

Virtual Enterprise Networks: The Next Generation of Secure Enterprise Networking

Germano Caronni, Sandeep Kumar, Christoph Schuba, Glenn Scott

Sun Microsystems Laboratories
901 San Antonio Road
Palo Alto, CA 94303, USA

E-mail: `Firstname.Lastname@Sun.COM`

Abstract

We present a vision of computing environments in which enterprise networks are built using untrusted public infrastructures. The vision allows for networks to dynamically change depending on the need of their users, rather than forcing the users to build organizations around networks. This vision is realized through a design abstraction called Virtual Enterprise Networking, or short Supernetworking. A first prototype of such a Supernet has been implemented on Linux.

Supernetworking introduces a new layer of abstraction in a layered model of computer networking. The Supernet layer sits directly above the network layer and includes its own addressing structure and security services which protect all data transmitted by the network layer.

A key component of a Supernet is communications tunneling. Instead of the traditional two endpoints, our tunnels have as many endpoints as there are computers participating in a Supernet. While tunneling has been repeatedly used to implement infrastructure services such as multicasting, virtual private networks, and support for mobility, we distill these technologies into a single, simple abstraction.

This new abstraction enables the ability to out-source network infrastructure services in a transparent and secure manner, mobility, and the creation and administration of secure ad-hoc virtual computer networks.

1 Introduction

Remote and reliable computer networks have already begun to transform the way many organizations manage themselves. Such organizations have come to depend upon computer networks to conduct their business. Telecommuting to work has become popular, often-times necessary, for those who cannot be at a fixed location at all times.

To take full advantage of networked computing and information exchange, organizations have had to adapt to the architecture and peculiarities of the underlying network implementation, instead of the network adapting to them. Businesses today are often required to build and maintain an expensive enterprise network infrastructure that typically has little to do with their core business and expertise. For small organizations, creating and maintaining such an infrastructure is often untenable. Even for large organizations that have the means to create and support enterprise networks, such attempts often lead to duplication of effort since each organization produces a custom solution to common sets of problems.

Because contemporary enterprise networks are geographically bound, several methods of remote access have been devised which introduce the problem of securing the enterprise network from external sources. Remote users have their remoteness thrust in their face, and consequently devise ad-hoc mechanisms to handle the problems introduced by remoteness. Such actions can and often do create new security vulnerabilities.

A popular example of how to secure an enterprise network is a network firewall. Firewalls attempt to enforce security policies on communication traffic entering or leaving a network policy domain [Sch97]. But firewalls are insufficient because they protect primarily against external threats, they hinder ubiquitous access, and their mechanisms are inadequate to provide anything but coarse-grained control.

While firewall technology enforces security policies at the network boundary, organizations may exist in multiple locations, and will want to securely connect their internal networks using the Internet. This is the driving force behind virtual private networks (VPN.) VPNs are built using network layer security features, such as the IPSec authentication header (AH [KA98a]) and encapsulating security payload (ESP [KA98b].) VPNs, however, use these features mainly on a network-to-network basis to securely tunnel all

traffic between participating networks. Individual computers can participate in a VPN, but using the VPN in this manner is not scalable. Overall, the generation, maintenance, administration, and dissolution of secure VPN communications is costly and cumbersome.

In this paper, we propose a new model for *secure* enterprise networks, giving organizations the ability to organize themselves and their computers in network-independent ways. Organizations will be able to rely on generic service providers that can offer the desired enterprise infrastructure securely and reliably. Ultimately, they can take advantage of economies of scale when it is possible to lease communications, storage, and computing, in much the way society currently uses the infrastructure provided by the gas, telephone, and electric utilities.

2 The Vision

Imagine users, associated with one or more organizations, able to access each organization's information infrastructure in a secure fashion from any location on the Internet. Secure here means that authenticity, access control, auditing (where necessary), and confidentiality are assured. The hardware components of the organization's infrastructure may be at locations which may not be owned by the organization itself. For example, the organization could have arrangements with multiple providers for disk space and backup services, with other companies providing e.g., network cabling, routing, and Internet access.

Except for a tiny set of superusers, a worker should be able to do whatever he does from *anywhere*, and should not be restricted by physical location.

Ultimately, an organization freed from the details of providing information technology (IT) services could stop caring about most of the IT aspects. Except for the actual information, everything should be securely out-sourced. Even though the network infrastructure, storage devices and computing facilities were out-sourced, providers of those services would not gain access to the organizations data, and would only gain a small amount of control over it.

3 Secure Out-Sourcing

Typical enterprise network services are easily separated into different components. For now, our work concentrates on *secure communications*, and to a lesser extent, on *secure storage*, and *secure computing*, as well as the *management* of their configuration, as depicted in Figure 1.

Stored data and communications are the lifeblood of many organizations. Any organization that out-sources all or part of its computing infrastructure must be guaranteed that the provided security will be at least as good as if the

organization built and maintained the enterprise network itself. As a consequence, we consider security to be the single most important requirement of out-sourcing an enterprise network. Others are described in Section 3.3. The individual infrastructure components are described next.

3.1 Infrastructure Components

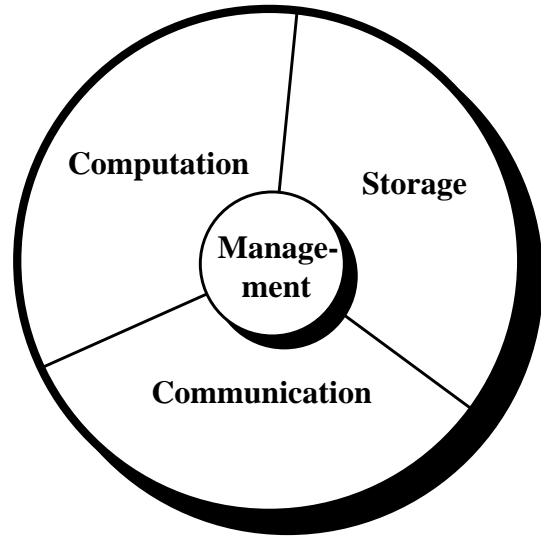


Figure 1. Outsourceable infrastructure components

Secure communications. Secure, out-sourced enterprise network communications are exemplified by the Supernet as a private network over a public communication infrastructure.

Because Supernet-based enterprise networks are topologically independent of the underlying transport network, the public communication infrastructure is used only as a transport mechanism for network traffic. This model differs from the present-day one, in which topology is location dependent—unless the physical and logical network topology are co-located, remote users must handle all problems with remote connectivity themselves.

As the network communications are transported over a public infrastructure, everyone, except for the members of the Supernet, must be prevented from understanding or forging communications traffic. The Supernet achieves this through cryptography.

Secure storage. In a world in which out-sourced data storage will be hosted on untrusted servers, integrity, authenticity and confidentiality of the stored data and

meta-data (file names, directory structures, access permissions) must be assured. This should be achieved with few requirements made of the service provider and without unnecessary trust assumptions. The outsourced data and its meta-data need to be protected so that the provider is unable to modify or delete it without being detected.

Stored data could be important for decades or longer, placing special requirements on the involved key management schemes. Furthermore, changes in the organization's membership require efficient handling of encryption keys and access control. For example, when a former member of an organization gains access to encrypted data stored by the storage provider after he left the organization, he must be unable to decrypt the data.

Secure computing. Perhaps the most difficult part of the vision is the secure outsourcing of applications and computation. Here, a computing provider receives and runs programs without ever seeing them in the clear, having those programs consume and produce data streams which are inaccessible to anything but members of the Supernet. At the same time, the computing provider may also play the role of an applications provider, giving the organization tailored and secure access to trusted applications on demand.

We imagine that this component may require substantial changes to existing system architectures, involving CPUs, memory controllers, and perhaps systems busses with cryptographic capabilities.

Configuration management. The Supernet infrastructure components need to be subject to security policies and controls. The security policy is the interface between what the organization desires and an implementable machine-readable description. While the choice of policy is controlled by the organization, its enforcement is usually a shared responsibility between the organization and the service provider, governed by a contractual relationship. This model assures the separation of policy content and expression and its enforcement.

At a high level, the configuration management component provides the expression, administration, and enforcement of security policies that govern the enterprise infrastructure components. At a lower level, configuration management applies to each component: key management, role and group management, and access control.

3.2 Benefits of Out-Sourcing

The out-sourcing of IT infrastructure components brings with it the following benefits.

Cost effectiveness. If service providers purchase, maintain, and manage their equipment for a multitude of customers they benefit from economies of scale. This advantage would translate into cost savings for customers.

Business Focus. By outsourcing, corporations will be able to maintain their focus on their main activities, rather than being distracted by IT concerns.

Access to current and new technology. A critically important benefit is that service providers can more easily implement new software and hardware, thus providing corporations fast access to state-of-the-art technology.

No provider lock-in. A key problem faced by network administrators today is that of address renumbering and address aggregation resulting from relocation. This partly results from the fact that IP addresses often belong to their Internet service providers (ISP), and not to the organization. When organizations switch ISPs, they have to renumber their internal addresses. The costs of this renumbering essentially force companies to stay with one ISP. With the right technology available, this problem can be eliminated, thus freeing companies to choose providers at will.

3.3 Design Space for Solution

Any solution that offers the out-sourcing of IT infrastructure components is subject to certain constraints that must be satisfied. In addition to the security requirements outlined in Section 3.1, a good solution should have the following characteristics.

Dynamic coalition capability. Today, it is difficult for organizations to collaborate with others wanting to share only portions of their enterprise networks. We desire a technology that enables the secure ad-hoc generation, maintenance, and dissolution of virtual computer networks. This would allow dynamic collaborations, such as small businesses wanting to rapidly deploy partner networks, telecommuters wanting to connect home and office networks, and in general any online community wanting to establish networked workgroups.

Lifetime independence. Ideally we can use the same technology independent of the expected lifespan of the virtual computer network, i.e., short, medium, or long term.

Application reuse. Existing applications must be reusable without modification. The inertia and cost to rewrite or retrofit existing applications into a new environment makes any other approach unlikely to be deployed. For

the Supernet, this requires that the IP addressing semantics be maintained.

Transparency and efficiency. Except for the new functionality such as the new security services offered by the Supernet, it should be indistinguishable for users that they are using a Supernet instead of a traditional Internet. At the same time, a Supernet should be as robust as the underlying network infrastructure.

Scalability. Because IP's structure has many mechanisms built in to enable scaling, IP networks can easily grow to become large networks. While Supernets can take advantage of these mechanisms because they rely on IP networks for packet transport services. Some Supernet specific mechanisms, such as the group key management, need to include scalable mechanisms to allow efficient communications in the flat Supernet structure.

The remainder of the paper concentrates on the novel architecture to enable secure communications and its management, and properties and benefits derived therefrom.

4 Supernetworking

Our solution to the problems described in Section 1 is based on a design abstraction that we call *Supernetworking*, or *Supernets* for short. Instances of the abstraction are also called *Supernets*. Supernets are analogous to enterprise networks today, but augmented to provide the properties described in Sections 3.2 and 3.3.

4.1 The Model

On the communication side, Supernets are built through the introduction of a new layer of abstraction in a layered model of computer networking. The Supernet layer is located architecturally on top of the networking layer. It includes its own addressing structure and security services that protect all data that are transmitted by the network layer. In our current design, IP is encapsulated within the Supernet security layer (SNSL), which in turn is encapsulated in IP. This approach has repeatedly been used to implement infrastructure services such as multicasting, virtual private networking, and mobility; in short, any technology that relies on tunneling IP over IP (e.g., [TH98].)

The model is best described by a picture. The lower half (a) of Figure 2 depicts the model of internetworking as it is in practice today. The upper half (b) illustrates a virtual computer network, called a Supernet, that spans the internetwork depicted below. The internetwork provides the data transmission service for Supernets. The virtual network looks no different to processes running on computers participating in a Supernet than the internetwork does to computers today.

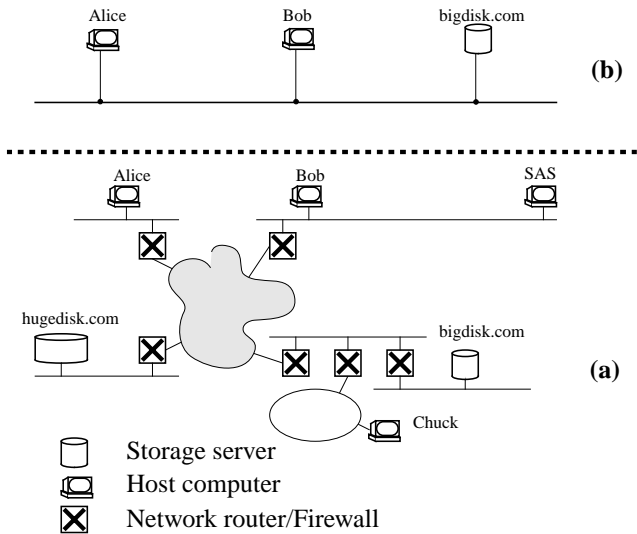


Figure 2. Network topologies: (a) Model of internetworking, (b) Model of Supernetworking

Each Supernet contains one or more communication *channels* and consists of a set of *nodes*. A *channel* is a communication abstraction that defines an association between Supernet members through a shared key. Communication packets that live within Supernets are always associated with exactly one channel. They are encapsulated in datagrams that can be interpreted and transmitted on the global IP network. This encapsulation provides – on a per group basis – authentication and confidentiality services. Because the knowledge of the key used is restricted to group members, encapsulated packets cannot be interpreted outside the Supernet channel to which they belong. For all practical purposes, a channel can be regarded as a local network.

In short, a Supernet aggregates nodes, placing them under a single administrative domain and giving them a network address space. Channels within a Supernet are used to segregate nodes and services within the Supernet.

A *node* is an entity identified by an IP address within a Supernet. Examples for nodes include processes, process trees, processes with the same user identity, or some other operating system dependent abstraction. A node can participate in multiple channels, but at most one Supernet. Nodes can communicate with other nodes only if they belong to the same channel on a Supernet, i.e., if they share a cryptographic communication key.

To support the concept of nodes, network addresses are therefore assigned at a finer granularity than one address per physical network interface. If TCP/IP is the basis for the addressing structure in a given Supernet, existing TCP/IP applications can be reused in Supernets. The concept of

Supernets does not preclude the use of other addressing schemes. The benefits of using a local name space for each Supernet are discussed in Section 6.

Each channel is protected from all other channels via cryptographic mechanisms. This allows the creation of multiple trust domains within Supernets. An example for the usefulness of multiple trust domains within a Supernet is given in Section 4.5. A channel appears to all nodes on the channel as a single shared medium (such as a classic CSMA-CD Ethernet). The channel abstraction does not need to include routing because end-to-end delivery, including routing, of data is provided by the underlying communication network. Therefore, on a Supernet level, every node is one hop away (i.e., directly connected) from every other node on the same channel. Only nodes that participate in a channel are addressable (and therefore reachable) by other nodes on the same channel. Because a Supernet can be comprised of nodes that are located in any host that is connected to the underlying communication network, Supernets afford a convenient abstraction on which to build shared networks whose members are not constrained to physical locations, i.e., a network in which an address does not indicate a physical point of attachment in the network infrastructure.

In the following sections we describe the components and the architecture that we used to implement this model, subject to the requirements described earlier.

4.2 Architectural Components

Supernets require five architectural components to ensure secure and robust communication among its nodes.

Supernet admission control service. First, the system provides authentication, admission control, and audit so that nodes become members of the Supernet under strict control that prevents unauthorized access. Both, authentication and admission control are governed by an explicitly expressed security policy.

Supernet security layer (SNSL). Second, the Supernet provides communication security services so that the sender of a message is authenticated. Communication between end points occurs in a secure manner by using encryption. These security services are achieved through datagram encapsulation and packet headers that carry the necessary data, such as message authentication codes or Supernet identifiers. Our architecture reuses the IPSEC (Internet Protocol SECURITY) standards for authenticity and integrity protection (authentication header, AH [KA98a]) and confidentiality services (encapsulating security payload, ESP [KA98b]), providing its own keying material to AH and ESP.

Groupkey management service. Third, the system provides a groupkey management service for the commu-

nication security layer. It is used to provide all nodes within one channel with the same SNSL traffic encryption and authentication keys. While unicast security is well-known and has widely advanced into production state, it can not be used to efficiently manage channel keys. Depending on the Supernet-specific policy, it may be necessary to provide all channel members with new keying material whenever a node joins, leaves, or is expelled from the channel. This can be done efficiently by group key management schemes tailored to scale well to large groups and to support multicasting communications, such as [CWSP98].

Address resolution service. Fourth, the system provides address translation in a transparent manner. Because the Supernet is a private network constructed on top of the public network infrastructure, the Supernet has its own internal addressing scheme. To send a packet over the public network, the Supernet performs address translation from the internal scheme to the addressing scheme of the underlying public network. System-level components of the Supernet perform this translation on behalf of the individual nodes in a fashion transparent to the nodes. If the IP addressing scheme is chosen for the Supernet-internal addressing, preexisting programs can run without modification within a Supernet ([LS00]).

OS-level enforcement of node compartmentalization. Fifth, the Supernet provides operating system-level enforcement of node compartmentalization in that an operating system-level component treats a Supernet node differently than it treats other processes running on that computer. The operating system recognizes that certain processes are executing as part of a Supernet node and enforces that all communications to and from them travel through the security infrastructure of the Supernet. The result is that a node can communicate only with nodes in the same channel of the Supernet, and that non-members of the channel cannot access a node. Additionally, this operating system-level enforcement of node compartmentalization allows more than one Supernet node to run on the same computer, regardless of whether the nodes are on the same Supernet.

4.3 System Architecture

Figure 3 illustrates how the architectural components interoperate. On the left side, Figure 3 depicts a typical computer that hosts one Supernet node. The right side shows a computer that runs the three infrastructure services that are necessary to make Supernets work. These services do not have to be run on the same computer and can be deployed

in a distributed fashion. In the other extreme, a supernet administration machine as illustrated on the right side can also host supernet nodes.

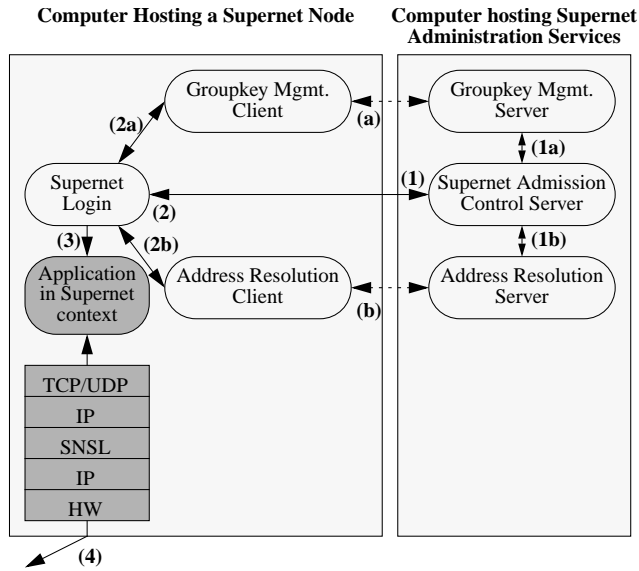


Figure 3. System architecture component interaction

A node is established through the action of *attaching to* a Supernet, i.e., a client process (Supernet login) is started that communicates with the remote Supernet admission control server (1). At this time, the client specifies its identity and the name of the Supernet it wishes to join. Upon successful authentication of the client, the Supernet admission control server performs an admission control decision to determine if the client is authorized to join the specified Supernet. If the client is admitted, a new virtual address valid only in the given Supernet network layer is negotiated for the node. The Supernet admission control server then informs the group key management server that a new group member has joined the supernet (1a). Furthermore, it informs the address resolution server of the mapping between the node's new *virtual* address and the IP address by which the node can be reached on the underlying transport network (1b).

The Supernet admission control server responds to the client with configuration data (2) that is necessary on the node's side to configure the group key management client (2a) and address resolution client (2b). From this point on, both the group key management and address resolution clients communicate with their respective servers asynchronously using the *Versakey Protocol* ([CWSP98]) (a) and the *Virtual Address Resolution Protocol* (VARP, [LS00]) (b).

Finally, all actions have been performed that were necessary to prepare the creation of a node. Subsequently, any application can be placed into the new Supernet context (3), with its own virtual network layer address and the key material to communicate securely to other Supernet members (4).

A node can leave its Supernet voluntarily or it can be administratively forced to leave its Supernet. In the former case, a client process (e.g., Supernet logout) authenticates itself to the Supernet admission control server similar to the way the client was authenticated on behalf of Supernet login. Upon successful authentication, the Supernet admission control server informs the address resolution service that the node mapping is no longer valid. Furthermore, it causes the group key management service to remove the node's key identifier and to start a key change for the channels the node used to belong to. This action allows every node in the channel, except for the newly-departed one, to learn the new key.

An administratively forced removal of a node works analogously, except that the node's removal is not initiated by the node, but by the Supernet admission control server or its operator.

4.4 Layering Architecture

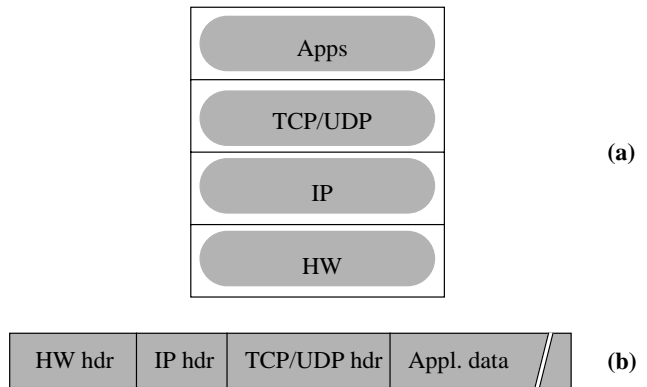


Figure 4. Traditional TCP/IP layering

In traditional computer systems, the networking stack is arranged as illustrated in Figure 4. Applications use their TCP/IP interface for all communications (Figure 4 (a)). Application data is encapsulated with a TCP header, which is encapsulated with an IP header. IP in turn is encapsulated in the appropriate hardware frame before it is sent towards its destination. Figure 4 (b) illustrates a hardware frame with its set of headers.

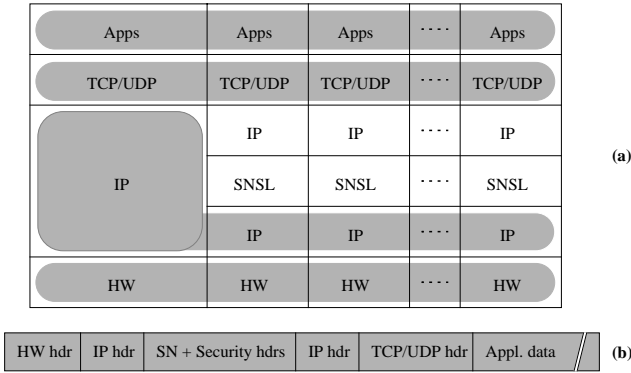


Figure 5. Layering architecture in a computer hosting Supernet nodes

In a Supernet context, two additional layers are introduced into the networking stack (see Figure 5 (a)):

- the inner IP layer which is used primarily for addressing within the scope of a Supernet, and
- the *Supernet security layer* (SNSL) which provides demultiplexing identifiers, such as Supernet and channel identifiers and security headers for encryption and authenticity protection.

Again, Figure 5 (b) illustrates a typical application datagram with its encapsulation headers. Figure 5 (a) also depicts the case where one computer hosts multiple nodes. Applications that are executing in the context of different stacks (i.e., which are executing in different nodes) are restricted to their Supernet channels. The underlying operating system enforces this encapsulation.

4.5 Example: Two Supernets

Figure 6 illustrates two Supernets and their underlying, preexisting internetwork infrastructure (Figure 6 (a)).

For example, Supernet 0x4711 in Figure 6 (b) consists of three nodes, a_1 , c_1 , and $bigdisk_1$. While nodes a_1 and c_1 participate in two channels each, channel 0x01 and channel 0x123, node $bigdisk_1$ participates only in channel 0x123. This configuration enables secure communications between nodes a_1 and c_1 without allowing node $bigdisk_1$ to be able to participate in their conversation, while at the same time allowing all nodes to communicate for storage purposes. This example illustrates how channels can be used to segregate secure communications at a fine-grained level.

The example in Figure 6 (c) for the Supernet 0x0000cafebabe00050100 consists of three channels, in which channels 0x02 and 0x03 are used for securing

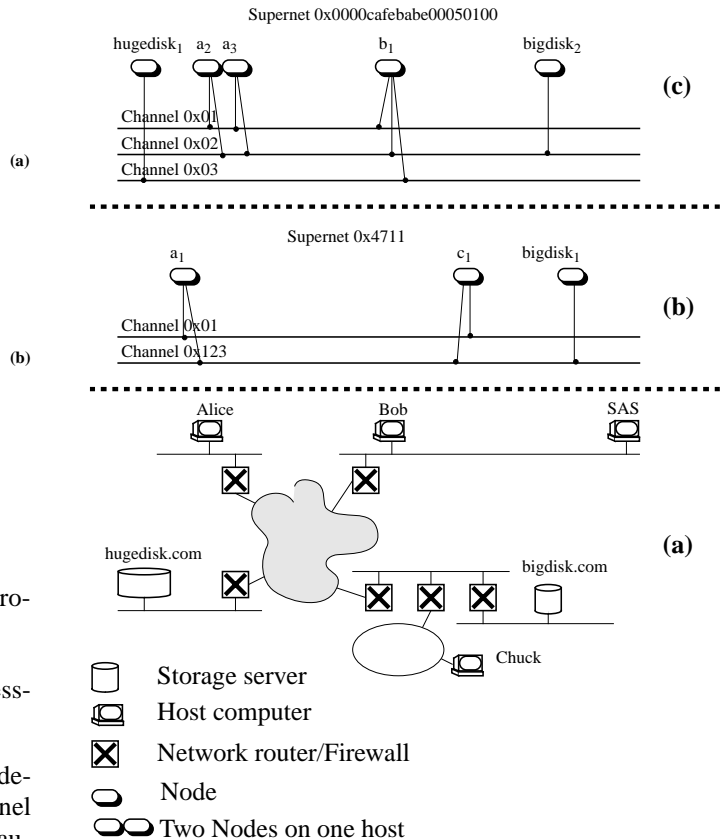


Figure 6. Example of two Supernets

network storage. Such a configuration could be chosen, for example, if network storage is out-sourced to different providers with no mutual trust.

4.6 Implementation

The system architecture presented in Section 4.3 has been implemented on a Linux platform. While the user-space components are generic and operating system independent, the kernel modifications are specific to the 2.2.13 version of the Linux kernel. In this section we give the salient features of our implementation and describe our experiences with it. We intend to include more detailed descriptions in future papers.

A node is implemented as an operating system process that is bound to a Supernet context. Each process's `task_struct` has a pointer to such a context. The context consists of a reference counted copy-on-write data structure that holds, among other things, the node's Supernet-id, its channel set, and its virtual IP address. The reference count is maintained across process creation and termination (i.e.,

`forks` and `exits`.) When the pointer to the context is non-zero, the process is said to be in Supernet context.

A node represents an end-point of communication and it can choose a channel on which to communicate. Whenever the process creates a socket within a Supernet context, this socket is bound to a specific channel. This is implemented by copying the triple (Supernet-id, channel-id, virtual IP address) from the caller's context into the `sock` structure of the newly created socket in function `inet_create()` in file `net/ipv4/af_inet.c`. By storing the triple in the socket and using it for demultiplexing datagrams, we virtualize the transport layer with respect to Supernets.

When a datagram is generated from such a socket, the (Supernet-id, channel-id) pair is copied from the `sock` to the `skbuff`, which is the data structure used to store outgoing datagrams in the Linux kernel. As the `skbuff` traverses down the IP stack, a transport layer header (TCP or UDP) is added followed by a network layer header (IP in our case). However, because this datagram refers to virtual addresses that are generally not routable on the physical network, it is encapsulated as the payload of another datagram that is routable on the physical network. This requires virtual addresses to be translated to physical addresses.

Address translation is performed within a kernel-loadable module. Before constructing the final datagram, the encapsulated datagram is encrypted in the kernel using keys in conjunction with another user-space program that manages keys per (Supernet-id, channel-id) pair. The resulting datagram is then routed to its physical destination using the *same* IP stack through which it traversed for the inner packet assembly. This is necessary in order to be independent of the mechanisms used within the stack for routing table lookups and interface delivery.

While most network traffic is generated in user space, there are some special cases where the kernel generates network traffic on behalf of a user process, or even independently. A good example for this are NFS operations, where the kernel translates disk reads and writes on the client side into remote procedure calls to a server computer. Two other examples are TCP packets being retransmitted, and ICMP replies to other computers.

To make NFS functionality accessible within the Supernet context, the `mount` command was modified to convey additional information to the kernel, and now whole filesystems can be placed within one or more Supernet contexts. The in-memory superblock of each filesystem carries information about which Supernets the filesystem belongs to. Whenever network traffic is generated for actions in the filesystem, the corresponding Supernet, channel, and node information are passed down the many FS and NET layers to where kernel-internal sockets are created and maintained. Filesystem access is limited by the same key-management constraints as processes are.

4.7 Experiences

Small pilot networks with several channels and nodes, and changing memberships have been run with stock applications such as telnet, Apache web server, etc., and custom test applications. Tools such as a modified `tcpdump` and other monitors have been created to observe and verify behavior of the running system.

When running applications from the confined network visibility of a Supernet context, we observed their dependency on services that we take for granted in today's distributed networking environments. Services, such as the domain name system (DNS) or network file system (NFS) are invoked on behalf of applications at times least expected. These services typically read their configuration from system-wide files that refer to addresses on the physical network rather than Supernet-specific ones. This means these services must either be replicated on a (Supernet-id, channel-id) pair basis, or be tunneled out of this context. Furthermore, configuration files must contain channel-relevant data and be visible only in their appropriate Supernet context.

Our solution has been oriented towards the latter, with the configuration files being replicated dynamically in the `/proc` file system dependent on the Supernet context of the caller. For example, in our implementation the DNS resolver configuration file `/etc/resolv.conf` is a symbolic link pointing to file `/proc/self/resolv.conf`, which will contain different data depending on the given processes Supernet context.

As an extension to the solely communication-based aspects of Supernets, we have also implemented and tested the mapping of NFS filesystems into Supernets, and realized a Supernet-specific filesystem which provides for client-based data encryption services, similar to CFS and TCFS.

Our experiences with all subsystems and the whole architecture have been very encouraging, and we plan to run larger-scale tests in the near future. After having more robust secure storage available we are now ready to place parts of our own work environment within a Supernet.

5 Related Work

Supernets combine many different ideas into one conceptual framework. In this section we compare some of these ideas as it is practiced today with its use within Supernets.

The X-Bone

The X-Bone ([TH98]) primarily concerns itself with building experimental virtual networks called "overlays." These overlays may be either protocol level or application level tunnelling. However, in the X-Bone, there are defined

routing sites which do not exist in the Supernet approach. In the Supernet, the virtual network appears to be one LAN segment with no routing.

Network Address Translation

The use of arbitrary IP addresses to number an internal network appeared when IP addresses directly routable on the Internet became scarce. The need for hosts in such private networks to communicate with hosts on the Internet led to the development of network address translation ([EF94]), or NAT for short. NAT typically uses a host different than the communicating source to rewrite packet addresses for traffic both going onto the Internet and returning to the private network.

NAT, however, suffers from several weaknesses. It breaks the end-to-end assumption implicit in many application-layer protocols. Furthermore, connections that originate outside a private network and are destined to the inside are difficult to manage. Triad ([DG00]) is an attempt to alleviate many of these issues. Supernets effectively perform NAT at each host with the additional benefit of end point addressability because virtual addresses are maintained and resolved transparently through an address resolution service. Supernets also maintain the end-to-end assumption because end points are directly addressable and routable at the application layer.

Plan 9

The use of per-process name spaces appeared in Plan 9 ([PPD⁺95]) which mapped traditional services into the file name space. These two features allow processes to map remote services into their local name space through the use of gateway processes on remote machines that acted as file servers executing the 9P protocol. However, there are several concepts pervasive in our design of Supernets which are not so easily mapped onto the Plan 9 design. One of those is the limitation of the visibility of resources to certain groups. This allows the partitioning of information and services. It is coarse enough to work in large, practical settings, yet provides fine-grained control where such coarse mechanisms are insufficient. Other crucial differences are outlined in Section 6.

Per-process name spaces have previously been implemented as shared libraries (e.g., union directories and versioning in n-DFS ([Kri95, Section 2.5].) We chose not to rely on the use of shared libraries because it may be undesirable to force processes to unconditionally link with a specified library.

6 Advantages Provided by the Supernet Abstraction

6.1 Independent Addressing

A key consequence of the Supernet abstraction is an independent addressing scheme for entities on the Supernet. This is an additional layer of indirection in naming and routing to real entities on the underlying communications network. This indirection allows several benefits:

Different Supernet entities can coexist on the same physical host. Supernets are the basic building block in our vision of out-sourcing. Because out-sourced resources such as computation and storage may be managed independently, the same physical host that provides more than one resource may be assigned to multiple Supernets. In such a case, decoupling physical addresses from Supernet addresses is very desirable because the same physical host may be assigned different addresses in each such Supernet.

Easier migration of networks. As described in Section 3.2, a key practical problem faced by network administrators today is that of address renumbering and address aggregation resulting from relocation. By decoupling Supernet addresses from the addresses used by the underlying communication infrastructure (for routing for example), Supernetworking eliminates this problem for corporate and ISP network managers. Renumbering can happen as often as needed at the underlying IP network, with minimal disruption of network services. This is made possible via automatic address mapping registration in the virtual address resolution service with flexible validity expiration periods.

Supernet topology and internal addresses can remain concealed. By decoupling Supernet addresses from the addresses on the underlying communication infrastructure, we can hide the Supernet addresses from all outsiders. Using the channel based security architecture access control to services on channels is trivially enforced through key management. Key revocation is enforced through a forced group key change as described in [CWSP98].

Logical proximity can be independent of the underlying network. Many service-location and multicasting schemes rely on local network addressing to provide access to nearby resources, or for scalability.

By decoupling these addresses, we can create Supernets where the addressing reflects this logical closeness, without forcing this on the underlying network.

Thus, a logically *nearby* printer may be the one provided by some external organization and it can be brought *close* to the address space in the Supernet, without requiring that the print service provider and the customer share an ISP.

6.2 The Value of Abstractions

The value and strength of our approach is to discern how simple and clean it is to realize our goals using the abstraction of Supernets, rather than from first principles. This abstraction forces a compartmentalization of functions and information between layers, and is yet compatible with traditional network-layer security solutions because it is possible to use them in addition to (i.e., on top of) our abstraction.

On the one hand, one might be tempted to treat this property as a system and software engineering problem, reasoning that a properly implemented system is robust. On the other hand, we argue that robustness is concomitant with simple design. Clean abstractions lead to simpler implementations; this leads to more easily maintained systems. Simpler solutions are generally easier to understand and reason with. When complex systems are built using them, it is sometimes easier to deduce system properties because they are based on simple unifying abstractions. By demonstrating that a design to which an implementation corresponds enforces an agreed upon approved abstraction, a convincing argument can be made that the system is correct.

7 Summary

This paper motivated and proposed a novel architecture that enables the out-sourcing of network infrastructure services in a secure and transparent manner. Communication, network storage, and computation services can be out-sourced to one or a set of providers. The fact that some or all infrastructure services are out-sourced is transparent to users, applications, and servers. Existing applications continue to work as they do on enterprise networks today. The technology enables the secure ad-hoc generation, maintenance, and dissolution of virtual enterprise networks.

Acknowledgements

Many thanks go to Tom Markson and Amit Gupta who were instrumental in the formative phases of Supernets, and to Sheueling Chang, Susan Landau, and Raphi Rom for many helpful discussions and contributions.

References

- [CWSP98] Germano Caronni, Marcel Waldvogel, Dan Sun, and Bernhard Plattner. Efficient Security for Large and Dynamic Multicast Groups. In *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 1998)*, June 1998.
- [DG00] Cheriton David and Mark Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture, 2000.
- [EF94] Kjeld Egevang and Paul Francis. The IP Network Address Translator (NAT). RFC 1631, May 1994.
- [FD99] Domenico Ferrari and Luca Delgrossi. Supranets, 1999.
- [KA98a] Stephen Kent and Randall Atkinson. *RFC-2402 IP Authentication Header*. Network Working Group, November 1998.
- [KA98b] Stephen Kent and Randall Atkinson. *RFC-2406 IP Encapsulating Security Payload (ESP)*. Network Working Group, November 1998.
- [Kri95] Balachander Krishnamurthy, editor. *Practical Reusable UNIX Software*. John Wiley, 1995.
- [LS00] Yuefeng Liu and Christoph Schuba. Virtual Address Resolution in Supernets: VARP. Technical report, Computer Science Department, Stanford University, August 2000.
- [PPD+95] Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom. Plan 9 from Bell Labs. *The USENIX Association, Computing Systems*, 8(3):221–254, Summer 1995.
- [Sch97] Christoph L. Schuba. *On the Modeling, Design, and Implementation of Firewall Technology*. PhD thesis, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, December 1997.
- [TH98] Joe Touch and Steve Hotz. The X-Bone. Third Global Internet Mini-Conference in conjunction with Globecom '98, 1998.