A message passing protocol allows two clients to establish a connection even when the clients are behind different NAT devices such as NAT firewalls. Beneficially, the protocol does not require that either client has knowledge of where the other client is located (e.g., behind the same NAT device or behind a different NAT device). When two clients want to establish a connection, the clients exchange identifying information with each other by passing the information through a rendezvous server. Based on the identifying information, each client determines and sends a plurality of synchronization packets to a number of different predicted addresses. When synchronization packets reach the actual addresses of both devices, a connection can be established between the clients.
FIG. 2
FIG. 3
NETWORK FIREWALL AND NAT TRAVERSAL FOR TCP AND RELATED PROTOCOLS

RELATED APPLICATIONS

[0001] This application claims priority from U.S. provisional application No. 61/311,141 entitled “High Performance Peer-To-Peer Assisted Live Content Delivery System and Method” filed on Mar. 5, 2010, the content of which is incorporated by reference herein in its entirety.

BACKGROUND

[0002] 1. Field of the Invention
[0003] The invention relates generally to the field of computer networking and more particularly to traversing NAT devices and firewalls.

[0004] 2. Description of the Related Arts
[0005] In many computer network applications (e.g., peer-to-peer networking), it is desirable for two or more clients to establish direct connections with each other without requiring all information to pass through a centralized server. To connect to another network node, a client generally sends a message to a recipient node requesting a connection. However, if the intended recipient node is behind a firewall or other network address translation (NAT) device, this connection request may be blocked. This is because a NAT device may be configured to only allow messages to reach an intended recipient when the message is in response to communication initiated by the recipient. Unsolicited communications are generally dropped. Alternatively, a NAT device may map a given internal IP address and port to a different external IP address and/or port for a multitude of reasons, making it similarly difficult to reach a given node.

[0006] When two nodes are both behind different NAT devices, neither node is able to initiate the connection to the other node because the incoming connection requests are blocked by the NAT device on the receiving end. As a result, the nodes will be unable to connect to each other. This creates a challenging problem in applications such as peer-to-peer networking, or other network applications that are not specifically peer-to-peer, but nevertheless utilize direct communication between two devices on a network.

SUMMARY

[0007] A system, method, and computer-readable storage medium enables clients to connect to each other even when the clients are behind different NAT devices (e.g., NAT firewalls).

[0008] In one embodiment, a first client receives identifying information for the remote client and generates, based on the identifying information for the remote client, a plurality of predicted addresses for the remote client. The first client sends a plurality of synchronization packets addressed to each of the plurality of predicted addresses for the remote client. The first client then receives a reply synchronization packet addressed from the remote client’s actual current address, which is one of the predicted addresses. The first client then establishes a connection for communicating with the remote client using the actual address.

[0009] In one embodiment, an initiating client requests a connection to a remote client by sending a connection request to a rendezvous server, which forwards the request to the remote client together with an identity of a selected rendezvous helper. The remote client receives the request forwarded by the rendezvous server and in response, sends its private address to the selected rendezvous helper. The initiating client receives identifying information for the remote client from the rendezvous server. For example, in one embodiment the identifying information comprises a most recently observed public address (e.g., public IP address and public TCP port) of the remote client and the private address (e.g., a private IP address and private TCP port) of the remote client. The initiating client may also receive a request for its current private address from the rendezvous server and in response, the initiating client provides identifying information (e.g., its private address) to a selected rendezvous helper. The rendezvous server forwards the initiating client’s identifying information (e.g., a private address and its most recently observed public address) to the remote client. Thus, the remote client receives the identifying information (e.g., a private address and most recently observed public address) of the initiating client. Based on the identifying information for the remote client, the initiating client generates a plurality of predicted addresses for the remote client. The initiating client then sends a plurality of synchronization packets addressed to each of the plurality of predicted addresses. Similarly, the remote client generates a plurality of predicted addresses for the initiating client and sends a plurality of synchronization packets addressed to each of the plurality of predicted addresses. Assuming one of the initiating client’s predictions is correct, the initiating client will receive the synchronization packet addressed from the remote client’s actual current address. The initiating client can then send an acknowledgement to the remote client’s actual current address, thus allowing the remote client to learn the initiating client’s actual current address. Thus, the initiating client and remote client each learn each other’s addresses and can establish a connection with each other. This technique is particularly beneficial in, for example, peer-to-peer networking where clients may need to connect with each other for data sharing. Furthermore, this technique is applicable to other network applications that are not specifically for peer-to-peer sharing, but nevertheless utilize communication between clients that may be behind NAT devices.

[0010] The features and advantages described in the specification are not all inclusive and, in particular, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The teachings of the embodiments of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings.

[0012] FIG. 1 is a system diagram illustrating an example of a network environment in accordance with an embodiment of the present invention.

[0013] FIG. 2 illustrates a message passing protocol for traversing NAT devices and establishing a connection between clients in accordance with an embodiment of the invention.
FIG. 3 is a system diagram illustrating an example of a computing device in accordance with an embodiment of the invention.

DETAILED DESCRIPTION

Reference in the specification to “one embodiment” or to “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least one embodiment of the invention. The appearances of the phrase “in one embodiment” or “an embodiment” in various places in the specification are not necessarily all referring to the same embodiment.

Some portions of the detailed description that follows are presented in terms of algorithms and protocols and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps (instructions) leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic or optical signals capable of being stored, transferred, combined, compared and otherwise manipulated. It is convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. Furthermore, it is also convenient at times, to refer to certain arrangements of steps requiring physical manipulations or transformation of physical quantities or representations of physical quantities as modules or code devices, without loss of generality.

However, all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as “processing” or “computing” or “calculating” or “determining” or “displaying” or the like, refer to the action and processes of a computer system, or similar electronic computing device (such as a specific computing machine), that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that the process steps and instructions of the present invention could be embodied in software, firmware or hardware, and when embodied in software, could be downloaded to reside on and be operated from different platforms used by a variety of operating systems. The invention can also be in a computer program product in which computer executable instructions for carrying out the processes described herein are stored on a computer-readable storage medium, and can be executed by a processor on a computing system.

The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the purposes, e.g., a specific computer, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer-readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Memory can include any of the above and/or other devices that can store information/data/programs. Furthermore, the computers referred to in the specification may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the method steps. The structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references below to specific languages are provided for disclosure of enablement and best mode of the present invention.

In addition, the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention.

Overview

A method for traversing firewalls and network address translation (NAT) devices enables computing devices (“clients” or “nodes” or “peers”) to connect to each other. FIG. 1 illustrates an example of a network architecture 100. A number of clients 110 are coupled to a first local area network (LAN) 120 that is behind a first NAT device 130 with respect to a wide area network (WAN) 140, such as, for example, the Internet. A number of clients 170 are also coupled to a second local area network 160 that is behind a second NAT device 150 with respect to the WAN 140. A rendezvous server 180 and a number of rendezvous helpers 190 are coupled to the WAN 140 in front of the NAT devices 130, 150.

To facilitate routing of data packets to and from different clients 110 on the LAN 120, each client 110 is assigned a private IP address that uniquely identifies the client 110 on the LAN 120. Each client 110 may host a number of different TCP ports on which data may be communicated. A data packet transmitted from a client 110 specifies both an IP address of the transmitting client 110 (a “source IP address”) and a port for the communication (a “source port”). To be routed to the appropriate recipient, a transmitted data packet also specifies an IP address of the intended recipient (a “destination IP address”) and a communication port for the intended recipient (a “destination port”). The communication path between a particular source IP address/source port and a destination IP address/destination port is referred to herein as a “connection.” A private IP address/port tuple is referred to herein as a “private address.” Thus, a client 120 may host a plurality of private addresses each corresponding to a differ-
ent connection. Routing of data packets to and from clients 170 on LAN 160 follow similar operation.

The NAT device 130 multiplexes outbound traffic from the LAN 120 and provides the traffic to the WAN 150 as if it were coming from a single IP address (the “NAT IP address” or “public IP address”). In certain cases, a NAT device may use multiple public IP addresses, however functionally such a device is otherwise the same as described here. The NAT device 130 maintains a NAT table that provides a mapping which enables the NAT device 130 to forward an IP packet destined for a specific public address (IP address and port) to a specific private address (IP address and port) on the LAN 120. Depending on the specific internal implementation of the NAT device 130, this mapping may involve some subset of the source IP address, destination IP address, source port, and destination port fields of the IP packet. Some NAT devices additionally specifically track connections and are able to use information beyond what is carried in a given IP packet. When receiving an outbound packet from a client 110, the NAT device 130 changes the original source address (IP address and port) in the outbound packet from the private address to the corresponding public NAT address (IP address and port) such that a returning packet will be appropriately forwarded to the correct destination (client 110), and the NAT device 130 creates the relevant mapping entry in the NAT table as needed. Thus, devices outside the NAT device 130 are unable to see the private address associated with the data packet, but instead only see the public IP address and public port (collectively referred to herein as a “public address”) that corresponds to the private address in the NAT table.

Incoming packets from outside the NAT device 130 all have as their destination IP address one of the public IP addresses of the NAT device 130, and may have different port numbers depending on the intended destination client/port. When the NAT device 130 receives an incoming packet from the WAN 140, it looks up the appropriate destination address (IP address and port) in its map and forwards the packet to the appropriate client/port on the LAN 120. If the destination cannot be determined from the NAT table, the packet will generally be rejected. NAT device 150 follows similar operation.

The process of assigning IP addresses and ports is dynamic. Thus, a client’s private IP address may change over time, and the ports it uses for communication may change. Furthermore, the mapping between a private address and a public address in the NAT table may change over time. Frequently, each mapping in the NAT table has a time-out associated with it. If no communication is received or transmitted on a particular client/port after a period of time, the mapping is removed from the NAT table. The mapping is added back to the NAT table only when the client initiates a new outbound packet, and the NAT device may assign a completely different public port than the one previously used.

When configured as firewalls protect their respective clients 110, 170 by preventing external data from reaching a client behind the firewall unless the connection is initiated by the client. Thus, under a conventional scenario, a connection request sent from a client 110 to a client 170 will be blocked by NAT device 150, and similarly, a connection request sent from a client 170 to a client 110 will be blocked by NAT device 130.

In order to avoid this problem and achieve connections between clients 110, 170, the clients 110, 170 may utilize the help of the rendezvous server 180 and one more rendezvous helpers 190. The rendezvous server 180 and rendezvous helpers 190 are connected directly to the WAN 140 and their addresses are therefore always public (i.e., not behind a protective firewall). Thus, clients 110 behind NAT device 130 and client 170 behind NAT device 150 can all communicate with the rendezvous server 180 and rendezvous helpers 190. Furthermore, once a client initiates communication with the rendezvous server 180 or a rendezvous helper 190, a hole is opened in the NAT device 130 or 150 which allows the rendezvous server 180 or a rendezvous helper 190 to send return communications. Thus, two-way communication can be established between a client and the rendezvous server 180 and between a client and a rendezvous helper 190. Using the process described below, the rendezvous server 180 and rendezvous helpers 190 can assist the clients 110, 170 in traversing the NAT devices 130, 150 and establishing direct connections with each other.

Rendezvous Server Operation

The rendezvous server 180 is a specialized computing device that facilitates connections between clients 110, 170. In one embodiment, the rendezvous server 180 is owned and operated by a network manager. The rendezvous helpers 190 comprise computing devices that act as sub-servers to assist the rendezvous server 180 in facilitating the connections between clients 110, 170 and may also be hosted by a network manager.

Upon startup, the rendezvous server 180 listens on a TCP port P for connection attempts from new clients. The rendezvous server 180 also allocates a number of rendezvous helpers 190 to listen on public ports P1, P2, P3, and so on, possibly on a variety of different machines. Each time a new client connects, the rendezvous server 180 stores information about the connection in a table for later use and begins processing packets for the connecting client. For example, the rendezvous server 180 may store a “public name” identifying the connecting client and may store additional information about current or prior connections established by the clients. As used herein, a “name” is an identifier for a given client. The public name used by the rendezvous server 180 may correspond to the initial public address of a client seen by the rendezvous server 180 when the client first contacts the rendezvous server 180. Alternatively, the rendezvous server may store a different public name identifying client. The operation of the rendezvous server 180 and rendezvous helpers 190 are described in further detail below with reference to FIG. 3.

Protocol for Nat Device Traversal

A message passing protocol allows two clients to establish a connection with each other even when the clients are behind different NAT devices such as NAT firewalls. Furthermore, the protocol allows clients to establish a connection when both clients are behind the same NAT device. Beneficially, the protocol does not require that either client has knowledge of where the other client is located (e.g., behind the same NAT device or behind a different NAT device).

In one embodiment applicable to peer-to-peer networking applications, a client decides whether and/or when to establish a connection to another a client according to a peer-to-peer network membership management protocol. In one such protocol, each “node” (i.e., a connected client) on the network establishes a neighboring relationship (i.e., a con-
nection) with one or more other nodes on the network. Neigh-
bor nodes can therefore directly share data, and non-
neighboring nodes can indirectly share data via hops between
neighboring nodes. In one embodiment, the membership
management protocol uses a gossip-based protocol with
some level of centralized control from the server 180. In one
embodiment, the decision of whether or not to connect to
another node is made according to a probabilistic algorithm.
In this way, a new client effectively gains connections to a
random subset of nodes on the network. Furthermore, the
membership management protocol allows nodes to peri-
dodically re-subscribe to the network, thus periodically rebal-
anc ing network graph.

[0033] Other examples of membership management proto-
cols may also be compatible with the NAT device traversal
protocol. Thus, prior to attempting a connection using the
NAT traversal technique described below, a client is first
made aware of at least a subset of other nodes on the network
(e.g., via knowledge of a public name or other identifying
information), and have determined to initiate a connection
attempt in accordance with the membership management
protocol.

[0034] The techniques described below may also be appli-
cable to networking applications that are not specifically for
peer-to-peer sharing. In these applications, a client may seek
a connection to another client based on some other criteria
external to the NAT traversal process described below. Thus,
the description that follows assumes that a client seeking a
connection with another client has been made aware of the
other client's presence through some external mechanism and
has learned some identifying information for the client (e.g.,
a public name or other identifying information).

[0035] FIG. 2 illustrates an example message passing
sequence between two clients A and B, a rendezvous server
180, and rendezvous helpers R' and R" when client A and B
want to establish a connection in accordance with the
network membership management protocol. The message passing
protocol is not limited to the particular sequence of messages
illustrated in FIG. 2. Thus, although the messages are
described as occurring in a particular order for clarity of
description, the particular sequence is only exemplary and the
messages may be sent in different orders depending on the
particular network configuration and circumstances.

[0036] In the illustrated example, clients A and B each
initially register with the rendezvous server 180. This limita-
tion is enforced implicitly because a client might not initially
know its own public name. In one embodiment, client A sends
a REGISTRATION REQUEST message 0 to the rendezvous
helper 180 requesting a public name. Similarly, client B sends
a REGISTRATION REQUEST message 0' to the rendezvous
server 180. In response, the rendezvous server 180 sends a
REGISTRATION RESPONSE message 1 to client A which
provides client A's public name Ap. Similarly, the rendezvous
server 180 sends a REGISTRATION RESPONSE message 1'
to client B providing client B's public name Bp.

[0037] In one embodiment, the public names Ap and Bp
provided by the rendezvous server 180 are merely the public
addresses assigned by their respective NAT devices and
observed by the rendezvous server 180 when it receives the
REGISTRATION REQUEST messages 0, 0'. Alternatively,
the public names Ap and Bp could comprise different identi-
fiers assigned by the rendezvous server 180. In this case, the
rendezvous server 180 maintains a table mapping the clients'
actual public addresses assigned by the NAT device to the
public names Ap and Bp used by the rendezvous server 180.

[0038] Optionally, clients A and/or B may specify a public
name as an argument in the REGISTRATION REQUEST
messages 0, 0'. The specified public name could be, for
example, an old public name previously used by the request-
ing client. This allows the rendezvous server 180 to help
connect clients A and B without affecting existing connec-
tions clients A and B may have to other clients. Because, in
one embodiment, the operation does not use a client's public
name for anything other than as a way to refer to it, the
practice of allowing a client to claim an arbitrary name is
fairly safe. A risk is that one client might wish to claim the
same name as another client. The rendezvous server 180 can
address these naming conflicts by asking one of the clients to
initiate the request again.

[0039] A connection between a client A and a client B may
be initiated in one of two ways. In a first scenario, messages 0
and 1 (described below) are omitted, and the connection
process begins when an initiating client (Client A in the
illustrated example) initiates the process by sending a CON-
NECTION REQUEST message 2. In another scenario, a
client (client B in the illustrated example) may initiate another
client (client A in the illustrated example) to initiate the
connection by sending an optional REVERSE CONNECTION
REQUEST message 0. Thus, messages 0 and 1 are
optional and depend on whether client B induces client A to
initiate the connection request or whether client A initiates the
request on its own. Under either scenario, client A is referred
to herein as the "initiating client" and client B is referred to as
the "remote client." However, any client can operate as either
an initiating client or a remote client depending on the cir-
cumstances. For example, at a different time, a third client
(e.g., a client C) may initiate a connection to client A. Thus,
in this process client A would instead act as the remote client,
with client C acting as the initiating client.

[0040] Under the first scenario described above, client B
sends the optional REVERSE CONNECTION REQUEST
message 0 to the rendezvous server 180 specifying client B's
public name Bp as the source and client A's public name Ap as
the destination. The rendezvous server 180 sends a
REVERSE CONNECTION REQUEST message 1 to client
A specifying client B's public name Bp as the source and
client A's public name Ap as the destination.

[0041] The REVERSE CONNECTION REQUEST mes-
sages 0, 1 allow client B to induce the NAT traversal almost
as if client A had initiated the traversal. This feature enhances
the success rate of the NAT traversal when one of the two
clients involved is behind a so-called "busy firewall." A busy
firewall is a firewall which may alter its port allocation
scheme for a given client over time—for instance, because it
has used the ports it would have given to that client for some
other device. Thus, issuing a REVERSE CONNECTION
REQUEST message 0 helps penetrate otherwise impen-
etrable firewalls. A client chooses whether or not to use a
reversed connection each time it begins a connection attempt.

[0042] In one embodiment, to identify if a client should use
a reversed connection or not, a technique is utilized in which
each client maintains a "doForward" token. When a traversal
attempt fails due to a mismatch of ports in the SYN exchange
described below (which in one embodiment manifests as a
timeout), the client flips the value of the doForward token
(i.e., if true, changes to false; if false, changes to true). The
client then uses the token to decide whether to initiate travers-
als in a forward direction (when doForward=true) or reversed direction (when doForward=false). Clients that are behind a busy firewall generally need forward traversals to succeed and will tend to fail when doForward is false. Thus, in the next attempt, doForward be true and the traversal attempt is more likely to succeed. Clients behind a firewall that is not busy will tend to succeed more often if doForward is false. Thus, an unsuccessful attempt when doForward is true, will be more likely to succeed in the next attempt when doForward is false. In one embodiment, the server could also track the success and failures of the connection attempts, and instruct a client whether or not to reverse its connection attempts.

[0043] If a client A wants to connect to client B (either in response to a reverse connection attempt by client B, or because client A decides to initiate a forward connection attempt on its own), client A sends a CONNECTION REQUEST message to the rendezvous server specifying client A’s public name Ap as the source and node B’s public name Bp as the destination. The rendezvous server forwards a CONNECTION REQUEST message to client B indicating to client B that client A would like to connect to it. This message specifies client A’s public name Ap as the source and client B’s public name Bp as the destination.

[0044] Client A also sends a GET CURRENT ADDR REQUEST message to the rendezvous server specifying client A’s public name Ap as the requester and client B’s public name Bp as the host. This message requests client B’s identifying information (e.g., most recently observed public address and private address). When the rendezvous server receives a GET CURRENT ADDR REQUEST message from a client, the server randomly chooses one of a number of rendezvous helpers 190 and forwards the public address (i.e., IP address and TCP port) of the selected rendezvous helper 190 to the requesting client. For example, in one embodiment, the rendezvous server sends a GET CURRENT ADDR COMMAND message to client B designating the selected rendezvous helper 190 (in the illustrated embodiment, the selected rendezvous helper has a public address R*). This message specifies client A’s public name Ap as the requester, node B’s public name Bp as the host, and the selected rendezvous helper’s public address R*. In an alternative embodiment, messages 2 and 2 can be combined into a single message.

[0045] Client B similarly sends a GET CURRENT ADDR REQUEST message to the rendezvous server requesting client A’s current address. This message specifies client B’s public name Bp as the requester and client A’s public name Ap (e.g., public IP address and public TCP port) as the host. The rendezvous server selects a rendezvous helper 190 (in this case, a rendezvous helper having a public address R*). The rendezvous server then sends a GET CURRENT ADDR COMMAND message to client A specifying client B’s public name Bp as the requester, node A’s public name Ap as the host, and the rendezvous helper’s public address R*.

[0046] Client B also initiates a new connection to the rendezvous helper R* specified in the GET CURRENT ADDR COMMAND message. Here, client B sends a NEW CURRENT ADDR REQUEST message to rendezvous helper R* providing client B’s identifying information. For example, in one embodiment, NEW CURRENT ADDR REQUEST message specifies client A’s identifying information. For example, in one embodiment, NEW CURRENT ADDR REQUEST message specifies client A’s public name Ap as the requester, and node B’s public name Bp as the host, and provides node B’s private address Bs. Because client B is unlikely to have communicated with this specific rendezvous helper R* recently, the rendezvous helper R* is able to observe a newly crafted public address (IP address and TCP port) for client B. Having an up-to-date reading on the ports being allocated by the NAT device of a given client allows for a better informed estimate of what port future SYN packets will be assigned to.

[0047] Upon receiving the NEW CURRENT ADDR message, the rendezvous helper R* forwards the message to the rendezvous server. Then, the rendezvous server forwards this identifying information to client A in a GET CURRENT ADDR RESPONSE message. This message specifies client B’s public name Bp as the host, node B’s private address Bs, and node B’s most recently observed public address Bp0 as seen by rendezvous helper R* upon receiving the NEW CURRENT ADDR message.

[0048] In one embodiment, immediately before client B sends message Ap, a handshake packet exchange is performed between the rendezvous helper R* and client B, with client B initiating the sequence. For example, in one embodiment, a traditional TCP handshake sequence is used in which the initiator (client B) sends a SYN packet, the recipient (R*) responds with a SYN/ACK packet, and the initiator (client B) responds back with an ACK packet. This handshake establishes a TCP connection between the two nodes and allows the setup of the various details needed to initialize the TCP state machines on either side. In some instances it may be possible to send data in conjunction with some of these packets. However, for clarity of description, it is assumed that the handshake happens before any data packets are exchanged.

[0049] Client A similarly sends a NEW CURRENT ADDR message to its specified rendezvous helper R*. As this connection is a new TCP connection, client A may also initiate a TCP handshake with R* (e.g., a traditional SYN-SYN/ACK-ACK sequence) prior to sending message 6. The NEW CURRENT ADDR message specifies client B’s public name Bp as the requestor, client A’s public name Ap as the host, and client A’s private address As. The rendezvous helper R* then forwards the message to the rendezvous server. Then, the rendezvous server forwards the information to client B in a GET CURRENT ADDR RESPONSE message. This message specifies client A’s public name Ap as the host, client A’s private address As, and client A’s most recently observed public address Ap0. In an alternative embodiment, messages 5 and 5 can be combined into a single message.

[0050] When client A has received messages 5 and 5, it has enough information to begin its half of the simultaneous open protocol. Based on the identifying information in message 5, client A determines a plurality of predicted addresses for the actual address of client B. Client A then sends out a plurality of synchronization packets to these addresses in SYN packets 6, 7, 8, 9. For example, in one embodiment packet 6 specifies Bp0+0, packet 7 specifies Bp0+1, packet 8 specifies Bp0+2, and packet 9 specifies Bp0+3. Generally, the predicted addresses are selected as those that client B will most likely have as its current public address (e.g., as currently assigned by its NAT device). For example, the most recently observed public address Bp0 provides a good prediction because for most NAT devices, there is a good probability that B’s public address has not changed since Bp0 was observed. Bp0+1 and Bp0+2 are effective because many NAT devices allocate new port assignments by a small increment (which may or may not be constant) from the previous port. These addresses are also effective for so-called busy firewalls which assign ports similarly. The last synchronization packet in 9 specifying client

Sep. 8, 2011
B’s private address may be an effective because client A and client B might be behind the same firewall. In this case, client B’s reported private address is the most likely current address where client A can find client B. In one embodiment, the plurality of synchronization packets are sent in a single pass without waiting for feedback between packets. Thus, in one embodiment, the plurality of synchronization packets is sent in a fixed manner without trial and adjustments. Although, the SYN packet 6, 7, 8 may be blocked by a NAT device on client B’s side sending the packets will cause the NAT device on client A’s side to temporarily open a hole that will allow return communication to client A from any of the IP address/ports specified in the synchronization packets.

In one embodiment, immediately before client A sends 6, a handshake sequence (e.g., a SYN/SYN/ACK/ACK sequence) is initiated by client A with the rendezvous helper R7, so that A can eventually send message 6 to R7 over a new TCP connection. The SYN from this handshake and message 6 are sent by A back to back.

Client B waits for a period of time (e.g., between 10 ms and 5 seconds) after receiving the GET CURRENT ADDR RESPONSE message 7. Then client B sends a plurality of synchronization packets to predicted addresses for client A in SYN packets 8, 9, 10, 11. The waiting period gives client A time to send out all of its SYN packets 6, 7, 8, 9, (thereby opening the relevant holes in its NAT device) before client B responds with its corresponding set of SYN packets 8, 9, 10, 11. For example, in one embodiment packet 8 specifies Ap0, packet 9 specified Ap0+1, packet 10 specifies Ap0+2, and packet 11 specifies As+0. As with client A, in one embodiment, the plurality of synchronization packets sent by client B are all sent in a single pass without waiting for feedback between packets. Thus, in one embodiment, the plurality of synchronization packets is sent in a fixed manner without trial and adjustments. For some types of NAT devices this greatly improves the chances of traversal success. For example, some NAT devices may send “end of connection” packets (e.g., RST and FIN packets in traditional TCP protocol) in response to unsolicited SYN packets that are intended to terminate a connection. The waiting period ensures that any RST or FIN packets sent in response to unknown SYN packets arrive before the subsequent SYN packets.

Assuming both client A and client B are able to predict correctly in one of their SYN packets, holes will be opened on both sides to allow client A and client B to complete the three-way TCP handshake (e.g., the SYN/SYN/ACK/ACK sequence). In some cases, an extra SYN or SYN/ACK is present instead of the traditional three-way handshake, but most implementations of TCP allow this behavior. The end result is a working TCP connection between client A and client B, allowing them to communicate in spite of the possible existence of NAT devices between them.

The operation sequence related to 6, 7, 8, 9, 10, and 11 can be customized based on the requirement of the application scenario. Generally, the plurality of packets includes a SYN packet sent to the most recently observed public addresses (e.g., in messages 6 and 8) and one or more SYN packets addressed to addresses that are a fixed increment above or below the most recently observed public address (e.g., using small integral offsets). Specific options for customization include, but are not limited to: additional SYN messages (not shown) specifying different or additional addresses may be sent; 9 and/or 11 can be omitted; 9 and/or 11 can be conditionally omitted if the private address matches the most recently observed public address; 8 and/or 10 can be omitted; 8, 9, 10, and 11 can be omitted; the order of 6, 7, 8 and 9 can be re-arranged; the order of 8, 9, 10, 11 can be re-arranged; any one or more of 6, 7, 8, and 9 can be omitted; any one or more of 8, 9, 10, and 11 can be omitted; the SYN messages from client A and client B may be asymmetric; rendezvous server 180 can keep track of observed behavior of client A and client B and use the history record to omit one of 6, 7, and 8. Further, rendezvous server 180 can keep track of observed behavior of client A and client B and determine which of 6, 7, or 8 will be sent out first, and which one will be sent out second (if a second one is needed), and which one will be sent out third (if a third one is needed). If the application scenario requires, the implementation may also change the number of SYN packets sent in steps 6-9 (sent by client A), and the number of SYN packets sent in steps 8-11 (sent by client B) correspondingly. Different port additions may also be used here instead of those illustrated in the described example.

In one embodiment, client B acquires a MUX/EX prior to sending message 4 and maintains it until after it sends message 11. The MUX/EX protects other synchronous traversal attempts from initiating connections in the span of time between when client B sends message 4 until after client B sends message 11. Similarly, client A may acquire a MUX/EX prior to sending message 6 and releases it after sending message 9, thus preventing other synchronous traversal attempts from initiating connections with client A during the time period between message 6 and message 9. The MUX/EX operation helps to reduce the likelihood that clients A and B will open other connections during the traversal process. For many NAT devices, this prevents interference with the port allocation estimate used in the traversal.

In one embodiment, the rendezvous server 180 provides fault protection by informing a client if it transmits an undeliverable message. For example, if a client sends a message specifying a destination name unknown to the rendezvous server 180, the rendezvous server 180 replies with an ERR NO SUCH HOST message indicating that the destination name is unknown. Furthermore, in one embodiment, if a client sends a message specifying a destination name that cannot currently receive a message (e.g., because it is busy processing other requests), the rendezvous server 180 replies with an ERR HOST BUSY message indicating that the intended recipient is busy. In one embodiment, the rendezvous server 180 may send an ERR CONNECTION FAILED HOST BUSY message if a connection attempt is unsuccessful.

In one embodiment, NAT/TID messages are exchanged bi-directionally between clients once a connection is established. A NAT/TID message specifies the name of the sending client. These messages ensure that the client on each side is connected to the intended client.

Although FIG. 2 illustrates one exemplary message passing scenario, under different conditions, the messages may be sent in a different order than that illustrated. For example, in one embodiment, messages may occur in any order under the constraints illustrated in the table below in which the notation a, b—c means that message c must follow messages a and b, and the notation a—b—c means message c must follow message b and message b must follow message a:
This traversal process described above beneficially works without the need to determine the type of the NAT device involved. Rather, the same generic process can be applied directly by the clients interested in connecting to each other. Thus, the traversal process works with a variety of NAT devices and firewalls including, for example, full cone devices, address restricted cone devices, port restricted cone devices, and symmetric devices. Furthermore, the traversal process works with NAT devices supporting a variety of different port allocation schemes including, for example, independent allocation, incremental allocation using a small number, busy firewalls and NAT devices, and random allocation scenarios.

This traversal mechanism works using a short, compact and generic packet sequence and processing logic. Only a small number of TCP SYN requests are required for its effectiveness. It is compatible with existing systems and network deployment and does not require modification or any change to existing computer systems that support TCP/IP protocols. Furthermore, the traversal mechanism works within the common system resource limit regarding network and operating system resources usage. At the same time, the traversal mechanism allows multiple TCP connections to be established from one computer to another systems rapidly in a short time frame, without excessive delay between different connections attempts and slowing down other applications utilizing the network at the same time.

Additional Alternative Embodiments

In one embodiment, a pair of connections may be established between two clients in specific application scenarios for communication; one connection is used exclusively for control messages and the other is used exclusively for data. For TCP, each connection is strictly in-order. Using a pair of connections in this manner allows control packets to be received out-of-order with respect to the data packets, preventing the larger (slower) data from delaying the more time-sensitive control packets.

In one embodiment, traversal attempts are separated into incoming attempts (initiated by a remote device) and outgoing attempts (initiated locally). This technique works well for operating systems that place a limit on the number of “half-opened” TCP connections, i.e., a TCP connection which has sent a SYN packet but not yet received a SYN-ACK from the remote side. This value is typically 10, system-wide, after which point subsequent attempts to open new connections are queued up until the ten half-open connections either succeed or fail. Examples of operating systems having such a limitation include those provided by MICROSOFT CORPORATION of Redmond, Wash.

As a result of the limitation described above, one embodiment limits the number of concurrent traversal attempts. Each traversal attempt may open up to five separate connections after message 4 (one to the rendezvous helper, one to the private address, and three to variants of the public address). As a result, a client may safely attempt only two traversals at a time under the worst case scenario. Different implementations may reduce the number of SYNs sent in step 6', 7', 8', 9', 8, 9, 10, and 11 in order to increase the number of traversal attempts at a time by one client. In one embodiment, to prevent livelock, a client allows only one outgoing (locally initiated) traversal attempt and one incoming (requested remotely) traversal attempt on any given node at any time. This allows a client to aggressively pursue traversal attempts (an outgoing attempt) without preventing other clients from trying to reach it (an incoming attempt).

In one embodiment, two connection attempts can be performed with certain overlap. The second connection does not need to wait for the first one to complete entirely before being initiated. For example, after the first traversal attempt has completed step 9', the second one is free to start, thus allowing multiple outstanding traversal attempts at the same time. This enables fast connection establishment with a large number of clients within a short time period. For example, suppose client A wants to connect to client B and client C. Right after client A has completed step 9' with client B, client A is free to initiate a connection to client C. The two connection attempts from client A to client B and from client A to client C do not interfere with each other. This also allows control and data connections be established back-to-back, next to each other, as long as the first one completes its sequence up to message 9' before the second one is initiated from message 2, without excessive delay and wait between two connection attempts.

Message and Data Types

The list below illustrates examples of types of messages that various devices can send and the basic data types used in the message, in accordance with one embodiment of the message passing protocol:

**Basic Data Types:**

- **n16:** 16-bit unsigned integer in network byte order
- **n32:** 32-bit unsigned integer in network byte order
- **addr:** 48-bit structure `{n32 ip_address, n16 tcp_port}`
- **name:** 48-bit structure `{n32 ip_address, n16 tcp_port}`

**Message Types:**

- **NATT ID:** `{n16 messageId, addr myAddr}`
- **REGISTRATION REQUEST:** `{n16 messageId, addr clientPublicAddr}`
- **REGISTRATION RESPONSE:** `{n16 messageId, addr clientPublicAddr}`
- **CONNECTION REQUEST:** `{n16 messageId, name source name dest}`
- **REVERSE CONNECTION REQUEST:** `{n16 messageId, name source name dest}`
- **ERR NO SUCH HOST:** `{n16 messageId name host}`
- **ERR HOST BUSY:** `{n16 messageId name host}`
- **ERR CONNECTION FAILED HOST BUSY:** `−{n16 messageId, name source, name dest}`
System Architecture

[0083] FIG. 3 is a high-level block diagram illustrating an example of a computing device 300 that could act as a client (e.g., a client 110 or a client 170) or a rendezvous helper 190. Illustrated are at least one processor 302, and input controller 304, a network adapter 306, a graphics adapter 308, a storage device 310, and a memory 312. Other embodiments of the computer 300 may have different architectures with additional or different components. In some embodiments, one or more of the illustrated components are omitted.

[0084] The storage device 310 is a computer-readable storage medium such as a hard drive, compact disk read-only memory (CD-ROM), DVD, or a solid-state memory device. The memory 312 store instructions and data used by the processor 302. The processing device 326 is a mouse, trackball, or other type of pointing device, and is used in combination with the keyboard 324 to input data into the computer system 300. The graphics adapter 308 outputs images and other information for display by the display device 322. The network adapter 306 couples the computer system 300 to a network 330.

[0085] The computer 300 is adapted to execute computer program instructions for providing functionality described herein. In one embodiment, program program instructions are stored on the storage device 310, loaded into the memory 312, and executed by the processor 302 to carry out the processes described herein.

[0086] The types of computers 300 operating on the peer-to-peer network can vary substantially. For example, a client comprising a personal computer (PC) may include most or all of the components illustrated in FIG. 3. Another client may comprise a mobile computing device (e.g., a cell phone) which typically has limited processing power, a small display 322, and might lack a pointing device 326. A rendezvous server 180 or rendezvous helper 190 may comprise multiple processors 302 and/or a graphical computing device 326. In other embodiments, the server, clients, or helpers could comprises other types of electronic device such as, for example, a personal digital assistant (PDA), a mobile telephone, a pager, a television “set-top box,” etc.

[0087] The network 330 enables communications among the entities connected to it (e.g., the nodes and the server). In one embodiment, the network 330 is the Internet and uses standard communications technologies and/or protocols. Thus, the network 330 can include links using a variety of known technologies, protocols, and data formats. In addition, all or some of links can be encrypted using conventional encryption technologies. In another embodiment, the entities use custom and/or dedicated data communications technologies.

[0088] Upon reading this disclosure, those of skill in the art will appreciate still additional alternative designs for membership management having the features described herein. Thus, while particular embodiments and applications of the present invention have been illustrated and described, it is to be understood that the invention is not limited to the specific construction and components disclosed herein and that various modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus of the present invention disclosed herein without departing from the spirit and scope of the invention as defined in the appended claims.

What is claimed:

1. A method for establishing a connection between a first client and a remote client, the method comprising:
   - receiving, by the first client, identifying information for the remote client;
   - generating, based on the identifying information for the remote client, a plurality of predicted addresses for the remote client;
   - sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses for the remote client;
   - receiving a reply synchronization packet addressed from the remote client’s actual current address, wherein the remote client’s actual current address comprises one of the predicted addresses; and
   - establishing a connection for communicating with the remote client using the actual address.

2. The method of claim 1, wherein the identifying information for the remote client comprises a most recently observed public address of the remote client and wherein receiving the identifying information comprises:
   - receiving the most recently observed public address of the remote client from a rendezvous server that previously received communication specifying the most recently observed public address of the remote client.

3. The method of claim 2, wherein sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses comprises:
   - sending a first synchronization packet to the most recently observed public address of the remote client; and
   - sending a second synchronization packet to an second address that is a fixed increment above or below the most recently observed public address.

4. The method of claim 2, wherein receiving the identifying information further comprises receiving the private address of the remote client from the rendezvous server, and wherein sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses further comprises sending a third synchronization packet to a private address of the remote client.

5. The method of claim 1, further comprising:
   - prior to receiving the identifying information, sending a registration request message to a rendezvous server requesting registration of the first client; and
   - responding to receiving a registration request message, receiving a registration response message from the rendezvous server specifying a public name for the first client.

6. The method of claim 5, wherein the registration request message specifies a desired public name previously used by
the first client, and wherein the public name sent to the first client specifies the desired public name.

7. The method of claim 1, further comprising:
   prior to receiving the identifying information, sending to a rendezvous server, a connection request message requesting a connection to the remote client; and
   sending to the rendezvous server, a get current address request message requesting the current public address of the remote client.

8. The method of claim 7, further comprising:
   prior to sending the connection request message, receiving from the rendezvous server, a reverse connection request message specifying the remote client, wherein receipt of the reverse connection request message induces the first client to request the connection to the remote client;
   responsive to the token being in the first state, attempting to initiate the connection to the remote client in a forward direction by sending a connection request message from the first client requesting a connection to the remote client;
   responsive to the token being in the second state, attempting to initiate the connection to the remote client in a reverse direction by sending a reverse connection request message from the first client that induces the remote client to initiate the connection request to the first client; and
   responsive to a connection attempt failing, flipping the state of the token.

9. The method of claim 1, further comprising:
   receiving from the rendezvous server, a get current address command message requesting that the first client provide its current private address to a selected rendezvous helper; and
   transmitting, by the first client to the selected rendezvous helper, a new current address message specifying the first client’s private address.

10. The method of claim 9, further comprising:
    following transmission of the new current address message, acquiring a mutex that prevents synchronous connection attempts from devices other than the remote client; and
    releasing the mutex after sending the plurality of synchronization packets.

11. The method of claim 1, further comprising:
    concurrently with attempting to establish the connection with the remote client, attempting to establish an inbound connection with a third client, wherein the inbound connection is initiated by the third client.

12. The method of claim 1, wherein the first client is behind a first NAT device and wherein the remote client is behind a second NAT device different than the first NAT device.

13. A method for establishing a connection between an initiating client and a remote client, the method comprising:
    receiving from a rendezvous server, a connection request specifying a public name of the initiating client seeking the connection and identifying a selected rendezvous helper;
    in response to receiving the connection request, providing the remote client’s private address to the selected rendezvous helper;
    receiving from the rendezvous server, a most recently observed public address of the initiating client and a private address of the initiating client;
    generating, based on the most recently observed public address for the initiating client, a plurality of predicted addresses for the initiating client;
    sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses for the initiating client and the initiating client’s private address; and
    receiving acknowledgment of receipt of one of the plurality of synchronization packets, the acknowledgment addressed from the initiating client’s actual current address; and
    establishing a connection with the initiating client using the initiating client’s actual current address.

14. The method of claim 14, further comprising:
    after providing the remote client’s private address to the selected rendezvous helper and prior to sending the plurality of synchronization packets, waiting for a period of time long enough to allow the initiating client to send a plurality of synchronization packets to the remote client.

15. The method of claim 14, further comprising:
    prior to receiving the connection request from to the initiating client, sending a registration request message to the rendezvous server requesting registration of the remote client; and
    responsive to sending the registration request message, receiving a registration response from the rendezvous server specifying the remote client’s public name.

16. The method of claim 14, wherein receiving the connection request comprises:
    receiving from the rendezvous server, a connection request message requesting the connection to the remote client; and
    receiving from the rendezvous server, a get current address command message requesting that the remote client provide its current private address to the selected rendezvous helper.

17. The method of claim 16, further comprising:
    sending to the rendezvous server, a get current address request message requesting the current public address of the initiating client.

18. The method of claim 14, further comprising:
    prior to receiving the connection request from the rendezvous server specifying the initiating client, transmitting to the rendezvous server, a reverse connection request message specifying the initiating client, wherein the reverse connection request message induces the initiating client to request the connection to the remote client.

19. The method of claim 14, further comprising:
    receiving, by the first client, identifying information for the remote client;
    generating, based on the identifying information for the remote client, a plurality of predicted information for the remote client;
    sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses for the remote client;
21. The computer-readable storage medium of claim 20, the instructions when executed further causing the processor to perform steps including:

receiving a reply synchronization packet addressed from the remote client's actual current address, wherein the remote client's actual current address comprises one of the predicted addresses; and

establishing a connection for communicating with the remote client using the actual address.

in response to receiving the incoming connection request, providing the first client's private address to the selected rendezvous helper;

receiving from the rendezvous server, a most recently observed public address of the third client and a private address of the third client;

generating, based on the most recently observed public address for the third client, a plurality of predicted addresses for the third client;

sending a plurality of synchronization packets addressed to each of the plurality of predicted addresses for the third client and the third client's private address; and

receiving acknowledgement of receipt of one of the plurality of synchronization packet, the acknowledgement addressed from the third client's actual current address;

and

establishing a connection with the third client using the third client's actual current address.

* * * * *

* * * * *

* * * * *