(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 March 2003 (06.03.2003)

PCT

(10) International Publication Number
WO 03/019404 A1

(51) International Patent Classification⁷: G06F 15/173

(21) International Application Number: PCT/IL02/00714

(22) International Filing Date: 29 August 2002 (29.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/316,198 30 August 2001 (30.08.2001) US

(71) Applicant *(for all designated States except US)*: **RIVERHEAD NETWORKS INC.** [US/US]; 3000 Sand Hill Road, Bldg. 4, Suite 180, Menlo Park, CA 94025 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **PAZI, Guy** [IL/IL]; 50 Haprahim Street, 47231 Ramat Hasharon (IL). **BREMLER-BAR, Anat** [IL/IL]; 8 Hashomer Street, 58272 Holon (IL). **RIVLIN, Rami** [IL/IL]; 23A Sasha Argov Street, 69620 Tel Aviv (IL). **TOUITOU, Dan** [IL/IL]; 7/30 Matityahu Mendel Street, 52287 Ramat Gan (IL).

(74) Agents: **SANFORD T. COLB & CO.** et al.; P.O. Box 2273, 76122 Rehovot (IL).

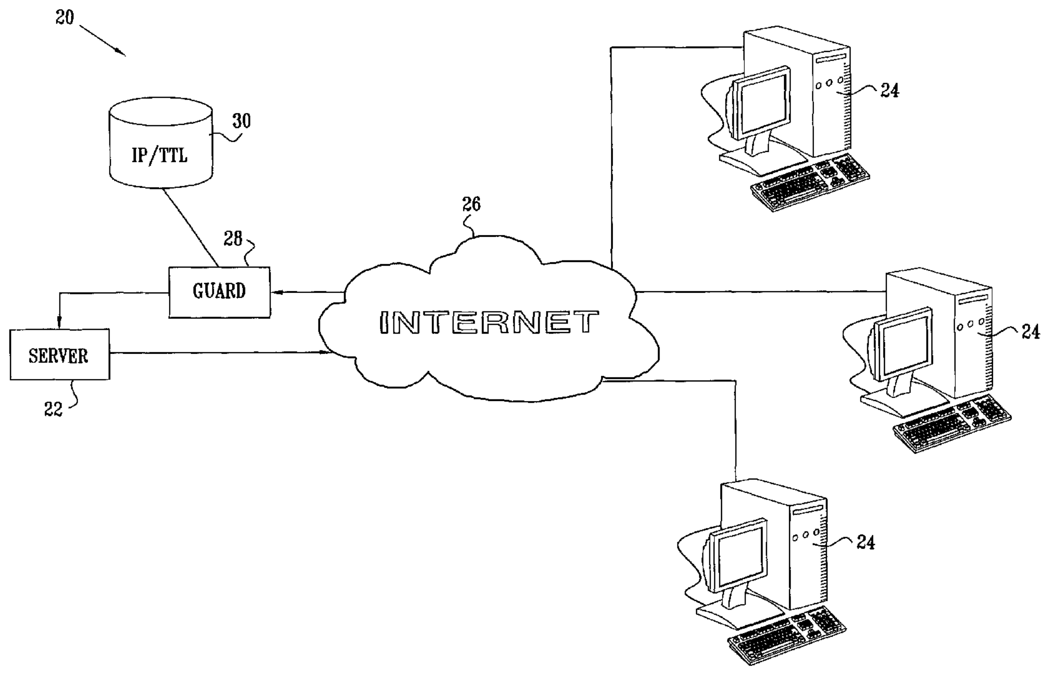
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *with international search report*

*[Continued on next page]*

# (54) Title: PROTECTING AGAINST DISTRIBUTED DENIAL OF SERVICE ATTACKS



**(57) Abstract:** A method for authenticating packet communication traffic includes receiving a data packet sent over a network (26) from a source address (24) to a destination address (22) and reading from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address. The authenticity of the source address is assessed responsive to the value.

WO 03/019404 A1

# PROTECTING AGAINST DISTRIBUTED DENIAL OF SERVICE ATTACKS

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application 60/316,198, filed August 30, 2001, entitled "Methods and Apparatus for Protecting Against Spoofed Packet-Based Distributed Denial of Service Attacks." It is related to co-pending U.S. Patent Application 09/929,877, filed August 14, 2001, entitled "Methods and Apparatus for Protecting Against Overload Conditions on Nodes of a Distributed Network." Both these related applications are assigned to the assignee of the present patent application, and their disclosures are incorporated herein by reference.

## FIELD OF THE INVENTION

The present invention relates generally to computer networks, and specifically to methods and systems for protecting against denial of service attacks in computer networks.

## BACKGROUND OF THE INVENTION

In a Denial-of-Service (DoS) attack, an attacker bombards a victim network or server with a large volume of message traffic, consuming the victim's available bandwidth, CPU capacity, or other critical system resources, and eventually bringing the victim to a situation in which it is unable to serve its legitimate clients. Distributed DoS (DDoS) attacks can be even more damaging, as they involve creating artificial network traffic from multiple sources simultaneously. In a "conventional" massive-bandwidth attack, the source of the attack may be traced with the help of statistical analysis of the source Internet Protocol (IP) addresses of incoming packets. The victim can subsequently filter out any traffic originating from the suspect IP addresses, and can use the evidence to take legal action against the attacker.

Many attacks, however, now use "spoofed" IP packets – packets containing a bogus IP source address – making it more difficult for the victim network to defend itself against attack. Various methods have been proposed for detecting DDoS attacks using spoofed packets. For example, Poletto describes a number of uses of traffic statistics in detecting DDoS attacks in "Practical Approaches to Dealing with DDoS Attacks," presented at *North America Network Operators' Group* (NANOG22, Scottsdale, Arizona, May, 2001). Random distribution of IP source addresses is one sign of spoofing. Another possible sign of spoofing is randomization of the Time-To-Live (TTL) field in the IP packet header, which is supposed to indicate the number of hops that a packet has traversed through the network since leaving its source. Sudden changes in the statistical distribution of TTL values can also be detected. Although these statistical measures may enable a computer to recognize when an attack is occurring, however, they do not provide the means to distinguish between spoofed and legitimate packets during the attack.

## SUMMARY OF THE INVENTION

It is an object of some aspects of the present invention to provide methods and devices for detecting spoofed packets.

It is a further object of some aspects of the present invention to provide methods and devices for guarding a computer or network against DDoS attacks.

In preferred embodiments of the present invention, a network guard system screens the number of hops traversed by incoming packets in order to assess which packets constitute legitimate network traffic, and which are spoofed. Typically, in IP networks, the number of hops is indicated by the TTL field in the header of each packet. The guard system stores records of IP source addresses together with the expected TTL value (or number of hops) of packets arriving from each source address. If the TTL value of an incoming packet matches the stored TTL value or number of hops for the source address of the packet (to within a tolerance that may be allowed for network instabilities), the guard system recognizes the packet as likely to be legitimate. In this case, the guard system typically allows the packet to pass on to its destination, although further anti-DDoS processing measures may also be carried out before the packet is delivered.

Otherwise, if the TTL value of the packet does not match the stored value, the guard treats the packet as suspect, and either discards it or allows it to pass with a priority that is reduced relative to packets that are known to be legitimate. In this way, suspect packets are prevented from interfering with service of legitimate traffic. The guard system may be active at all times, or it may alternatively be activated only in response to traffic conditions that raise suspicion that a DDoS attack may be in progress.

The guard system is preferably provided with a mechanism for learning the proper TTL values for different IP source addresses. When the guard system receives a packet for whose IP source address there is no corresponding stored TTL value, the system initiates a procedure for verifying the TTL value of the packet. The verified value and the corresponding IP address may then be stored.

Although preferred embodiments are described herein with reference to specific communication protocols, the principles of the present invention may be applied using other protocols, as well, so long as means are provided for recording the number of hops a packet has traversed.

There is therefore provided, in accordance with a preferred embodiment of the present invention, a method for authenticating packet communication traffic, including:

receiving a data packet sent over a network from a source address to a destination address;

reading from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address; and

assessing authenticity of the source address responsive to the value.

Preferably, assessing the authenticity includes comparing the value of the field to a reference value associated with the source address. Most preferably, comparing the value of the field includes determining whether the value matches the reference value to within a predefined tolerance. Additionally or alternatively, comparing the value of the field includes comparing the value to an aggregate reference value associated with a subnet to which the source address belongs.

Preferably, assessing the authenticity includes, if there is no reference value associated with the source address that matches the value of the field, validating the value of the field for use as the reference value. Most preferably, reading the value of the field includes reading a first value of the field, and validating the value of the field includes sending a message packet over the network to the source address, receiving a response packet over the network from the source address in response to the message packet, reading a second value of the field from the response packet, and comparing the first and second values of the field.

Further preferably, sending the message packet includes encoding the first value of the field in the message packet, so that the encoded first value is returned in the response packet, and comparing the first and second values includes reading the first value from the response packet. In a preferred embodiment, encoding the first value includes inserting a cookie in the message packet, and receiving the response packet includes authenticating the response packet based on the cookie. In preferred embodiments, sending the message packet includes sending a Transport Control Protocol (TCP) SYN packet, or a Domain Name System (DNS) request packet, or a PING request.

Typically, receiving the data packet includes receiving a sequence of data packets having respective values of the field, and validating the value includes proceeding to validate the value only if a rate of validating the values for the sequence of data packets is no more than a predetermined maximum.

In a preferred embodiment, comparing the value of the field includes comparing the value to a plurality of reference values, which are determined by decrementing a plurality of respective initial values of the field by the same number of hops.

In another preferred embodiment, comparing the value of the field includes determining the reference value based on the value of the field in a protocol handshake exchanged between the source and destination addresses. Preferably, the protocol handshake includes a Transport Control Protocol (TCP) three-way handshake.

In a preferred embodiment, receiving the data packet includes receiving an Internet Protocol (IP) packet, and reading the value of the field includes reading a Time-To-Live (TTL) of the IP packet.

Preferably, receiving the data packet includes intercepting the data packet prior to the packet's reaching the destination address, and the method includes delivering the packet to the destination address depending on the assessed authenticity of the packet. Most preferably, delivering the packet includes discarding the packet if the packet is assessed as inauthentic. Alternatively, delivering the packet includes delivering the packet to the destination address with a reduced level of priority if the packet is assessed as inauthentic. Optionally, delivering the packet includes performing a further check to verify the authenticity after assessing the authenticity of the source address responsive to the value and before forwarding the packet to the destination address.

There is also provided, in accordance with a preferred embodiment of the present invention, apparatus for authenticating packet communication traffic, including a guard device, which is adapted to receive a data packet sent over a network from a source address to a destination address, to read from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address, and to assess authenticity of the source address responsive to the value.

Preferably, the apparatus includes a memory, which is adapted to store a record containing a reference value associated with the source address, wherein the guard device is adapted to read the reference value from the memory and to assess the authenticity of the source address by comparing the value of the field to the reference value.

There is additionally provided, in accordance with a preferred embodiment of the present invention, a computer software product for authenticating packet communication traffic, the product including a computer-readable medium in which program instructions are stored, which instructions, when read by a computer, cause the computer to receive a data packet sent over a network from a source address to a destination address, to read from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address, and to assess authenticity of the source address responsive to the value.

The present invention will be more fully understood from the following detailed description of the preferred embodiments thereof, taken together with the drawings in which:

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram that schematically illustrates a computer network system, in accordance with a preferred embodiment of the present invention;

Fig. 2 is a flow chart that schematically illustrates a method for screening incoming packets received from a network, in accordance with a preferred embodiment of the present invention; and

Fig. 3 is a flow chart that schematically illustrates a method for verifying TTL values, in accordance with a preferred embodiment of the present invention.

## DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 is a block diagram that schematically illustrates a computer network system 20, in accordance with a preferred embodiment of the present invention. A server 22 communicates with clients 24 via a wide-area network (WAN) 26, typically the Internet. To prevent DDoS attacks on server 22, a guard device 28 intercepts incoming packets from network 26 that are addressed to server 22. Optionally, the guard device may process outgoing traffic, as well. The guard device compares the IP source address and TTL field of each packet that it intercepts against reference values stored in a database 30. (Although in the present embodiment database 30 is used to store IP/TTL records, it will be understood that substantially any suitable memory device and data structure may be used for storing this information, and not only a database.)

If the source address and TTL value of an incoming packet match an entry in database 30, guard device 28 passes the packet on to server 22. Alternatively, further anti-DDoS processing measures may be carried out before the packet is delivered to the server. Otherwise, if the source address and TTL value do not match an entry in the database, the guard device regards the packet as suspect and, typically, either discards the packet or passes it on to the server with limiting conditions. For example, the packet may be passed on with reduced priority or limited rate, or with a certain queuing service option or some randomly-selected limitation. The methods used by the guard device in building database 30 and validating incoming packets are described in detail hereinbelow.

Typically, guard device 28 comprises a general-purpose computer, which is programmed in software to carry out the functions described herein. The software may be downloaded to the computer in electronic form, over a network, for example, or it may alternatively be supplied to the computer on tangible media, such as CD-ROM. Further alternatively, guard device 28 may be implemented in dedicated hardware logic, or using a combination of hardware and software elements. The guard device may be a standalone unit, or it may alternatively be integrated with other communication or computing equipment, such as a firewall or intrusion detection system.

For the sake of simplicity, Fig. 1 shows guard device 28 as protecting only server 22. In practical applications, however, the guard device may be used to protect a cluster of servers, or it may be used to protect an entire LAN, intranet or a collection of servers whose traffic is diverted to the guard device. The guard device may be deployed in configurations similar to firewalls known in the art. The methods described hereinbelow for detecting spoofed packets may also be used by a standalone computer, running appropriate software, without a separate guard device.

Before describing in detail the methods of operation of guard device 28, it will be useful to review certain features of the TTL field and its use in IP networks. Time-To-Live (TTL) is a mechanism used in IP to ensure that packets never travel more than a limited number of hops in the Internet. The value of the field is initialized to some value *T*, which is the bound on the number of hops the packet is allowed to traverse in the network. Each router should decrement the TTL value of each packet passing through it. If the TTL value drops to zero, the packet is removed (dropped) from the Internet.

At any point in the network, the TTL value of an incoming packet originating from some source depends on the initial TTL value that was set by the source. Different operating systems initialize the TTL field in IP packets they create to different values. The standard initial value is 256, but other initial values that are known to exist are 128 (Windows), 64 (Unix), 60, 32 and 30. Thus, when examining the TTL value of a packet, its initial value should be taken into account. Because the vast majority of source-destination path lengths in the Internet are smaller than even the smallest initial TTL value (30), it is generally possible to infer the number of hops the packet traversed from the source based on the value that appears in the packet at a given examination point. Because some of the possible initial TTL values are confusingly close (for example, 64 and 60, or 32 and 30), it may be necessary in some cases to record two alternative path length values. (Note, however, that the operating systems that use initial TTL values of 60 and 30 are old and may become obsolete.) Some operating systems use different initial TTL values for different transport protocols (such as the User Datagram Protocol - UDP and Transport Control Protocol – TCP). Therefore, in authenticating TTL values, the transport protocol may be taken into consideration, as well.

The TTL value of any packet received from the Internet is a function of the length of the Internet path between the packet source and the destination. The paths taken by successive packets can change, however, so that different packets from the same source may have different TTL values. Therefore, a mismatch between the TTL value of a packet received by guard device 28 and the reference value stored in database 30 is not conclusive evidence that the packet is spoofed.

Path changes typically occur due to the following reasons:

1. Failures in the Internet.
2. Changes in routing policies used in the Internet.
3. Changes in the Internet infrastructure or topology (such as addition of a new link).
4. Load balancing mechanisms.
5. Use of network address translator (NAT) machines for Internet access.

6. Protocol-dependent path variations, due to transport-layer processing at the packet source, for example, or to processing by intermediate devices, such as firewall machines.

The first three types of events are generally rare, and typically result in stabilization of the characteristic TTL at a new value after a few seconds. Guard device 28 can then validate the new value (as described below) and store it in database 30. In practice, the inventors have found that more than 90% of IP addresses exhibit stable TTL values over any given 24-hour period. The last three types of events typically lead to continual variations of the TTL value within a small, consistent range. Therefore, generally speaking, guard device 28 should be tolerant of small deviations of the TTL values of incoming packets from the corresponding values in database 30.

Furthermore, a given packet source may generate different IP packets with different initial TTL values. As noted above, some operating systems use different initial TTL values for different transport protocols. As another example, when a packet must be retransmitted, due to a TCP timeout, the packet source often sets the initial TTL value to 256 in order to give the packet a longer Internet lifetime, rather than the default initial TTL value that is usually used. Guard device 28 is preferably programmed to recognize such eventualities and to allow the retransmitted packet to reach server 22. Thus, if the guard device determines that the characteristic TTL value for packets received from a given source address is 103, for example, representing 25 hops from an initial value of 128, then the guard device should also be prepared to accept packets from this source address with a TTL values of 231, and possibly 39, 35, 7 and 5, as well.

Moreover, as noted above, the actual number of hops traversed by a packet may be ambiguous. For example, if the received TTL value is 56, the path length can be either 4 hops (corresponding to an initial TTL value of 60) or 8 (initial TTL value of 64). In such cases, guard device 28 may consider all the possible TTL values. Additional ambiguity may arise because of the protocol-dependent initial TTL values used by some operating systems. When there are ambiguities of this sort, guard device 28 may use a confidence rating system to classify incoming packets with TTL values that may be correct, rather than a simple "go/no go."

Database 30 may hold reference TTL values as either absolute values (such as 103 in the example above), relative values (25 in this example) or both. To keep this reference data up to date, guard device 28 preferably rechecks the values periodically, using the verification procedure described below. Preferably, only authenticated IP source/TTL entries are kept in the database. To save memory space, the guard device may manage the database as a cache, discarding old entries that have not been used recently. As a further means for saving memory and processing time, database 30 may hold TTL values by subnet (a TTL value for each different 24-bit IP address prefix, for example), in addition to or instead of entries for individual IP source addresses.

Fig. 2 is a flow chart that schematically illustrates a method used by guard device 28 for validating incoming packets from network 26, in accordance with a preferred embodiment of the present invention. The method may be initiated when the guard device receives an incoming packet from the network, at a packet interception step 40. Alternatively, guard device 28 may become active only under stress conditions, in which a DDoS attack on server 22 is suspected, due to particularly heavy incoming traffic or to other traffic statistics. In this case, some additional trigger is required for step 40. For example, guard device 28 may become active when it detects an unusually large number of incoming TCP SYN-ACK or DNS request packets.

Furthermore, some incoming packets may not be subject to suspicion, and therefore need not be validated by the guard device. For example, once a TCP connection has been established between server 22 and one of clients 24, using the conventional TCP three-way handshake, each side of the connection can be certain that the IP address of the other side is legitimate, not spoofed. Therefore, guard device 28 preferably allows TCP/IP packets on established connections to pass through to server 22 without interruption for validation. Handling of the TCP handshake itself by the guard device is described further hereinbelow. On the other hand, all packets sent over connectionless protocols, such as UDP or ICMP, must generally be validated. Packets using the Domain Name System (DNS) protocol must generally be validated, as well. The need to validate packets of other protocol types (or the absence of such need) will be apparent to those skilled in the art.

Guard device 28 checks the IP source address of the incoming packet against database 30, at a record checking step 42. If there is a record in the database corresponding to this source address, the guard device checks the TTL value held in the database against the TTL value recorded in the packet header. The comparison may be either exact or relative. In exact comparison, the TTL value of the incoming packet must match the record in the database exactly. Thus, if the TTL value recorded in the database is 103 (as in the example cited earlier), then the incoming packet is considered valid only if its TTL value is exactly 103. The advantage of this approach is that it reduces the probability that a hacker mounting a DDoS attack will successfully guess the correct TTL value to insert in a spoofed packet. The exact match criterion may be relaxed by ±1 or ±2 TTL steps in order to account for small variations due to load balancing, NAT machines and protocol-dependent effects, as noted above. Note that the exact comparison approach rejects TTL variations that may arise due to different initial TTL settings, as described above.

Alternatively or additionally, guard device 28 may validate an IP source address based on the TTL value. As long as an IP address is considered valid, the guard device allows all traffic from the address to pass through to server 22. An IP source address can be validated, for example, simply by testing an incoming packet from the address. If the TTL value of the packet is correct, the IP source address is then considered valid for a certain period of time. As another example, the IP source address may be validated if the fraction of packets arriving from the address that have the correct TTL value is reasonably large (say 20%). The benefit of such an IP validation is that it allows faster handling of most packets, and reduces the overhead associated with testing the TTL.

In relative comparison, guard device 28 checks the number of hops that the incoming packet has taken, rather than the exact value of the TTL field. In other words, referring again to the example above, if a value of 25 hops is recorded in database 30 for a given IP source address, then TTL values of 5, 7, 35, 39, 103 and 231 will be considered valid. As noted above, when there is more than one probable value for the number of hops, there may be even more possible TTL values. Relative comparison, in contrast to exact comparison, reduces the likelihood of rejecting valid packets. When TTL values are stored in database 30 by subnet, only relative comparison can generally be used, since different packet sources in the same subnet may have different operating systems and will therefore set different initial TTL values. The relative match criteria are also preferably relaxed by a number of TTL steps to account for the small variations mentioned above, as well as for differences in path length within a subnet when database 30 stores TTL values by subnet.

If the TTL value of the incoming packet matches the database value, guard device 28 allows the packet to pass on for further processing, at a packet delivery step 44. The matching TTL value does not conclusively prove that the packet is not spoofed, since there is a certain probability that a hacker will have correctly guessed the expected TTL value. Therefore, the packet may be subject to further anti-DDoS processing before it is submitted to the server. Alternatively, guard device 28 may pass the packet through directly to server 22. In any case, the probability of a spoofed packet reaching the server is substantially reduced.

If the TTL value of the incoming packet does not match a value in database 30 (either because there is no entry for the IP source address of the packet, or because the TTL value in the database is different), guard device 28 treats the packet as suspect, at an invalid packet processing step 46. The simplest response for the guard device to implement is to consider the suspect packet to be invalid and discard it. This approach will lead the guard device to drop a certain number of valid packets. In most cases, higher-level protocols running on clients 24 will resend the packets later, by which time the guard device will have learned the appropriate IP source address and TTL value, as described below. Alternatively, the guard device may allow the suspect packet to pass through to server 22, but at a lower service level than the presumably valid packets delivered at step 44. The decision as to how to treat suspect packets may be made by a system operator of server 22 and may also depend on the overall level of incoming traffic at the time the suspect packet is received.

When guard device 28 has received a suspect packet for which there is no entry in database 30, or in any case in which there is a likelihood that the suspect packet is, in fact, valid, the guard device attempts to verify that the TTL value of the packet is correct, at a TTL validation step 48. Preferably, the guard device sends a message to the IP source address of the suspect packet, and then examines the reply in order to verify the TTL value, at a TTL verification step 50. This method is described in detail hereinbelow with reference to Fig. 3. If the TTL value is successfully verified, guard device 28 stores the value in database 30 along with the corresponding IP source address, at a database update step 52. The verified TTL value is then available for use in validating packets received subsequently by the guard device.

Otherwise, if step 50 is unsuccessful, the TTL value and IP source address of the suspect packet are simply discarded, at a discard step 54.

Optionally, guard device 28 limits the number of new source addresses that it will validate during any given period of time. Otherwise, a hacker could bombard the guard device with packets from so many new addresses requiring validation, that the guard device might be unable to process the flow of legitimate packets.

Fig. 3 is a flow chart that schematically shows details of validation step 48, in accordance with a preferred embodiment of the present invention. Guard device 28 reads the IP source address and TTL value of the suspect incoming packet (received at step 40), at a header reading step 60. The guard device uses this information to construct an outgoing message packet addressed to the IP source address of the suspect packet, at a message generation step 62. The message is of a type that requires the recipient to respond, in such a way that if the guard device receives a response, it will know that the source IP address is real, not spoofed. A number of different types of messages may be used for this purpose, for example:

- A TCP SYN packet. This is the first step in the well-known TCP three-way handshake for establishing a TCP connection. The recipient is expected to respond with a TCP SYN-ACK packet, or with a RST packet if it rejects the connection. In either case, the returned packet will contain a TTL value and a sequence number corresponding to the packet sequence number of the original SYN packet. The TCP SYN packet may also include a cookie, as described below.
- A DNS request packet. The recipient is expected to return a DNS response packet, which will contain a TTL value and an identifier (ID) field, corresponding to the ID field inserted by the guard device in the request packet.
- A PING request sent over ICMP. The PING reply will contain information that enables the guard device to determine the TTL value. Preferably, a cookie containing encoded information is embedded in the 8-bit code field of the ICMP packet. In most cases, the PING reply will contain the same code field without change.

Preferably, guard device 28 inserts encoded information into the outgoing message packet (for example, into the TCP SYN or DNS request packet) that it sends at step 62, in such a way that the encoded information will be returned in the response from the remote IP source address. This information assists the guard device in verifying the TTL value provided by the response. For example, the guard device may set a cookie in the sequence number of the TCP SYN packet, and may set the source port field in the TCP header to be equal to the TTL value of the suspect packet that it has received. Alternatively or additionally, the TTL value of the suspect packet may be encoded in the cookie. SYN cookies are known in the art, although for purposes other than that described here.

As another example, guard device may set a cookie in an outgoing DNS request packet that it sends to an unrecognized IP source address. This approach is useful particularly for verifying the source of DNS replies sent from clients 24 to server 22. Preferably, the guard device sets the recursive flag of the DNS request to "NO." The query contained in the DNS request includes the original domain name of the incoming DNS reply, concatenated with a cookie indicating the received TTL value and certain information received in the DNS reply (such as the original recursive flag and original DNS ID). The new DNS ID of the outgoing DNS request contains at tag indicating that a cookie has been inserted in the packet and a pointer to the cookie in the domain name field. These elements of the DNS ID enable the guard device to easily recognize packets that do not contain a cookie and to efficiently process the contents of packets that do include cookies, as described below.

Guard device 28 waits to receive the expected response from the remote IP source address, at a response reception step 64. (If no response is received, the guard device will take no further action with respect to this address.) In the case of the encoded TCP SYN packet described above, for example, the response should be a SYN-ACK or RST packet. In this case, the guard device reads the sequence number of the received packet, and authenticates the packet by checking for the cookie in the sequence number. The guard device then reads the value of the TTL field from the IP header of the received packet, and compares this value to the destination port value in the packet's TCP header, at a TTL comparison step 66. The destination port value should be equal to the source port value set by the guard device in the SYN packet, i.e., to the TTL value of the suspect packet received earlier, at step 40. If the current and previous TTL values are equal, the guard device concludes that the values are correct and valid, and stores the IP source address and TTL value in database 30.

When guard device 28 receives a DNS reply, it first verifies the cookie. If the cookie is authenticated, the guard device compares the TTL value of the DNS reply packet to the encoded TTL value in the DNS domain name. If the comparison is successful, the remote IP source address is verified as a domain name server. Preferably, if the "recursive desired" flag was set in the original DNS request from the domain name server, the guard device now sends another DNS request to domain name server, using the original domain name and original DNS ID. Otherwise, the guard device sends a reply to server 22, "redirecting" it to the same external domain name server in order to avoid timeout and referral to a different domain name server.

Other methods may also be used for encoding information in the outgoing TCP SYN or DNS request packet at step 62. As another example, the TTL value of the suspect packet may be encoded in the sequence number of the SYN packet sent to the IP source address. When the guard device receives the SYN-ACK or RST packet in response, it decodes the sequence number in order to retrieve the previous TTL value and compare it to the TTL field of the current packet at step 66.

The procedure of Fig. 3 may also be applied when guard device 28 receives an incoming TCP SYN or SYN-ACK packet at step 40, and the IP source address and TTL value of the packet do not match any entry in database 30. Since there is not yet a connection established between server 22 and the client 24 that sent the SYN or SYN-ACK packet, the guard device cannot know whether the IP source address of the incoming packet is legitimate or spoofed. It therefore suspends or discards the SYN or SYN-ACK packet that it received. The guard device then sends its own TCP SYN packet to the IP source address of the incoming packet, and waits to receive the SYN-ACK or RST response packet in order to verify the TTL value, as described above. When the client subsequently resends its original SYN or SYN-ACK, the verified TTL value will have been entered in database 30, and guard device 28 will allow the packet to pass through to server 22.

As noted above, guard device 28 may actively screen incoming packets at all times, or it may alternatively screen packets only under certain stress conditions. In the latter case, even while the guard device is inactive, it may still collect TTL values for inclusion in database 30. For example, the guard device may observe TCP handshakes carried out by server 22 and various clients 24 over network 26. A client initiating such a handshake sends a SYN packet to server 22, which responds by sending a SYN-ACK packet back to the client. The client must then respond with its own ACK in order to complete the connection. The SYN, SYN-ACK and ACK packets contain the appropriate sequence numbers. If the "client" is a spoofed source, it will not receive the SYN-ACK, and therefore will not return the final ACK. Thus, if guard device 28 observes SYN and ACK packets from the same IP source address, with the same TTL value and the appropriate sequence number in the ACK packet, it can be assured that the IP source address and TTL value are valid, and may make an entry in database 30 accordingly.

Although the preferred embodiments described herein are based specifically on IP and on certain transport and application protocols that run over IP, the principles of the present invention may similarly be applied using other network protocols, as well as other transport and application protocols meeting the functional requirements described above. It will thus be appreciated that the preferred embodiments described above are cited by way of example, and that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and subcombinations of the various features described hereinabove, as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description and which are not disclosed in the prior art.

## CLAIMS

1. A method for authenticating packet communication traffic, comprising:

receiving a data packet sent over a network from a source address to a destination address;

reading from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address; and

assessing authenticity of the source address responsive to the value.

2. A method according to claim 1, wherein assessing the authenticity comprises comparing the value of the field to a reference value associated with the source address.

3. A method according to claim 2, wherein comparing the value of the field comprises determining whether the value matches the reference value to within a predefined tolerance.

4. A method according to claim 2, wherein comparing the value of the field comprises comparing the value to an aggregate reference value associated with a subnet to which the source address belongs.

5. A method according to claim 2, wherein assessing the authenticity comprises, if there is no reference value associated with the source address that matches the value of the field, validating the value of the field for use as the reference value.

6. A method according to claim 5, wherein reading the value of the field comprises reading a first value of the field, and wherein validating the value of the field comprises:

sending a message packet over the network to the source address;

receiving a response packet over the network from the source address in response to the message packet;

reading a second value of the field from the response packet; and

comparing the first and second values of the field.

7. A method according to claim 6, wherein sending the message packet comprises encoding the first value of the field in the message packet, so that the encoded first value is returned in the response packet, and wherein comparing the first and second values comprises reading the first value from the response packet.

8. A method according to claim 7, wherein encoding the first value comprises inserting a cookie in the message packet, and wherein receiving the response packet comprises authenticating the response packet based on the cookie.

9. A method according to claim 6, wherein sending the message packet comprises sending a Transport Control Protocol (TCP) SYN packet.

10. A method according to claim 6, wherein sending the message packet comprises sending a Domain Name System (DNS) request packet.

11. A method according to claim 6, wherein sending the message packet comprises sending a PING request.

12. A method according to claim 5, wherein receiving the data packet comprises receiving a sequence of data packets having respective values of the field, and wherein validating the value comprises proceeding to validate the value only if a rate of validating the values for the sequence of data packets is no more than a predetermined maximum.

13. A method according to claim 2, wherein comparing the value of the field comprises comparing the value to a plurality of reference values, which are determined by decrementing a plurality of respective initial values of the field by the same number of hops.

14. A method according to claim 2, wherein comparing the value of the field comprises determining the reference value based on the value of the field in a protocol handshake exchanged between the source and destination addresses.

15. A method according to claim 14, wherein the protocol handshake comprises a Transport Control Protocol (TCP) three-way handshake.

16. A method according to any of claims 1-15, wherein receiving the data packet comprises receiving an Internet Protocol (IP) packet, and wherein reading the value of the field comprises reading a Time-To-Live (TTL) of the IP packet.

17. A method according to any of claims 1-15, wherein receiving the data packet comprises intercepting the data packet prior to the packet's reaching the destination address, and comprising delivering the packet to the destination address depending on the assessed authenticity of the packet.

18. A method according to claim 17, wherein delivering the packet comprises discarding the packet if the packet is assessed as inauthentic.

19. A method according to claim 17, wherein delivering the packet comprises delivering the packet to the destination address with a reduced level of priority if the packet is assessed as inauthentic.

20. A method according to claim 17, wherein delivering the packet comprises performing a further check to verify the authenticity after assessing the authenticity of the source address responsive to the value and before conveying the packet to the destination address.

21. Apparatus for authenticating packet communication traffic, comprising a guard device, which is adapted to receive a data packet sent over a network from a source address to a destination address, to read from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address, and to assess authenticity of the source address responsive to the value.

22. Apparatus according to claim 21, and comprising a memory, which is adapted to store a record containing a reference value associated with the source address, wherein the guard device is adapted to read the reference value from the memory and to assess the authenticity of the source address by comparing the value of the field to the reference value.

23. Apparatus according to claim 22, wherein the guard device is adapted to determine whether the value matches the reference value to within a predefined tolerance.

24. Apparatus according to claim 22, wherein the guard device is adapted to compare the value to an aggregate reference value associated with a subnet to which the source address belongs.

25. Apparatus according to claim 22, wherein the guard device is adapted, if there is no reference value associated with the source address in the memory that matches the value of the field, to validate the value of the field for use as the reference value.

26. Apparatus according to claim 25, wherein the value of the field that is read from the packet comprises a first value of the field, and wherein the guard device is adapted to validate the first value by sending a message packet over the network to the source address, receiving a response packet over the network from the source address in response to the message packet,

reading a second value of the field from the response packet, and comparing the first and second values of the field.

27. Apparatus according to claim 26, wherein the guard device is adapted to encode the first value of the field in the message packet, so that the encoded first value is returned in the response packet, and to read the first value from the response packet.

28. Apparatus according to claim 27, wherein the guard device is adapted to insert a cookie encoding the first value in the message packet, and to authenticate the response packet based on the cookie.

29. Apparatus according to claim 26, wherein the message packet comprises a Transport Control Protocol (TCP) SYN packet.

30. Apparatus according to claim 26, wherein the message packet comprises a Domain Name System (DNS) request packet.

31. Apparatus according to claim 26, wherein the message packet comprises a PING request.

32. Apparatus according to claim 25, wherein the data packet is one of a sequence of data packets having respective values of the field, and wherein the guard device is adapted to determine a rate of validating the values for the sequence of data packets, and to validate the value if the rate is no more than a predetermined maximum.

33. Apparatus according to claim 22, wherein the guard device is adapted to compare the value of the field to a plurality of reference values, which are determined by decrementing a plurality of respective initial values of the field by the same number of hops.

34. Apparatus according to claim 22, wherein the guard device is adapted to determine the reference value based on the value of the field in a protocol handshake exchanged between the source and destination addresses.

35. Apparatus according to claim 34, wherein the protocol handshake comprises a Transport Control Protocol (TCP) three-way handshake.

36. Apparatus according to any of claims 21-35, wherein the data packet comprises an Internet Protocol (IP) packet, and wherein the value of the field comprises a Time-To-Live (TTL) of the IP packet.

37. Apparatus according to any of claims 21-35, wherein the guard device is adapted to intercept the data packet prior to the packet's reaching the destination address, and to deliver the packet to the destination address depending on the assessed authenticity of the packet.

38. Apparatus according to claim 37, wherein the guard device is adapted to discard the packet if the packet is assessed as inauthentic.

39. Apparatus according to claim 37, wherein the guard device is adapted to deliver the packet to the destination address with a reduced level of priority if the packet is assessed as inauthentic.

40. Apparatus according to claim 37, wherein the guard device is adapted to perform a further check to verify the authenticity after assessing the authenticity of the source address responsive to the value and before conveying the packet to the destination address.

41. A computer software product for authenticating packet communication traffic, the product comprising a computer-readable medium in which program instructions are stored, which instructions, when read by a computer, cause the computer to receive a data packet sent over a network from a source address to a destination address, to read from the packet a value of a field that is indicative of a number of hops traversed by the packet since having been sent from the source address, and to assess authenticity of the source address responsive to the value.

42. A product according to claim 41, wherein the instructions cause the computer to read from a memory a reference value associated with the source address, and to assess the authenticity of the source address by comparing the value of the field to the reference value.

43. A product according to claim 42, wherein the instructions cause the computer to determine whether the value matches the reference value to within a predefined tolerance.

44. A product according to claim 42, wherein the instructions cause the computer to compare the value to an aggregate reference value associated with a subnet to which the source address belongs.

45. A product according to claim 42, wherein the instructions cause the computer, if there is no reference value associated with the source address in the memory that matches the value of the field, to validate the value of the field for use as the reference value.

46. A product according to claim 45, wherein the value of the field that is read from the packet comprises a first value of the field, and wherein the instructions cause the computer to validate the first value by sending a message packet over the network to the source address, receiving a response packet over the network from the source address in response to the message packet, reading a second value of the field from the response packet, and comparing the first and second values of the field.

47. A product according to claim 46, wherein the instructions cause the computer to encode the first value of the field in the message packet, so that the encoded first value is returned in the response packet, and to read the first value from the response packet.

48. A product according to claim 47, wherein the instructions cause the computer to insert a cookie encoding the first value in the message packet, and to authenticate the response packet based on the cookie.

49. A product according to claim 46, wherein the message packet comprises a Transport Control Protocol (TCP) SYN packet.

50. A product according to claim 46, wherein the message packet comprises a Domain Name System (DNS) request packet.

51. A product according to claim 46, wherein the message packet comprises a PING request.

52. A product according to claim 45, wherein the data packet is one of a sequence of data packets having respective values of the field, and wherein the instructions cause the computer to determine a rate of validating the values for the sequence of data packets, and to validate the value if the rate is no more than a predetermined maximum.

53. A product according to claim 42, wherein the instructions cause the computer to compare the value of the field to a plurality of reference values, which are determined by decrementing a plurality of respective initial values of the field by the same number of hops.

54. A product according to claim 42, wherein the instructions cause the computer to determine the reference value based on the value of the field in a protocol handshake exchanged between the source and destination addresses.

55. A product according to claim 54, wherein the protocol handshake comprises a Transport Control Protocol (TCP) three-way handshake.

56. A product according to any of claims 41-55, wherein the data packet comprises an Internet Protocol (IP) packet, and wherein the value of the field comprises a Time-To-Live (TTL) of the IP packet.

57. A product according to any of claims 41-55, wherein the instructions cause the computer to intercept the data packet prior to the packet's reaching the destination address, and to deliver the packet to the destination address depending on the assessed authenticity of the packet.

58. A product according to claim 57, wherein the instructions cause the computer to discard the packet if the packet is assessed as inauthentic.

59. A product according to claim 57, wherein the instructions cause the computer to deliver the packet to the destination address with a reduced level of priority if the packet is assessed as inauthentic.

60. A product according to claim 57, wherein the instructions cause the computer to perform a further check to verify the authenticity after assessing the authenticity of the source address responsive to the value and before conveying the packet to the destination address.

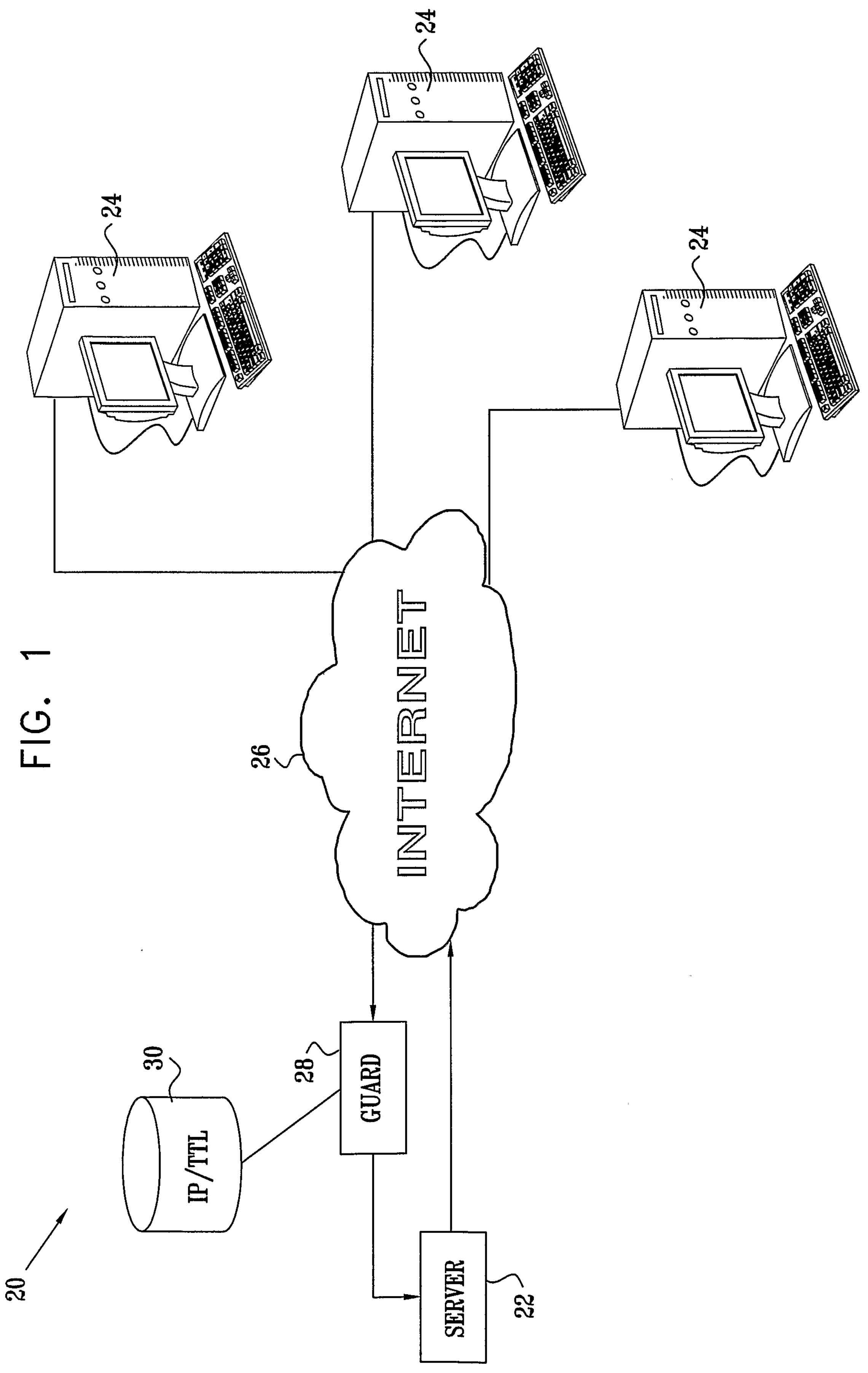


FIG. 1

FIG. 2

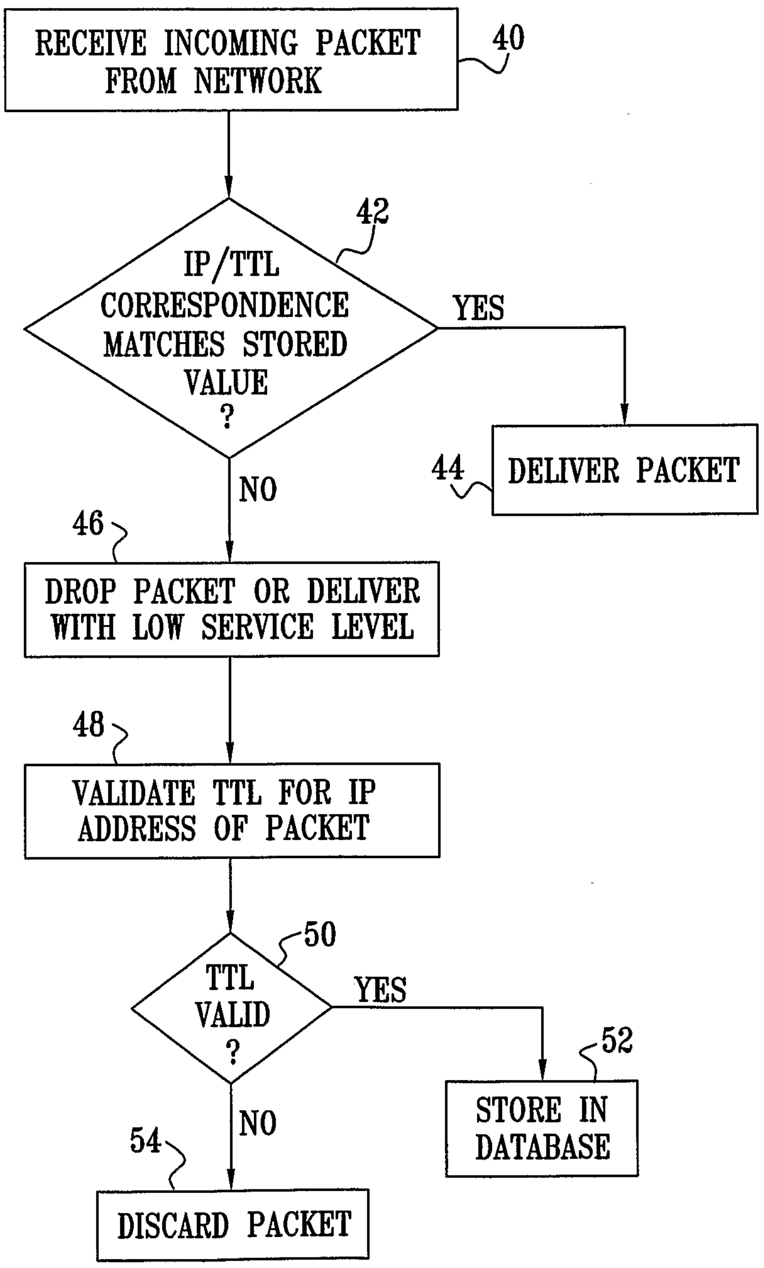
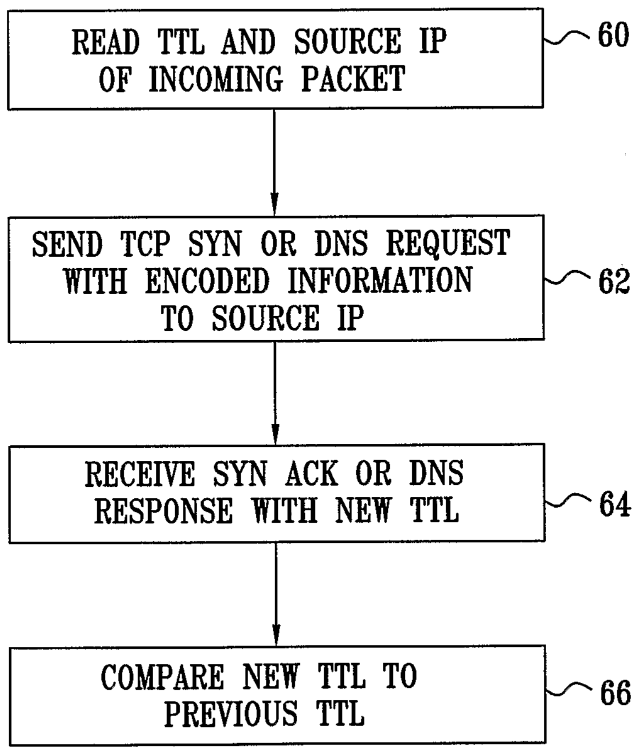

RECEIVE INCOMING PACKET FROM NETWORK
40
IP/TTL CORRESPONDENCE MATCHES STORED VALUE ?
42
YES
DELIVER PACKET
44
NO
46
DROP PACKET OR DELIVER WITH LOW SERVICE LEVEL
48
VALIDATE TTL FOR IP ADDRESS OF PACKET
50
TTL VALID ?
YES
52
STORE IN DATABASE
NO
54
DISCARD PACKET

FIG. 3


READ TTL AND SOURCE IP OF INCOMING PACKET
60
SEND TCP SYN OR DNS REQUEST WITH ENCODED INFORMATION TO SOURCE IP
62
RECEIVE SYN ACK OR DNS RESPONSE WITH NEW TTL
64
COMPARE NEW TTL TO PREVIOUS TTL
66

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL02/00714

**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(7) :G06F 15/173
US CL :709/225
According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/200, 223; 711/150; 713/160, 201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

US PTO BRS: EAST
search terms: authenticating packet, source, destination address, hop, TTL, IP packet, denial-of-service

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | US 6,055,236 A (NESSETT et al.) 25 April 2000, col. 5, lines 1-62; cols. 24-26, lines 29-45 | 1-60 |
| Y | US 6,185,680 B1 (SHIMBO et al.) 06 February 2001, Figs. 28-33; col. 3, line 32 - col. 9, line 38; col. 33, line 7 - col. 37, line 39 | 1-60 |
| X, E | US 6,502,135 B1 (MUNGER et al.) 31 December 2002, col. 2, line 65 - col. 6, line 10; col. 33, line 46 - col. 41, line 22. | 1-60 |

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

| | | | |
|---|---|---|---|
| * | Special categories of cited documents: | "T" | later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "A" | document defining the general state of the art which is not considered to be of particular relevance | | |
| "E" | earlier document published on or after the international filing date | "X" | document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "L" | document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) | "Y" | document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| "O" | document referring to an oral disclosure, use, exhibition or other means | | |
| "P" | document published prior to the international filing date but later than the priority date claimed | "&" | document member of the same patent family |

| | |
|---|---|
| Date of the actual completion of the international search<br>24 JANUARY 2003 | Date of mailing of the international search report<br>14 FEB 2003 |
| Name and mailing address of the ISA/US<br>Commissioner of Patents and Trademarks<br>Box PCT<br>Washington, D.C. 20231 | Authorized officer<br>for<br>MARK POWELL James R. Matthews |
| Facsimile No. (703) 305-3230 | Telephone No. (703) 305-9703 |