state and dynamics in the network. With decentralization, networks are created and modified in a distributed way. The control is specified and exercised in an indirect way through local rules that capture constraints, invariants and goals. Knowledge about the network is discovered as opposed to specified.

Control. The control of the network is based on local rules for the inter-network and intra-network daemons. Rules can cover (1) pre- and post-conditions for allocation of resources such as names, links, and hosts including admission control rules (2) resolution of naming and addressing conflicts (3) Dynamics of the networks such as frequency of reorganization (relative sizes could be a hint) and persistence of networks created (4) life times and cleanup.

Knowledge. Since the network will be deployed in a decentralized fashion, the information and resources must be discovered. There is no easy way to find out if two networks with the same name are the same network that has either reachability issues or partitioned or whether they are two separate networks

Integration. AVIs are isolated from each other. DANSE does not address the issue of integration of a sub-network beyond co-existence. The challenges this creates include having to create large-scale AVIs even when the changes are small or local and extending applications to handle multiple AVIs.

2.4.3 Issues Not Addressed

Several issues are addressed in this work. First, software distribution aspects are also out of scope. Second, the architecture does not address mobility. We do make the observation that mobility can be supported with existing tunnel establishment and address negotiation mechanisms. The virtual address that the application sees need not change during the process of renegotiation making mobility relatively transparent. Third, auto-configuration requires new application architecture that supports multiple contexts within the application-network space. Last, the policy framework for managing the creation and destruction of the AVIs is also unspecified by the architecture.

3. PROTOTYPE

An initial prototype that reuses code from the X-Bone software system has been built. The system, written in Perl, uses a combination of user-level control processes and custom kernel extensions to FreeBSD.

3.1 Building Blocks

Three important building blocks for the prototype include the intra-host isolation mechanism, the network layer protocol and distributed algorithm used for coordination. We discuss each of them below in more detail.

3.1.1 Clonable stacks and extensions

Clonable stacks [29] for FreeBSD supports full-fledged virtual hosts and routers by integrating FreeBSD Jails with replicated kernel network data structures such as the TCP control block list, routing table and interface list.

In the virtual internet context, two-layer tunnels are used and IP packets cross the network contexts. The network-layer tunnel is configured in the overlay and link-layer tunnel in the base network. After the network-layer tunnel encapsulates the packet with the link-layer tunnel's outer header, the packet is injected in the base network. The packet is then forwarded to the link-layer tunnel, which in turn encapsulates the packet. This crossing of contexts was implicit in the original VI deployment systems, X-Bone, because there was no kernel support for virtual hosting and therefore the issue was never addressed. Custom kernel modifications to the clonable stacks explicitly mark the tunnel interfaces as being link or network layer tunnels and in case of network-layer tunnels, an explicit identification of the underlying network is added. A different, but minor, extension allowed the creation of tunnels using link-local addresses.

Clonable stacks implementation supports processes that can simultaneously exist in multiple virtual hosts and listen to sockets from multiple hosts. This capability is used in the implementation to reduce number of processes and association communication overhead and code complexity.

3.1.2 IPv6

The architecture is agnostic with respect to the version of IP used. Neighboring nodes must agree upon a mutually acceptable range. The current implementation uses IPv6 as the default-addressing scheme and a predefined fixed block of IPv6 addresses for tunneling. IPv6 has link local addressing that until recently was missing from IPv4. Stability of the link local address is necessary because tunnels are constructed with them as the outer addresses. Current techniques to obtain IPv4 link local addresses make the address a function of time. This is however not a huge problem because certain private address spaces 10.0.0.0/8 are large enough to support many concurrent networks and can be reused across links. The lower bits of a hash of the name can be used to generate stable link local address.

3.1.3 Echo algorithm

DANSE uses a lightweight distributed consensus algorithm, Echo algorithm [23]. The algorithm is best effort and has low complexity. It works by establishing a spanning tree at run time to ensure that all nodes receive a given message or that particular run of the algorithm fails. The only knowledge required for the correct functioning at each node is the host-local information regarding links – the number and next hops. Because the algorithm does not use messages that cross multiple hops, end-to-end addressing is not required. The module is a plug-in and a more robust implementation that provides support for lost packets and NACKs can also be used instead.

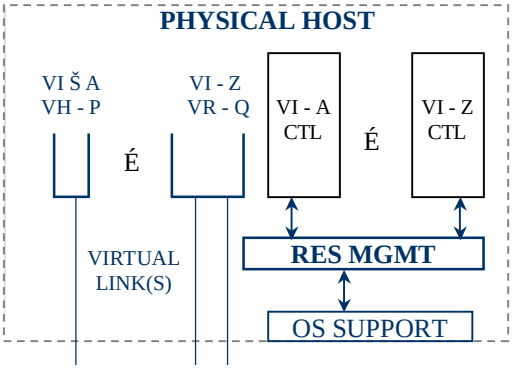

Figure 6 Network-specific control daemons communicate with network-independent host-local resource management service to configure virtual hosts and links in a decentralized way. (Notation: VI: Virt.Internet,VH:Virt.Host, VR:Virt.Router)

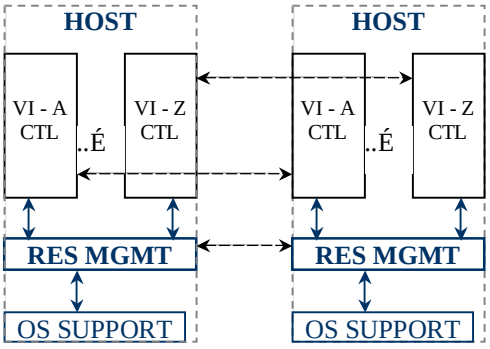

Figure 7 Cross host interaction. Resource management service instances coordinate virtual host and link creation. Network-specific control instances coordinate with each other and with host-local resource management service for network-level operations.

3.2 Design

The user-level processes consist of two logical modules (Figure 6) – resource management module and control module.

Resource management module: In general, this handles all aspects that need information or capability that cannot be obtained from within the virtual network or when there are cross-network dependencies. There is one module per physical host. The functions include management of virtual hosts and links, one-hop multicast-based discovery, basic link establishment including address negotiation, persistent network database, and access control. The resource management is driven by network-specific control daemons and does not initiate operations independently other than discovery.

Control module: Each network has a separate control module that encodes the high-level design of the network. Through appropriate calls to the Resource management module and coordination with other control modules of the same network, the control module implements multi-step multi-way coordinated operations (Figure 7) including network creation, destruction, move, and duplication. A distributed consensus algorithm such as the echo algorithm is used to achieve the multi-way coordination. This module interacts with the user and provides high-level abstractions to the user. The control module can be further split into a intra-network coordination service

and class-dependent control service. Alternatives are being explored here.

Graphs may be expressed in a compressed form. This could arise because the size of the graph is too big, or if the number and distribution of the nodes is not available at the time of the time of the network deployment. The language to express the topology or properties depends on the design of the policy module. There could be multiple such modules at any point in time.

4. DANSE APPLICATIONS

DANSE makes the network deployment and subsequent management more programmable by adding a level of indirection in the form of reusable policy and intra-network daemons that are type-dependent. This has two advantages. First, custom extensions to the VI deployment system can be reused across sites. Second, the dynamics of the network can be more sophisticated than what a single implementation can provide. There a variety of situations where programmability might be useful:

4.1 Customizable Testbeds

Experimental Testbeds have been the original motivation of the Virtual Internet [14] and the artifact, X-Bone system. Other testbeds include Emulab [30] and Planet-Lab [12] In each case a reusable experiment-specific script is used to customize the environment that is instantiated. However, the testbed itself is not programmable and one size does not fit all needs. For example, in Planet-Lab, one cannot experiment with alternative addressing or routing. Here the knowledge of when to add nodes, what is the minimal set of services and configuration required is dependent on the testbed.

4.2 Self-Organization

A self-organizing network with minimal network administrator involvement has been an ideal for a long time. While this problem has been address in specific environments such as wireless ad-hoc networks, a generic solution does not exist for Internet. DANSE can support self-organization in a straightforward way. A self-organizing network control can be written that can driven by simple rules such as: (1) Two nodes which are one hop away (physical or logical) can create a new network and (2) Two networks can merge if they have at least one common node. Such a daemon is under development.

4.3 Automatic (Re)Configuration

With DANSE the cost of reconfiguration is reduced significantly and the connections could continue without major disruptions. Assuming that the production network itself is a virtual Internet – which is a major assumption – a network reconfiguration attempt might include the following steps: (1) Create a new network, with properties derived from the existing network including topology. (2) Refine the new network over time independent of the old network. If that doesn't work, go to step (1). (3) When done, migrate users to the new network. (4) Move the old network over the new network. (5) Garbage collect the old network when done. The procedure is non-disruptive and amenable to automation. Among the issues that must be addressed include application migration between versions of the network and cross-VI communication.

Similar to copy-on-write mechanisms in operating systems, a network redesign mechanism based on VIs can be built upon a copy-on-write network where a network change would involve creating a copy of the network and applying the modifications to the copy. Here we assume that every network is a VI. When the new network has been tested, then users are migrated over time to the new network. The old and the networks are decoupled in time. The key advantage that this approach gives us is that we can construct arbitrary number of new networks each possibly using a different algorithm. These individual networks can be refined in parallel and tested "in the field" before committing to one or more resulting deployments [22].

5. RELATED WORK

DANSE touches upon many areas including internet architecture, automatic network configuration, and evolutionary systems. We discuss related work in each of the areas below.

5.1 Internet Architecture

Virtualization is increasingly an important but still an optional component of the Internet. Virtualization of memory [17] is ubiquitous and that of host [24] [25] and software [27] is becoming more common. X-Bone and its variation Global X-Bone use centralized deployment model. P2P-XBone[18] supports decentralization of control but within the larger XBone system architecture. DANSE generalizes the P2P-Xbone by completely eliminating the overlay manager, simplifying the VI that deployed and building in relationships between the VIs deployed. DANSE suggests a micro-kernel like structure for the control architecture of the Internet.

On the specific issue of communication across networks, the approach of a routing layer with hybrid nodes differs from existing proposals that depend on header modifications [26] or translation [13].

This effectively revisits an old discussion on application-specific memory management. In case of virtual memory, the discussion has effectively ended with few applications needing or seeking control over memory management. Both the techniques have improved as well as the gain from the additional complexity was not considered substantial. The discussion is yet to happen in the case of network virtualization.

5.2 Complexity

The complexity of network configuration has been growing with time because of increasing number devices and modules within, module features and the number of possible interaction. There are relative few measures [4] of the same. It harder to compute deterministically the correctness or performance of a specific configuration and therefore network configuration is increasingly a search process in which various alternative configurations are tried and successful ones retained. Internet is already being discussed as a complex system[3] and observations from other domains including system science encourages us to think in alternative ways to handle complexity and emergent properties through hierarchical abstractions and stable configurations at a network-level [1][2]. DANSE does not provide the solutions to complexity issues but rather provides a framework to experiment with alternative approaches at low cost. DANSE allows a variety of controls including agents driving self-organization. The general idea is reused but not the specifics. Evolvability has been studied in software systems context [5] but they have focused on single systems. Evolution of networked systems is more difficult and recent work on software updates with versioning support [11] is promising. Network-level evolvability complements the software-level solutions.

5.3 Automatic Configuration

The two basic approaches have been (1) Configuration-database which that either stores the actual configurations [6] or parameters to be used along with a model database [7]. In both cases, a central server generates the appropriate configuration for each host that is then distributed to the individual hosts (2) Agent-based approach in which an agent running on each host continuously refines the configuration until the required goal is achieved [15][20]. The problem with the first approach is two fold. First, the complexity of the configuration is hidden in the database/models and not eliminated [16]. Second, there is no easy way to recover if the configuration is wrong. DANSE is closer the agent-based approach. The cost of experimenting with alternate configurations is lower because the experimentation can be performed on an experimental VI instead of production VI.

Autonomic computing [19] is a control-theory approach to tuning software systems. It is not unclear how this might help obtain the base configuration necessary before tuning can be done and in a distributed fashion.

While Simple Network Management Protocol (SNMP) [8] has been around for a long time, it is mainly used to gather information to gain insight into network patterns than configuration of systems. The Management Information Base (MIB) is too low level an interface and the support for MIBs vary in quality. Recent discussions in IETF have focused on the interface [9] or in a limited context [10].

6. CONCLUSION AND FUTURE WORK

This paper mainly discussed the architecture of the deployment system. Developing artifacts consistent with this architecture will be a major part of the future work including the specifics of the daemons, a language for network-level service deployment and advanced network control written as policy daemons.

7. ACKNOWLEDGMENTS

The X-Bone group for their inputs.

8. REFERENCES

[1] Simon, H. A. *The Sciences of the Artificial*. MIT Press, Cambridge, MA.1981.

[2] Heylighen, F. *Principles of Systems and Cybernetics: Evolutionary Perspective*. Cybernetics and Systems, World Science, Singapore, 1992.

[3] Willinger, W., and Doyle, J. *Robustness and the Internet: Design and Evolution*, Unpublished manuscript. March 2002. URL http://netlab.caltech.edu/internet/.

[4] Brown, A. B., and Hellerstein, J. L., *An Approach to Benchmarking Configuration Complexity.* Proceedings of the 11th ACM SIGOPS European Workshop, Leuven, Belgium, September 2004.

[5] Lehman, M. M. *Laws of Software Evolution Revisited.* C. Montangero, editor, Software Process Technology (EWSPT 96), Volume 1149

of LNCS, Springer-Verlag, Nancy, France, 1996. 108-124

[6] Hewlett-Packard. *OpenView Operations Manuals*. 2003. http://ovweb.external.hp.com/lpe/doc_serv

[7] Caldwell, D., et al. *The Cutting EDGE of IP Router Configuration*. Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II), Cambridge, MA, November 2003.

[8] Case, J., Fedor, M., Schoffstall, M. and Davin, J. *Simple Network Management Protocol*. STD 15, RFC 1157, May 1990.

[9] R. Enns, *NETCONF Configuration Protocol*. IETF Work in Progress (draft-ietf-netconf-prot-09), October, 2005.

[10] Williams, A. *Requirements for Automatic Configuration of IP Hosts.* IETF Work in Progress (draft-ietf-zeroconf-reqts-12.txt), September, 2002.

[11] Ajmani, S. *Automatic Software Upgrades for Distributed Systems.* Ph.D. Thesis, MIT, Boston, MA, 2004.

[12] Peterson, L. L., Culler, D., Anderson, T., and Roscoe, T. *A Blueprint for Introducing Disruptive Technology into the Internet*. Proceedings of HotNets-I, October 2002.

[13] Crowcroft, J., Hand, S., Mortier, R., Roscoe, T., and Warfield, A. *Plutarch: An Argument for Network Pluralism*. ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA'03), August, 2003.

[14] Touch, J., Wang, Y., Eggert, Y., and Finn, G. *Virtual Internet Architecture*. In ACM SIGCOMM Workshop on Future Directions in Network Architecture, Karlsruhe, Germany, August 2003.

[15] Burgess, M. *Cfengine A Site Configuration Engine*. USENIX Computing systems, Vol8, No. 3, 1995

[16] Amey, P. *Logic versus Magic in Critical Systems*. Ada-Europe 2001: 49-67

[17] Denning, P. J. *Virtual memory*. ACM Computing Surveys (CSUR), 2(3):153--189, 1970.

[18] Fujita, N., Touch, J., Pingali V., Wang., Y., *P2P-XBone: A Virtual Network Support for Peer-to-Peer Systems.* Technical Report ISI-TR-2005-607, USC/ISI, September 2005.

[19] Kephart, J. O. and Chess, D. M. *The Vision of Autonomic Computing*. IEEE Computer, 36(1) IEEE, January 2003. 41—50.

[20] Hori, K., Yoshihara, K., Horiuchi, H. *Automatic Configuration of IP Networks and Routers*. KDDI Labs, Japan

[21] Touch, J., and Hotz, S., *The X-Bone*. Third Global Internet Mini-Conference at Globecom Sydney, Australia Nov. 8-12, 1998 pp. 59-68

[22] Bar-Yam, Y. *About Engineering Complex Systems: Multiscale Analysis and Evolutionary Engineering.* Engineering Self Organising Systems: Methodologies and Applications, S. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, R. Nagpal (Eds.), ESOA 2004, LNCS 3464, Springer-Verlag, 16-31, 2005.

[23] Andrews, G.R. *Paradigms for process interaction in distributed programs.* ACM Computing Surveys 23,1(March 1991), 49-90.

[24] Smith, J. E., Nair, R.*The Architecture of Virtual Machines*. IEEE Computer 38(5): 32-38 (2005)

[25] Barham, P., Dragovic, B., Fraser, K., Hand, S., harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. *Xen and the Art of Virtualization*. In Symposium on Operating Systems Principles (SOSP '03) Oct. 2003.

[26] Francis, P. and Gummadi, R. *IPNL: A NAT-Extended Internet Architecture.* Proc. ACM SIGCOMM, San Diego, CA, USA, August 2001, pp. 69-80.

[27] Potter, S. and Nieh, J. *AutoPod: Unscheduled System Updates with Zero Data Loss.* Abstract in Proceedings of the Second IEEE International Conference on Autonomic Computing (ICAC 2005), Seattle, WA, June 13-16, 2005, pp. 367-368.

[28] Touch, J., Wang, Y., Pingali, V., Eggert, L., Zhou, R., and Finn, G. *A Global X-Bone for Network Experiments*. Invited paper, Proc. of Tridentcom, Trento, Italy, February 21-25, 2005, pp. 194-203.

[29] Zec, M., *Implementing a Clonable Network Stack in the FreeBSD Kernel*, in Proceedings of the 2003 USENIX Annual Technical Conference, FreeNIX track, San Antonio, June 2003.

[30] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Barb, C., and Joglekar, A. *An Integrated Experimental Environment for Distributed Systems and Networks*, Proc. of OSDI, Boston, MA, December 2002