WinGate's Winsock Redirection Protocol (WRP)
A white paper comparing WRP with SOCKS and Network Address Translation (NAT)

Introduction

The purpose of this document is to provide insight and understanding for the need to introduce WinGate's Winsock Redirector Protocol (WRP) as a means of IP Multiplexing. IP Multiplexing for the purpose of our discussion, is the sharing of a single Internet connection and IP address to provide Internet connectivity for an entire network. WinGate provides this basic functionality. WinGate and its competitors are collectively known as "proxy server / firewall products".

We recognize that the current protocols and methodology used by today's proxy server vendors provide a significant amount of benefit to LAN based Internet users. Work is still required to enable these solutions to be ready for prime time. Our research of existing protocols led to the fact that there was a need to extend or create a new protocol. This protocol will provide the additional functionality required, allowing WinGate to be pervasive. Further research indicates that no current standard protocol will afford an elegant way to extend to the level of functionality needed.

The WinGate development team (led by Adrien de Croy) developed WRP to allow WinGate to extend into the Home Networking / Internet Sharing market, while providing additional benefit for all WinGate users. WRP combines the best features of existing protocols, while overcoming their limitations.

For all versions of WinGate prior to Version 3, WinGate used the application level proxy approach to Internet sharing. This is a server-based solution requiring users to possess at least a basic understanding of networking technologies. In addition, application level proxies, as well as other current competing solutions share three basic (less than desirable) tendencies. Our observations of WinGate and WinGate's competitors provided the motivation required to overcome these deficiencies. The three main objectives we needed to overcome were:

1. Complexity - Most current proxy server offerings are complicated to install and configure. This limits their ability to penetrate the mass markets, and in particular the consumer market.

2. Application Limitations - Most current proxy server offerings have a substantial number of Internet application limitations.

3. Compatibility - Some current proxy server offerings (based upon a proprietary Winsock.DLL) do not provide complete compatibility with today's Internet applications, and have raised a list of other compatibility issues.

With this document we will explore how WRP provides a solution to these problems.

IP Multiplexing Methods

There are two basic methods of IP multiplexing in use today; the proxy server method and Network Address Translation (NAT) method. The proxy server method requires the configuration of client machines. The user can manually configure the client machine, or use client software to meet the configuration requirements. The proxy server method provides a higher level of functionality, as it maintains a relationship between the client machine and the proxy server. The NAT method eliminates the need for configurations at the client level. The server machine

actually alters the data packets changing the originating IP address. The client machine does not realize that the packets filter through a server before reaching the host on the Internet. By working at the packet level, the NAT method provides greater performance. While performance is increased, functionality is sacrificed due to a lack of higher level communication between the client and NAT server.

The outside world views the multiplexed IP address as a single machine and single IP address. For connections that originate from within the firewall, this is not normally a problem. The application receiving the connection does not need the originating IP address. However, many socket based applications use more than one connection and often these additional connections originate from outside the firewall. The firewall server must determine which internal machine is expecting a connection from the outside world. The problem compounds with server network applications that listen for only incoming connections. Users behind a firewall may be running server applications that wait for incoming connections.

The File Transfer Protocol: A Case Study

To illustrate the problem it is helpful to consider one of the more popular socket based applications in use today. All known applications that use the File Transfer Protocol (FTP) are socket-based applications. FTP is a good case study for firewalls because it uses two connections, one for FTP commands, and the other for file and directory-listing data. The first connection, called the *control* connection, originates from the FTP application. The connection will complete on an external computer on the Internet. Firewalls can easily handle the control connection, because it is an outgoing connection on a known port. The second connection, called the *data* connection, originates from an outside FTP server. The server will then try to connect back to the machine requesting the data. The data connection is difficult for firewalls to handle because the firewall server must determine which internal machine originated the FTP session.

Consider the following comparison. A common firewall protocol method is SOCKS. Another method is IP Network Address Translation (NAT). How does each handle the FTP issue? The SOCKS method is straightforward: each client machine (behind the firewall) has special software installed that communicates with the firewall server. The client machine tells the firewall to expect an incoming connection on a specific port number. The firewall recognizes the incoming connection, and passes it to the originating internal machine. The NAT method uses an option available with all FTP software. FTP has a special mode called passive mode that forces FTP to originate both connections from the client machine.

NAT Design, Features, and Limitations

You can read all about the low-level details of NAT in RFC 1631 “The IP Network Address Translator (NAT)”(1994). For the purpose of this discussion, a NAT firewall hides multiple *private* IP addresses behind a smaller number of *global* (public) IP addresses. A global IP address is a unique number, which refers to only one resource on the Internet, at any given time. These global addresses are allocated by The Internet Assigned Numbers Authority (IANA). The IANA also allocates a number of private IP addresses for use on a LAN that is not directly connected to the Internet. Most people use the same set of private IP addresses for their LAN preventing direct connection to the global Internet.

IP Masquerading is one common form of the NAT protocol, which can hide a number of private IP addresses behind a single global address. This feature will allow for a large number of private IP addresses on a LAN, to connect to the Internet using one global IP address. The internal requests appear to the Internet as originating from the machine assigned the global IP address.

One advantage of NAT is that the client machines require no additional software to connect to the firewall. The conversion takes place entirely on the NAT gateway. The NAT module works at the IP packet level and simply changes the source address and port to that of the NAT machine. The client machines assume they are sending and receiving via a direct connection to the Internet. The software running on the client has no way of determining its IP address as viewed by the outside world. It creates a problem for any high-level protocol that transmits an IP address, as part of its protocol. FTP, as shown earlier, is a good example. Other examples include any protocol used for audio and video conferencing.

Other limitations of the NAT protocol include; any software that accepts inbound connections on a predefined port number can not run behind a NAT gateway. The NAT machine will reject incoming connections from the Internet because the internal machine has no way of telling the NAT gateway to listen for incoming connections. NAT does not handle relaying of incoming User Datagram Protocol (UDP) messages to the appropriate internal machine. Unlike TCP, UDP is a connectionless protocol. A NAT server has no default way of determining which port to route the incoming packets.

Proponents of the NAT solution have presented the idea of a more intelligent NAT gateway. The new gateway will guess where to route incoming connections based on knowledge obtained from previous connection activity. For example, if internal machine A connects to global IP address X, the NAT software would record that connection. Later an incoming connection attempt from the same IP address X would be assumed for the original machine A. Though these heuristic rules work well for a simple protocol like FTP, they are not designed for more difficult tasks such as communication to an internal server. Moreover, where more than one client machine connect to the same server, there is an immediate problem – how can the NAT software tell which client to route the incoming connection to?

SOCKS v5 Design, Features, and Limitations

During the design phase of WinGate 3.0, we decided to design a client/server firewall protocol solution. The benefits of having the internal machine aware of the firewall eliminate the limitations of the NAT protocol. Most socket-based applications have internal support to cross SOCKS aware firewalls. The emergence of Winsock2 for Windows, and its support for Layered Service Providers (LSPs), provided a straightforward way to add SOCKS support for all Winsock based applications. WinGate 2.x has support for the latest SOCKS version 5. Early versions of the LSP filtered all Winsock calls and modified them for use with the SOCKS protocol.

As we wrote the LSP for SOCKS v5, we started hitting some serious brick walls with the protocol. For TCP connections, SOCKS control information transmits via the actual data connection. There is simply no way to transmit useful information after the initial connection. Extending the protocol was not an option because of its design. WinGate is a robust software solution, requiring status information throughout a session. Another problem is that you can not accept more than one connection from a listening socket, thus eliminating any possibilities of an internal server. Other problems with the protocol include; no way to set server socket options and incompatible error codes.

Though we were able to get a prototype SOCKS v5 enabled LSP working with the WinGate SOCKS server, the use of SOCKS was quickly abandoned for WinGate 3.0 because we found no reasonable way to extend SOCKS for our requirements.

WRP Design, Features, and Limitations

After thorough research, investigation and software prototyping, we discarded the existing methods of IP multiplexing in favor of a completely new protocol. The new protocol, named the Winsock Redirection Protocol (WRP), is our solution. It redirects Winsock calls from the client machine, to the machine running the WinGate firewall server. Any Winsock based application running behind the firewall on a private IP address is tricked to believe that it is actually running on the firewall machine which has a global IP address. The LSP intercepts all Winsock calls and returns proper IP addresses to handle applications such as FTP. WRP can handle the redirection of *bind*, *listen* and *accept* requests, providing support for internal servers.

WRP uses a separate control connection for each Winsock application running on the machines behind the WinGate firewall. Control information flows throughout the entire session. Authentication occurs only once per client machine during the control session, rather than doing all authentication over each data connection.

WRP handles the complete set of Winsock 1.1 calls. Winsock error codes from the firewall's WRP server return to the Winsock application to provide meaningful error codes to the user. The new protocol is also highly extendable and consistent with Winsock functionality. Currently, WRP queries the machine name, logged in user name, and application name.

All outgoing connections and relaying are specified on a per connection basis, so WRP can load share via central administration. Additionally, the server can tell the client whether to redirect or not. With this feature, local traffic connects directly bypassing the server. The client does not need any configuration and is not even aware it is redirecting. In fact, a WRP client normally requires zero configurations, whereas clients using other protocols (e.g. SOCKS) require the user to configure sub-nets for redirection. The additional sub-net configuration requirements of other protocols limit their usefulness when dealing with the consumer market.

The Future of WRP

The WRP protocol is connection-based at the TCP level, which may cause it to suffer in performance when compared with NAT solutions. With today's relatively slow Internet connections, this performance penalty may not yet be a factor. Performance enhancement options are in the planning stages for a future release. A NAT/WRP hybrid combines the performance of NAT with the superior functionality of WRP. The WRP server would setup the initial connection to the outside world and let a NAT packet driver take over during the application's main data transfer. WRP control connections would remain at the TCP level, but the data connection would drop down to the IP level.

Another enhancement under development for WRP involves redirecting driver level TCP calls for Windows device drivers that do not use Winsock. One such set of drivers is Microsoft's Windows Networking. Using WRP, network redirectors traverse a WRP enabled firewall. In addition, this interface will allow us to support core protocols such as PPTP and ICMP.

The Winsock Redirection Protocol is the answer for the home networking/Internet sharing consumer. With an intuitive interface, WRP is easy to install and requires little to no configuration. WRP is compatible with current and future Internet applications and is user transparent.