DESIGN OF A SELF-CONFIGURING TRANSPARENT SERVICE ENABLING SECURE LAYER 3 MERGING OF PRIVATE NETWORKS

Bas Boone, Jelle Nelis, Raf Hens, Jan Hollez, Filip De Turck, Frank Gielen
Ghent University - IBBT - IMEC
Department of Information Technology
Gaston Crommenlaan 8
9050 Gent, Belgium

ABSTRACT
Within a private network, users can use printers, share files and play games locally. It would be interesting if they could also use devices on remote private networks. However, when two users on different networks want to securely merge their networks, several problems arise. Technically they are in need of a site-to-site VPN between their networks, but most users don’t have the technical knowledge to set up this kind of configuration. Furthermore, they don’t want their networks to be reconfigured. In this paper these problems and their possible solutions are discussed. Using these solutions, a service running on the gateways is introduced that lets users merge their networks requiring a minimal effort. The gateways handle negotiation of parameters necessary to set up a connection and perform tasks such as address translation and routing to allow secure communication between the networks. Tests were performed comparing a direct connection with a connection set up by the system discussed in this paper. In a real-life situation the system does not cause a significant performance hit.

KEYWORDS
VPN, self-configuring, security, networking

1. INTRODUCTION
As an increasing number of consumers have their own private networks, it would be interesting to be able to share devices on their respective networks with each other.

An example of such networks can be seen in Figure 1. Both networks have an intelligent gateway as a single point of entrance. A number of different access technologies and devices can be used on both networks.

![Figure 1. Example of two private networks](image)

With the system presented in this paper, it is possible for a user to connect to a device on a remote network transparently without the hassle of manual configuration. No extra software has to be installed on his devices to make this connection and no reconfiguration of the network is necessary. Devices on a remote network behave indistinguishably from devices on a user’s own network. Accessing a server on a remote network is as easy as accessing one on the local network.
Since these networks are connected over the Internet, which is inherently insecure, security measures are taken. Traffic between connected networks is authenticated and confidential. Essentially, this constitutes a site-to-site VPN.

One of the difficulties that have to be dealt with is the fact most Internet Service Providers hand out IP addresses through DHCP, therefore addresses of gateways will differ over time. When making a connection, the current IP address of the remote device is needed. Without an extra abstraction layer, this is too difficult for users. Rather than letting the user figure out the correct IP address, the system presented in this paper lets the user identify the remote gateway using a human-readable name.

In order to merge the networks a connection between the two points of entrance is made. All traffic with a destination on the remote network is routed to the respective point of entrance. All other traffic is routed as normal. An important property of the system is that existing connections will not be broken when a connection between these networks is made. Since private IP addresses are not routable over the Internet, packets are encapsulated to be able to send them over the Internet. The routing and encapsulating process constructs a tunnel between these two gateways.

An addressing scheme has been designed to be able to distinguish between devices on different private networks using the same subnet.

Furthermore the system allows users to address local and remote devices using a human-readable name rather than an IP address.

2. RELATED WORK

A system called SoftWire enabling client-to-site SSL-VPN is discussed in (Enomoto, N. et al. 2005). It allows connecting a single client to an office network with a focus on ease of use and security.

In (Loeser, C. et al. 2003) a system is described that allows connecting multiple private networks to one virtual environment. In this environment, JXTA-enabled devices can communicate with each other using the JXTA peer-to-peer protocol.

In (Alchaal, L. et al. 2004) a system is presented to secure web services by dynamically setting up a VPN between the web service and its users. This constitutes a client-to-client VPN.

3. FUNCTIONAL DESCRIPTION OF THE DESIGNED SERVICE

3.1 Remote Deployment

To realize ease of use, a user must be able to easily and intuitively get the service installed on the gateway to his network.

To accomplish this, the software is remotely deployed by a Service Provider, which is displayed in Figure 2. Using a user interface provided by the Service Provider, the user requests a service. Subsequently the Service Provider pushes the necessary software on the gateway of the user. During this step the Service Provider can perform billing and other administrative tasks such as adding the newly registered user to a list of possible peers to connect to.

![Figure 2: Remote deployment](image-url)
3.2 Addressing and Routing

Since the subnets available for private networks are limited to subnets 10.0.0.0/8, 192.168.0.0/16 and 172.16.0.0/12 (Rekkhier, Y. et al, 1996), there is a possibility of address conflicts between clients on different private networks. Therefore, an addressing scheme is used that takes this into consideration. When two networks are connected that use different subnets, the IP addresses of the clients are used for addressing. When they use overlapping subnets, each device is given a virtual IP address that remote clients can use to address the device. For the clients themselves, this is transparent: their original network configuration remains intact. To facilitate looking up devices, they can be given a name.

In addition, the system handles routing of the IP traffic (with potentially virtual IP addresses) to the correct remote network (so that multiple private networks can be connected).

3.3 Configuration

When merging networks, several configuration parameters need to be known beforehand. Rather than letting the user figure out which situation he's in, resulting in manual configuration of several parameters, the gateways engage in a negotiation to automatically configure all needed parameters.

An example of such a parameter is the virtual subnet to use when addressing the remote network. This parameter depends on the local and remote network architecture.

3.4 Sample Service Usage Scenario

![Sequence diagram]

Figure 3. Sequence diagram which shows the interactions necessary to set up merging

In Figure 3 merging two private networks is displayed from the point of view of the user. If the user has requested the service before with the Service Provider, all necessary software will be installed on the local home gateway. Otherwise the user first needs to request the service.

To initiate a connection, the user visits a user interface on his gateway. On this interface, a list of all possible gateways to connect to is displayed, which the gateway gets from the Service Provider. When the user selects one of these gateways, the gateway will start to negotiate with the remote gateway upon receipt of this request.

The local gateway will send a negotiation request containing all necessary information to the remote gateway. The remote gateway will now decide which configuration parameters to use and send his decision to the local gateway. As a last step the local gateway will approve or reject this decision. If the negotiation succeeds, the networks will be merged.
Shutting down a connection can be done in completely the same way. Because the Service Provider knows when a connection is opened and closed, billing can be done per connection or per time unit.

4. DESIGN DETAILS

4.1 Addressing and Routing

4.1.1 Addressing Devices

There are two possibilities when trying to connect private networks.

Firstly, it's possible both private networks use distinct IP subnets. In this case there's no problem in terms of addressing devices. Users on one side of the connection can address devices on the other side of the connection by simply using the original IP address of the device.

Secondly, two networks can use overlapping IP subnets. This means devices can't be addressed unambiguously. Extra measures need to be taken to overcome this problem.

To solve this problem, virtual IP addresses are used. Both sides of the connection choose an IP subnet not conflicting with any other used subnets and use that subnet to address devices on the other side of the connection. Note that these subnets can only be (part of) the private subnets mentioned before to prevent conflicts with publicly routable IP addresses. In order to achieve this, an extra step of address translation is needed when packets are entering or leaving any of the networks.

Another solution to this problem would be to use layer 2 merging and to give each device a new IP address in a common subnet. However, this would mean that the system is no longer transparent for the clients: their network configuration has to be changed and established connections would be broken.

4.1.2 Address Translation

![Address translation diagram]

Figure 4: Address translation between two private networks. Network A uses subnet 192.168.20.0/24 as virtual subnet. Network B uses subnet 192.168.10.0/24 as virtual subnet.

Once every device can be uniquely addressed, packets still need to be able to reach their destination. When a client wants to send a packet to a device on the remote network, he sends it to the device's virtual IP address.

In Figure 4 an example of such a packet is displayed. Network A and network B both use subnet 192.168.0.0/24. Network A uses the subnet 192.168.2.0/24 for its virtual addressing, network B uses...
192.168.1.0/24. All devices keep their original IP addresses but are also addressable by their virtual IP addresses. In the solution presented, the gateway is only aware of the virtual IP addresses of his own private network.

When a client on network A (192.168.0.2) wants to send a packet to a client on network B it has to address it using its virtual IP address (192.168.1.4). When the packet reaches the gateway of network A the source address will be translated to the virtual address of the local client (192.168.2.2) for return addressing. The packet then gets sent over the Internet reaching the remote gateway which translates the virtual destination address (192.168.1.4) to the real IP address of the remote client (192.168.0.4).

Using this scheme no reconfiguration of the client devices on the private networks is needed since the gateways handle all necessary address translation.

4.1.3 Routing

Packets sent between clients on different private networks are not routable over the Internet.

To solve this, it is necessary to tunnel the packets between both gateways. Before routing an IP packet, it is encapsulated within a new IP packet. The source address of this packet will be the WAN IP of the local gateway whereas the destination address will be the WAN IP of the remote gateway. When the remote gateway receives the packet, it will unpack the original IP packet and send it on the remote network.

Traffic for the remote network will be routed towards the remote gateway using the tunnel and all other traffic will be routed as before.

4.1.4 Security

A tunnel is now made between the local and remote gateways. The packets sent over this tunnel are still in plain text; to assure confidentiality and authentication, an existing VPN solution will be used to set up the tunnel. The VPN solution will encapsulate and encrypt packets sent over the tunnel.

4.2 Naming Service Details

Most gateways will receive their IP addresses from ISPs through DHCP, which means that a static IP address cannot be guaranteed. Therefore gateways can register their IP addresses and names with the Service Provider, who provides a lookup service for them. This way, users can use human-readable names to address the different gateways.

A second naming service is provided to look up individual client machines. This is highly needed because client machines can be given a virtual IP address. Users can register a name for their client device after which other devices can find out its IP address using standard DNS.

4.3 Module Description

Figure 5 shows the different modules composing the service. The Negotiator module is a generic parameter negotiation module. Before a VPN connection is made, the Negotiator modules on both gateways will negotiate connection parameters over a secure connection. If negotiation fails because no compatible parameters can be found, the system will not attempt to create a VPN connection.

The AddressTranslation module is responsible for configuring address translation on the gateway. During negotiation, it decides if address translation is necessary (based on the subnets used by both networks) and if so, which virtual subnets to use.

The VPNInterface module is an interface to the underlying VPN technology. It is responsible for starting and controlling a VPN connection. It also configures routing after a VPN connection is set up. By placing this functionality in an abstract module, it is easier to change the actual VPN technology used.

The Gateway Naming System (GNS) module is responsible for registering the gateway name and its IP address with the GNS server. Communication between the GNS and the GNS server is secure (authenticated and confidential).

The GNSServer module keeps a database of (name, IP address)-pairs, which is updated by the GNS modules. This database can be queried using DNS. These mappings in the database are used by gateways to look up remote gateways prior to making a VPN connection to them. The GNSServer module is deployed on a server managed by the Service Provider.
The Client Naming System (CNS) has a similar function as the GNS and the GNSServer modules: it records name, IP address-pairs for clients’ names. These IP addresses can be virtual addresses if address translation is used. Users can register a name for their device by using the web interface. The CNS also acts as a DNS server for the clients on the local network. DNS requests for client names from the local network are resolved by the CNS; requests for client names on the remote network are forwarded to the correct remote gateway’s CNS and all other requests are forwarded to a default DNS server.

The web interface module provides a web interface for the users, enabling them to configure the system and to start or stop VPN connections.

![Diagram of modules and connections of the system](image)

Figure 5. Modules and connections of the system

5. IMPLEMENTATION DETAILS

A proof of concept system was developed, using the architecture described in section 4. The most important design decisions for this system are outlined in this section.

OSGi (The OSGi Alliance, 2003) was used as a platform for the gateways. This platform was chosen because of its suitability for intelligent gateways and because of the ease of deployment it provides for Service Providers.

The VPN technology used was OpenVPN (James Yonan, n.d.). This open source VPN implementation was chosen because of its availability for most platforms and because it is, as a self-contained userspace solution, possible to deploy remotely on many different platforms.

Address translation was only implemented for GNU/Linux platforms. The system can still be used on other platforms, if no overlapping subnets are used (and thus no address translation is necessary).

On GNU/Linux, the implementation makes use of `iptables` (The netfilter.org "iptables" project, n.d.) `NETMAP` target to handle remapping subnets to different subnets.

The GNS server was deployed on a GNU/Linux machine running the BIND (Internet Systems Consortium, n.d.) DNS server. A small program received updates from the gateways’ GNS modules, which were then made available to BIND by using the `nsupdate` tool.

From a user perspective, installing the system on a gateway (which runs OSGi) is as easy as surfing to a website and clicking on a link. The system is then automatically installed on the gateway.

To use the system, the user surfs to the URL of the web interface of the system and inputs the name of the gateway he wishes to connect to. A site-to-site VPN is then set up between both networks.
6. VALIDATION AND RESULTS

To validate the design, performance tests were performed on the proof of concept system.

6.1 Set-up

In the test set-up, two gateways and a GNS server were connected through a 100 Mbps switch. A client machine was connected to each gateway. Both gateways and client1 are computers with a Pentium III 500 MHz processor and 128 MB RAM running GNU/Linux (kernel version 2.6.8). Client2 is a laptop with an Intel Celeron 1066 processor and 256 MB RAM running GNU/Linux (kernel version 2.6.15). The GNS server has an AMD Athlon 750 MHz processor and 256 MB RAM and runs GNU/Linux (kernel version 2.6.10) and BIND 9.2.4. The OpenVPN version used on all computers was 2.0 and all tests used Blowfish encryption in cipher block chaining mode.

Performance tests were conducted for 3 situations: without using the system (clients can connect to each other directly), using the system without address translation and using the system with address translation. Throughput and delay were measured.

6.2 Throughput

In this test, a large file was transferred from client1 to client2 using the standard HTTP protocol. The clients used a maximum transmission unit of 1500 bytes. The throughput on the gateways was then measured with tcpdump and averaged over 60 seconds. The results of this test are shown in the first column of Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Throughput 100 Mbps</th>
<th>Throughput 512 kbps</th>
<th>RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct connection</td>
<td>58.521 Mbps</td>
<td>472.376 kbps</td>
<td>0.3 ms</td>
</tr>
<tr>
<td>Tunneled</td>
<td>19.305 Mbps</td>
<td>457.288 kbps</td>
<td>0.8 ms</td>
</tr>
<tr>
<td>Tunneled with address translation</td>
<td>20.735 Mbps</td>
<td>456.560 kbps</td>
<td>0.8 ms</td>
</tr>
</tbody>
</table>

A significant reduction of throughput is introduced when using the system, with or without address translation. The bottleneck in this situation is the processing power on the gateways: they are not able to encrypt and decrypt packets at a high enough rate.

However, since a symmetric 100 Mbps bandwidth is not typical for average Internet access connections, tests were also performed with a bandwidth limited to 512 kbps. The results for this set-up are shown in the second column of Table 1.

In this situation, throughput is only reduced by 3%. This reduction is caused by the extra headers sent over the network that are needed for the tunneling protocol. However, no packet fragmentation occurs since OpenVPN adapts the TCP maximum segment size (MSS) option in the TCP handshake to account for the extra OpenVPN headers. In this test, this meant a reduction in MSS from 1460 bytes to 1369 bytes.
6.3 Delay

The delay between the two clients was measured using the tool `ping`, which measures round trip time (RTT) between two devices using ICMP echo-request and reply messages. The results of these measurements, averaged over 20 'pings', are shown in Table 1.

The results show that the round trip time when using the system is 0.5 ms higher. For typical Internet connections, where round trip times will usually be several milliseconds, this will have a lower impact than in this test set-up.

7. FUTURE WORK

In the current design, users can register a name for their device with the system that other users can use to look up the device and make use of the services on it. Since the system knows all registered devices, it can present the user, on the web interface, a list of registered devices. This could be extended by also including the services provided by the devices. To achieve this, a service discovery protocol, such as Service Location Protocol (SLP), can be used (Guttmann, E. et al., 1999).

8. CONCLUSION

The system presented in this paper provides a transparent, easy to use solution to merge private networks. It enables people with little to no technical knowledge to share a private network securely. Configuration is done nearly automatically, installation of the system is handled by the Service Provider and starting a merge requires only the input of a gateway name. The system is completely transparent for devices on the participating networks: no reconfiguration of devices is necessary and no additional software has to be installed on the client devices. As a consequence communication of other users on the private network is unaffected when two networks are merged.

The system is limited to IP based protocols. Client name lookups are realized using DNS, therefore only applications using it can make use of this feature. Software that uses other means of name lookups (such as Windows file sharing) will not always find the remote devices.

Service Providers can offer the system as an extra service to customers. Billing can be done per installation, or per use of the system.

REFERENCES


APPLIED COMPUTING
February 18-20
Salamanca, Spain
Proceedings
Edited by:
Nuno Guimarães
Pedro Isaias
iadis international association for development of the information society