159159

GENERIC EXTERNAL PROXY

Field of the Invention

The invention generally relates to networking, and, more particularly, to supporting usage of communication protocols that embed non-accessible network configuration data within network traffic.

G00002

Background

Recent years has given significant advances in networking technology and reduced pricing, resulting in a significant buildup of networking infrastructure. Most businesses and households are interconnected through private and public networks, the most well known network being the Internet. Most networks now utilize the Transmission Control Protocol / Internet Protocol (TCP/IP) communication protocol, in which network locations are assigned a globally unique 32-bit numeric address typically presented in dot quad notation (four numbers each having values of zero to 255). TCP/IP network traffic is routed based on a destination IP address for the traffic.

Unfortunately, the explosive growth of the Internet has resulted in a shortage of available network addresses. To compensate, attempts have been made to share a single network address among multiple computers. One well-known example is Network Address Translation (NAT), which hides an internal network behind an access point in communication with an external network by routing network traffic through the access point. Since the internal network uses private network addresses the packets from this network are not routable in the Internet without translation. During operation, NAT modifies source IP address and ports of outgoing network traffic to map the traffic to an external or public

address and a unique NAT port. NAT also modifies destination IP address and port of incoming network traffic using the mapping of external address and unique NAT port back to the original internal address and port. NAT ignores network traffic not received in response to original outgoing network traffic, and incoming traffic to unmapped ports.

Network traffic translation performed by a translating access point such as a NAT gateway/router **102**, firewall **108**, or the like, is transparent to many applications. However, translations break protocols under certain circumstances, such as with audiovisual conferencing (e.g., International Telecommunication Union (ITU) standard H.323), IP Security (IPSec), end-to-end security models that cannot allow packet header alterations, and protocols that embed a machine's network address and/or communication port values as application data within network traffic, such as the File Transfer Protocol (FTP), multi-player network game protocols, etc.

For example, in FIG. 1, an H.323 client **110** inspects its network configuration and sends it to an H.323 gateway **118** as application data. Because H.323 client **110** is in a private network, the configuration indicated in the application data cannot be used by H.323 gateway **118** to access it from the Internet. That is, a translating access point modifies packet header data not application data. Therefore, the protocol fails because the protocol effectively reports the wrong information within the application data.

One proposed solution to this problem is the REALM specific IP (RSIP) protocol, an Internet Engineering Task Force (IETF) suggested revision to NAT. Assuming the International Organization for Standardization Open Systems Interconnection (ISO/OSI) model, networking protocol layers 3 and 4 are altered

to support RSIP in every translating access point. An RSIP access point grants a client, e.g., a machine in network **100**, resources (e.g., address, ports) in an external realm, e.g., network **104**.

Unfortunately, RSIP (and related solutions) are expensive and impractical. To work properly, all translating access points have to be revised to support RSIP; this solution fails if an upstream non-supporting translating access point is reached.

Brief Description Of The Drawings

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

FIG. 1 illustrates a prior art network configuration of computing devices interconnected through the Internet.

FIG. 2 is a generalized diagram, according to one embodiment, for supporting NAT with protocols that embed network configuration data in a NAT-inaccessible network traffic payload.

FIG. 3 is a flowchart illustrating, according to the FIG. 2 embodiment, communication between an application program and an endpoint that travels through a translating access point.

FIG. 4 illustrates one technique for implementing an embodiment of the FIG. 3 querying a server for an external address/port.

FIG. 5 is a diagram illustrating a specific application of the embodiment of FIG. 4 to an H.323 telecommunications application program.

FIG. 6 illustrates a suitable computing environment in which certain aspects of the invention may be implemented.

Detailed Description

FIG. 2 is a generalized dataflow diagram, according to one embodiment, for supporting NAT with protocols that embed network configuration data in a NAT-inaccessible network traffic payload. This general overview is presented in more detail in the following figures.

As illustrated, a networking application program **200** is in communication with network services **202** provided by an operating system, e.g., a software and/or hardware based operating system providing services to the application program **200**. During operation, a typical network application program requests the operating system to provide network configuration data the application program may use. In a TCP/IP environment, such a request typically comprises asking the operating system to identify the network address for the network interface (e.g., FIG. 6 item **618**) address of a host system executing the application program, and an available communication port. If the host system is in a private network, this will be a non-routable network address that cannot be used in the Internet.

All network traffic from the application program **200** is assumed to pass through a translating access point **206** (translator), such as a NAT device or equivalent, before reaching network **208**. Since translators do not alter application data, protocols that send network configuration data within application data fail to work from a private network. Consequently, to resolve the application program request for configuration data, a configuration determinator **204** component of the network services **202** contacts an external network configuration server **210**. The server provides the determinator with network configuration data that can be used

in a public network such as the Internet or equivalent network. For example, it may provide a network address and communication port of the server.

When the determinator **204** receives this configuration data from the server **210**, the network services **202** reports this data back to the requesting application program. When the application program embeds the configuration data within application data, a responsive network device (e.g., FIG. 1 H.323 client **116**) issues responses that are sent in accord with the data provided by server **210**. For example, assuming the server provided a network address and communication port of the server, the responsive network device sends responses to the server; the server then forwards the response for receipt and usage by the application program **200**.

FIG. 3 is a flowchart illustrating, according to the FIG. 2 embodiment, communication between an application program and an endpoint that travels through a translating access point such as a NAT gateway/router (translator).

An application program utilizes a protocol to communicate with the endpoint. It is assumed the protocol is one that breaks because a private network address is included in application data, however the protocol may be a simple network connection. The application program requests **300** operating system network services to identify the network address of the application program's host, and to obtain an available (or specific) communication port. Typically, this request asks the operating system to identify the host's IP address and an available UDP or TCP port.

The operating system in turn calls **302** appropriate installed network services (provided in software and/or hardware) to resolve this request **300**.

Assuming the application program is not aware of network traffic translation by an access point, in one embodiment, network services are configured to forward the call **302** to a proxy client. One exemplary implementation for forwarding to the proxy client, discussed below with respect to FIG. 4, is a Layered Service Provider (LSP) within Microsoft Windows network services. (Please note that all marks used herein are the property of their respective owners.) However, it will be appreciated by one skilled in the art that other forwarding mechanisms may be used.

The proxy client in turn requests a network address and communication port from an external proxy server. As used in the specification and claims that follow, an external proxy server comprises a server with a network connection, e.g., an Internet connection, not subject to translation by an access point. For example, device **516** of FIG. 5 has a direct non-translated connection to the Internet. This request passes through **306** a translating access point. The external proxy server replies **308** to the proxy client with an available address and communication port. This reply also passes through the translating access point. However, since the external proxy server has a non-translated network connection, the proxy client reports to the requesting **300** application program an address and port that is external to and not affected by translation by an access point.

In one embodiment, the proxy client establishes a tunnel with the external proxy server . The external proxy server connects **310** to the endpoint and establishes a communication session with the endpoint in accordance with the protocol utilized by the application program. The tunnel can be used to pass **314**

network traffic received at the address and port given **308** to the proxy client, and to carry network traffic from the application program to the endpoint.

The tunneling is transparent to both the application program and the endpoint because the application program believes the networking configuration values from the external server corresponds to values for the application program's host, and because the endpoint can directly communicate with the proxy server using the networking configuration values. Thus, based on such tunneling, a virtual direct link **316** unaffected by translating access points is created between the application program and endpoint.

FIG. 4 illustrates one technique for implementing an embodiment of the FIG. 3 querying **304** a server for an external address/port.

As illustrated, a Layered Service Provider (LSP) **404** is implemented in conjunction with the Microsoft Winsock Application Programming Interface (API) **406**. In the illustrated embodiment, a Microsoft Winsock API is extended with the LSP, which configures Winsock to pass on network calls on to the LSP **404** for primary processing. For more information on LSPs, please see "Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider" by Hua et al., Microsoft Systems Journal (May 1999); Internet URL: msdn-microsoft-com/library/periodic/period99/layeredService-htm. (To prevent inadvertent hyperlinks, URL periods have been replaced with dashes).

In the illustrated embodiment, an application program **400** on a computing device (such as FIG. 1 item **110**) is executed, where the application program utilizes a certain protocol to communicate with a peer (such as FIG. 1 item **116**) that normally breaks when the protocol is used from behind a translating access

point. For example, assume the application program identifies its host's network configuration, and sends the configuration through a translating access point **406** as application data sent to another computing device **116**. The application program embeds configuration values that are not accessible over the network **104** by the peer device **116**. Consequently, network traffic from the peer **116** directed to the embedded configuration data cannot be received by the application program **400**.

In the illustrated embodiment, when the application program **400** calls on operating system network services to identify its host's network configuration, the operating system calls Winsock **402** services, which in turn calls the LSP **404**. When control is passed to the LSP, the LSP obtains a network configuration not subject to translation, from an external proxy server **410**. The LSP provides the configuration to Winsock, which in turn provides it to the application program. The application program may safely embed the LSP provided configuration in application data.

In one embodiment, when control is initially passed to the LSP, the LSP initially determines whether to accept the call, or to simply pass it back to the Winsock API to let it handle the call. This allows application programs that are "aware" of the translating access point **406** to operate without intervention. For such applications, the call to the LSP is passed back to Winsock for regular Winsock processing. However, if the application program is unaware of translation, the LSP calls an external proxy server **410** which returns to the LSP a network address and communication port not subject to translation. A tunnel is established between the LSP and the proxy server so that communication by

other computing devices to the provided address/port can be transparently tunneled to the application program.

It will be appreciated by one skilled in the art that the invention is not operating system dependent; use of the Microsoft networking environment, and LSPs is for exemplary purposes due to broad familiarity with these environments. Other operating systems and network services may also be utilized. Also, some environments may utilize closed network services, e.g., an LSP type of construction is not available. In such environments, in one embodiment, calls to network services are intercepted and processed by a proxy client executing on the application program host. In an alternate embodiment, a virtual network interface and software router are used to receive network traffic which is routed through a physical network interface.

FIG. 5 is a diagram illustrating a specific application of the embodiment of FIG. 4 to an H.323 telecommunications application program. It will be apparent to one skilled in the art that the illustrated embodiment is applicable to any protocol that inspects its host's network configuration and embeds this data in application data.

As illustrated, an exemplary computing device **500** comprises an H.323 application program **502**, LSP network layer **504** (see also FIG. 4 item **404**), Winsock (or equivalent) network layer **506**, TCP/IP network layer **508**, and network interface **510**, communicates with an external network **514**, such as the Internet, by way of a NAT translator **512**. The computing device communicates with an external proxy server **518** embodied within an exemplary computing device **516** comprising the external proxy server, a sockets network layer **520**

(e.g., Microsoft Winsock or the like) for interfacing with a TCP/IP network layer **522**, and multiple network interfaces **526**, **528** to handle the incoming and outgoing network traffic. In one embodiment, the computing device **516** further comprises an optional server driver (discussed below).

The computing device **500** utilizes an H.323 application program **502** to establish a telecommunication session with an H.323 endpoint **530**. Rather than allowing the application program **502** to directly communicate with an external (with respect to the NAT translator **512**) computing device, the application program is tricked instead into establishing a circular networking connection **532** through the LSP **504** and Winsock **506** network layers, rather than with the endpoint.

That is, when the application program starts **502**, it queries its operating system for the network address of host computing device **500**, and an available communication port. As discussed above in FIG. 2, the Winsock **506** passes this query to the LSP **504**, which replies with address and port information retrieved from an external proxy server **518**. As the application program communicates with the endpoint **530**, the LSP receives the communication, forwards it to the proxy server, which in turn sends it to the endpoint. Responses from the endpoint are received by computing device **516**, tunneled back to the LSP, which in turn provides it to the application program through the Winsock. In such fashion, the application program and endpoint believe they are directly communicating.

When the application program **502** sends call setup data for the endpoint, according to the H.323 protocol, the application program provides its host's network address and communication port to the endpoint **530** as application data sent to the endpoint. Since the LSP is providing the application program with

configuration data from the proxy server **518**, call setup is not affected by the translating access point **512**. The application program waits for notification of a successful call setup.

In one embodiment, when the LSP forwards the call setup request to the external proxy server **518**, the proxy server attempts to set up the call with the endpoint **530** using the network configuration given to the proxy client for the application program. If call setup is successful, the LSP **504** is notified of the success, and the LSP in turn notifies the application program. A tunnel is established by the proxy client to the external proxy server that is used for forwarding the telecommunication session between the computing device **500** and the endpoint **530**. The proxy server may perform optimizations, such as compression, multiplexing, encryption, etc. to data transferred between the endpoint and the application program.

As noted above, computing device **516** may further comprise a proxy server driver **524**. This driver may be used to offload processing by the proxy server so that the proxy server is only responsible for establishing a protocol with an endpoint, and after successful establishment, the driver **524** then maintains the communication tunnel between the application program and the endpoint. For example, in the illustrated example, once a telecommunications session has been established, the UDP network traffic for the H.323 audio communication can be tunneled by the proxy server driver to the proxy client within the computing device **500**.

FIG. 6 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which certain aspects of the

illustrated invention may be implemented. For example, an exemplary system for embodying the proxy client or proxy server of FIG. 2 includes a machine **600** having system bus **602** for coupling various machine components.

Typically, attached to the bus are non-programmable and programmable processors **604**, a memory **606** (e.g., RAM, ROM), storage devices **608**, a video interface **610**, and input/output interface ports **612**. The machine may also include embedded controllers, Programmable Logic Devices (PLD), Programmable Logic Arrays (PLA), Programmable Array Logic (PAL), Generic Array Logic (GAL), Field-Programmable Gate Arrays (FPGA), Application Specific Integrated Circuits (ASIC), computers, smart cards, or another machine, system, etc.

The machine is expected to operate in a networked environment using logical connections to one or more remote machines **614**, **616** through a network interface **618**, modem **620**, or other communication pathway. Machines may be interconnected by way of a wired or wireless network **622** including an intranet, the Internet, local area networks, wide area networks, cellular, cable, laser, satellite, microwave, Blue Tooth, optical, infrared, or other carrier technology.

The invention may be described by reference to different high-level program modules and/or low-level hardware contexts that may be stored in memory **606** and/or storage devices **608**. Program modules include procedures, functions, programs, components, data structures, and the like, for performing particular tasks or implementing particular abstract data types. One skilled in the art will realize that program modules and low-level hardware contexts can be interchanged with low-level hardware instructions, and are collectively referenced hereafter as "directives." One will further appreciate that directives may be recorded or carried in a compressed, encrypted, or otherwise encoded format

without departing from the scope of this patent, even if the instructions must be decrypted, decompressed, compiled, interpreted, or otherwise manipulated prior to their execution or other utilization by the machine.

Memory **606**, storage devices **608**, and associated media, can store data and directives for the machine **600**. Program modules may be implemented within a single machine, or processed in a distributed network environment, and stored in both local and remote memory. Memory and storage devices include hard-drives, floppy-disks, optical storage, magnetic cassettes, tapes, flash memory cards, memory sticks, digital video disks, biological storage, and the like, as well as wired and wireless transmission environments, such as network **622**, over which directives may be delivered in the form of packets, serial data, parallel data, or other suitable transmission format.

Thus, for example, with respect to the illustrated embodiments, assuming machine **600** operates an H.323 telecommunication application program and the proxy client, then remote devices **614**, **616** may respectively be a machine embodying the proxy server, and an H.323 communication endpoint. It will be appreciated that remote machines **614**, **616** may be configured like machine **600**, and therefore include many or all of the elements discussed for machine. It should also be appreciated that machines **600**, **614**, **616** may be embodied within a single device, or separate communicatively-coupled components, and may include or be embodied within routers, bridges, peer devices, web servers, etc.

Illustrated methods, and corresponding written descriptions thereof, are intended to illustrate machine-accessible media storing directives, or the like, which may be incorporated into single and multi-processor machines, portable computers, such as handheld devices including Personal Digital Assistants

(PDAs), cellular telephones, and the like. Directives, when accessed, read, executed, loaded into, or otherwise utilized by a machine, causes the machine to perform the illustrated methods. The figures, written description, and claims may variously be understood as representing instructions taken alone, instructions as organized in a particular form, e.g., packet, serial, parallel, etc., and/or instructions together with their storage or carrier media.

Having described and illustrated the principles of the invention with reference to illustrated embodiments, it will be recognized that the illustrated embodiments can be modified in arrangement and detail without departing from such principles.

And, even though the foregoing discussion has focused on particular embodiments, it is understood that other configurations are contemplated. In particular, even though expressions such as "in one embodiment," "in another embodiment," or the like are used herein, these phrases are meant to generally reference embodiment possibilities, and are not intended to limit the invention to particular embodiment configurations. As used herein, these terms may reference the same or different embodiments, and unless implicitly or expressly indicated otherwise, embodiments are combinable into other embodiments. Consequently, in view of the wide variety of permutations to the above-described embodiments, the detailed description is intended to be illustrative only, and should not be taken as limiting the scope of the invention.

What is claimed as the invention, therefore, is all such modifications as may come within the scope and spirit of the following claims and equivalents thereto.

What is claimed is:

1. A method, comprising:

a first machine communicating with a second machine using a protocol that sends the first machine's network configuration data in an application data sent to the second machine;

receiving from a network configuration server a network configuration not subject to translation by the translating access point; and

providing according to the protocol said received network configuration to the second machine so that said communicating may traverse a translating access point which translates network traffic so as to apparently originate from the access point without breaking the protocol.

2. The method of claim 1, wherein the network configuration data comprises a network address, the method further comprising:

establishing a tunnel with the first machine for receiving network traffic sent to the network address.

3. The method of claim 2, wherein the tunnel is established between the network configuration server and the first machine.

4. The method of claim 1, wherein the protocol is a selected one of an audio protocol, a visual protocol, and audiovisual protocol, and a telecommunication protocol.

5. The method of claim 1, wherein the translating access point has a first interface communicatively coupled with a first network, and a second interface communicatively coupled with a second network, the method further comprising:

associating a network address with the second interface;

receiving network traffic from the first machine on the first interface; and

translating said received network traffic so that it appears to originate from the network address associated with the second interface.

6. A method for communicating through an access point coupling plural machines on a first network to a second machine on a second network by translating first network traffic so as to apparently originate from the access point, comprising:

receiving a request for a first address of a first machine on the first network;

allocating a second address from a server on the second network;

providing the second network address in response to the request;

transmitting through the access point at least one network packet comprising a packet origin, and a data payload comprising the second network address; and

translating by the access point of the packet origin so that the network packet apparently originates from the access point.

7. The method of claim 6, further comprising:

establishing a tunnel between the first machine and the server; and

the first machine receiving, through the tunnel, network traffic sent to the second address.

8. The method of claim 7, wherein the access point performs selected ones of: network address translation, and port translation on the at least one network packet.

9. The method of claim 6, further comprising:

providing a network address translation (NAT) based router between the first machine and the second machine so that communication between said first and second machines is NAT translated at least once.

10. The method of claim 9, wherein the NAT based router is the access point.

11. The method of claim 6, further comprising:

communicatively coupling the server to the second network such that network traffic from the server reaches the second network without translation.

12. The method of claim 11, wherein the second network is the Internet.

13. The method of claim 6, wherein the packet origin address is the first network address.

14. The method of claim 6, further comprising:

executing a networking application program, said program issuing the request for the first address of the first machine, and storing said provided second address as the data payload;

wherein the networking application program is unaware of said translating.

15. The method of claim 6, wherein the first machine comprises:

a network interface communicatively coupled to the first network;

a first memory for storing an operating system providing network services; and

a second memory for storing a network driver communicatively coupling the network interface to said network services, said network driver performing said allocating the second address, and providing the second address responsive to the request for the first address.

16. The method of claim 15, further comprising:

executing a networking application program which issues the request for the first address; and

the network driver providing the second network address responsive to said networking application program request.

17. A method for machines on an internal network to utilize protocols embedding machine network addresses within network traffic data when such traffic routes through an access point that shares a single address on an external network with said machines, the method comprising:

receiving first network traffic from a network driver executing on a first machine of the internal network, said first traffic having an apparent origin of the single address;

allocating an external address on the second network;

providing the external address to a network driver of the first machine; and

establishing a tunnel through the access point to the network driver so that network traffic for the external address is received by the network driver.

18. The method of claim 17, further comprising:

receiving second network traffic from an application program executing on the first machine, said second traffic having an apparent origin of the access point, and a data payload encoding an identified address determined by the application program for the first machine.

19. The method of claim 18, wherein the application program is a telecommunication program, the method further comprising:

contacting by the network driver of a call handling server on the external network, said server performing said allocating the external address and establishing the tunnel;

initiating a call by said program to an endpoint;

notifying said server of said initiating;

establishing said call to the endpoint by said server;

notifying the network driver of success/failure of said establishing; and

notifying said program of said success/failure.

20. The method of claim 19, wherein:

the application program telecommunicates with the network driver; and

the endpoint telecommunicates with the server.

21. A method for a first machine on an local area network (LAN) to communicate with a wide area network (WAN) through an access point configured to alter LAN network traffic so that it appears to originate from the WAN, the method comprising:

providing layer-based network services including an application layer, a network driver layer, and a session layer, wherein said driver is called before said session layer;

executing an application program configured to identify a first address of the first machine, embed said identified first address within network traffic data, and send said network traffic data to a communication endpoint;

providing a WAN address to said program so that said program can embed the WAN address within the network traffic data; and

establishing a first communication session between said program and said driver, a second communication session between said driver and the server, and a third communication session between the server and the said endpoint.

22. The method of claim 21, further comprising:

contacting a server on the WAN to obtain the WAN address;

receiving a call setup from said program for the endpoint;

establishing a call from the server to the endpoint;

connecting said program call to said driver; and

transparently forwarding said program call by said driver to the server.

23. The method of claim 21, wherein the session layer comprises the Microsoft Winsock Application Programming Interface.

24. The method of claim 21, wherein said network services are arranged according to the ISO/OSI model.

25. An apparatus, comprising a readable medium having instructions encoded thereon for execution by a processor, said instructions capable of directing the processor to perform:

communicating by a first machine with a second machine using a protocol that sends the first machine's network configuration data in an application data sent to the second machine;

receiving from a network configuration server a network configuration not subject to translation by the translating access point; and

providing according to the protocol said received network configuration to the second machine so that said communicating may traverse a translating access point which translates network traffic so as to apparently originate from the access point without breaking the protocol.

26. The apparatus of claim 25, said instructions comprising further instructions capable of directing the processor to perform:

establishing a tunnel with the first machine for receiving network traffic sent to the network address.

27. The apparatus of claim 25, wherein the translating access point has a first interface communicatively coupled with a first network, and a second interface communicatively coupled with a second network, said instructions comprising further instructions capable of directing the processor to perform:

associating a network address with the second interface;

receiving network traffic from the first machine on the first interface; and

translating said received network traffic so that it appears to originate from the network address associated with the second interface.

28. An apparatus for communicating through an access point coupling plural machines on a first network to a second machine on a second network by translating first network traffic so as to apparently originate from the access point, comprising a readable medium having instructions encoded thereon for execution by a processor, said instructions capable of directing the processor to perform:

receiving a request for a first address of a first machine on the first network;

allocating a second address from a server on the second network;

providing the second network address in response to the request;

transmitting through the access point at least one network packet comprising a packet origin, and a data payload comprising the second network address; and

translating by the access point of the packet origin so that the network packet apparently originates from the access point.

29. The apparatus of claim 28, said instructions comprising further instructions capable of directing the processor to perform:

establishing a tunnel between the first machine and the server; and

the first machine receiving, through the tunnel, network traffic sent to the second address.

30. The apparatus of claim 28, wherein a network address translation (NAT) based router between the first machine and the second machine translates communication between said first and second machines.

31. The apparatus of claim 30, wherein the NAT based router is the access point.

32. The apparatus of claim 28, said instructions comprising further instructions capable of directing the processor to perform:

executing a networking application program, said program issuing the request for the first address of the first machine, and storing said provided second address as the data payload;

wherein the networking application program is unaware of said translating.

33. A system for machines on an internal network to utilize protocols embedding machine network addresses within network traffic data when such traffic routes through an access point that shares a single address on an external network with said machines, the method comprising:

receiving means for receiving first network traffic from a network driver executing on a first machine of the internal network, said first traffic having an apparent origin of the single address;

allocating means for allocating an external address on the second network;

providing means for providing the external address to a network driver of the first machine; and

establishing means for establishing a tunnel through the access point to the network driver so that network traffic for the external address is received by the network driver.

34. The system of claim 33, further comprising:

receiving means for receiving second network traffic from an application program executing on the first machine, said second traffic having an apparent origin of the access point, and a data payload encoding an identified address determined by the application program for the first machine.

35. The system of claim 34, wherein the application program is a telecommunication program, the system further comprising:

means for contacting by the network driver of a call handling server on the external network, said server performing said allocating the external address and establishing the tunnel;

initiating means for initiating a call by said program to an endpoint;

notifying means for notifying said server of said initiating;

establishing means for establishing said call to the endpoint by said server;

notifying means for notifying the network driver of success/failure of said establishing; and

notifying means for notifying said program of said success/failure.

36. An apparatus for facilitating a first machine on an local area network (LAN) to communicate with a wide area network (WAN) through an access point configured to alter LAN network traffic so that it appears to originate from the WAN, the method comprising:

means for providing layer-based network services including an application layer, a network driver layer, and a session layer, wherein said driver is called before said session layer;

means for executing an application program configured to identify a first address of the first machine, embed said identified first address within network traffic data, and send said network traffic data to a communication endpoint;

means for providing a WAN address to said program so that said program can embed the WAN address within the network traffic data; and

means for establishing a first communication session between said program and said driver, a second communication session between said driver and the server, and a third communication session between the server and the said endpoint.

37. The apparatus of claim 36, further comprising:

means for contacting a server on the WAN to obtain the WAN address;

means for receiving a call setup from said program for the endpoint;

means for establishing a call from the server to the endpoint;

means for connecting said program call to said driver; and

means for transparently forwarding said program call by said driver to the server.