An Introduction to Computer Networks (week 4)

Stanford Univ CS144 Fall 2012

Contents

Articles

References

Article Licenses

Names and addresses

Address Resolution Protocol

Address Resolution Protocol (**ARP**) is a telecommunications protocol used for resolution of network layer addresses into link layer addresses, a critical function in multiple-access networks. ARP was defined by RFC 826 in 1982.[1] It is Internet Standard STD 37. It is also the name of the program for manipulating these addresses in most operating systems.

ARP has been implemented in many combinations of network and overlaying internetwork technologies, such as IPv4, Chaosnet, DECnet and Xerox PARC Universal Packet (PUP) using IEEE 802 standards, FDDI, X.25, Frame Relay and Asynchronous Transfer Mode (ATM), IPv4 over IEEE 802.3 and IEEE 802.11 being the most common cases.

In Internet Protocol Version 6 (IPv6) networks, the functionality of ARP is provided by the Neighbor Discovery Protocol (NDP).

Operating scope

The Address Resolution Protocol is a request and reply protocol that runs encapsulated by the line protocol. It is communicated within the boundaries of a single network, never routed across internetwork nodes. This property places ARP into the Link Layer of the Internet Protocol Suite, while in the Open Systems Interconnection (OSI) model, it is often described as residing between Layers 2 and 3, being encapsulated by Layer 2 protocols. However, ARP was not developed in the OSI framework.

Packet structure

The Address Resolution Protocol uses a simple message format that contains one address resolution request or response. The size of the ARP message depends on the upper layer and lower layer address sizes, which are given by the type of networking protocol (usually IPv4) in use and the type of hardware or virtual link layer that the upper layer protocol is running on. The message header specifies these types, as well as the size of addresses of each. The message header is completed with the operation code for request (1) and reply (2). The payload of the packet consists of four addresses, the hardware and protocol address of the sender and receiver hosts.

The principal packet structure of ARP packets is shown in the following table which illustrates the case of IPv4 networks running on Ethernet. In this scenario, the packet has 48-bit fields for the sender hardware address (SHA) and target hardware address (THA), and 32-bit fields for the corresponding sender and target protocol addresses (SPA and TPA). Thus, the ARP packet size in this case is 28 bytes. The EtherType for ARP is 0x806.

Internet Protocol (IPv4) over Ethernet ARP packet		
bit offset	**0 – 7**	**8 – 15**
0	Hardware type (HTYPE)	
16	Protocol type (PTYPE)	
32	Hardware address length (HLEN)	Protocol address length (PLEN)
48	Operation (OPER)	
64	Sender hardware address (SHA) (first 16 bits)	
80	(next 16 bits)	
96	(last 16 bits)	
112	Sender protocol address (SPA) (first 16 bits)	
128	(last 16 bits)	
144	Target hardware address (THA) (first 16 bits)	
160	(next 16 bits)	
176	(last 16 bits)	
192	Target protocol address (TPA) (first 16 bits)	
208	(last 16 bits)	

Hardware type (HTYPE)

This field specifies the network protocol type. Example: Ethernet is 1.

Protocol type (PTYPE)

This field specifies the internetwork protocol for which the ARP request is intended. For IPv4, this has the value 0x0800. The permitted PTYPE values share a numbering space with those for EtherType.[2][3][4]

Hardware length (HLEN)

Length (in octets) of a hardware address. Ethernet addresses size is 6.

Protocol length (PLEN)

Length (in octets) of addresses used in the upper layer protocol. (The upper layer protocol specified in PTYPE.) IPv4 address size is 4.

Operation

Specifies the operation that the sender is performing: 1 for request, 2 for reply.

Sender hardware address (SHA)

media address of the sender.

Sender protocol address (SPA)

internetwork address of the sender.

Target hardware address (THA)

media address of the intended receiver. This field is ignored in requests.

Target protocol address (TPA)

internetwork address of the intended receiver.

ARP protocol parameter values have been standardized and are maintained by the Internet Assigned Numbers Authority (IANA).[5]

Example

For example, the computers *Matterhorn* and *Washington* are in an office, connected to each other on the office local area network by Ethernet cables and network switches, with no intervening gateways or routers. Matterhorn wants to send a packet to Washington. Through other means, it determines that Washington's IP address is 192.168.0.55. In order to send the message, it also needs to know Washington's MAC address. First, Matterhorn uses a cached ARP table to look up 192.168.0.55 for any existing records of Washington's MAC address (00:eb:24:b2:05:ac). If the MAC address is found, it sends the IP packet on the link layer to address 00:eb:24:b2:05:ac via the local network cabling. If the cache did not produce a result for 192.168.0.55, Matterhorn has to send a broadcast ARP message (destination FF:FF:FF:FF:FF:FF) requesting an answer for 192.168.0.55. Washington responds with its MAC address (00:eb:24:b2:05:ac). Washington may insert an entry for Matterhorn into its own ARP table for future use. The response information is cached in Matterhorn's ARP table and the message can now be sent.

ARP probe

An **ARP probe** is an ARP request constructed with an all-zero *sender IP address*. The term is used in the *IPv4 Address Conflict Detection* specification (RFC 5227). Before beginning to use an IPv4 address (whether received from manual configuration, DHCP, or some other means), a host implementing this specification must test to see if the address is already in use, by broadcasting ARP probe packets.

ARP announcements

ARP may also be used as a simple announcement protocol. This is useful for updating other hosts' mapping of a hardware address when the sender's IP address or MAC address has changed. Such an announcement, also called a *gratuitous ARP* message, is usually broadcast as an ARP request containing the sender's protocol address (SPA) in the target field (TPA=SPA), with the target hardware address (THA) set to zero. An alternative is to broadcast an ARP reply with the sender's hardware and protocol addresses (SHA and SPA) duplicated in the target fields (TPA=SPA, THA=SHA).

An ARP announcement is not intended to solicit a reply; instead it updates any cached entries in the ARP tables of other hosts that receive the packet. The operation code may indicate a request or a reply because the ARP standard specifies that the opcode is only processed after the ARP table has been updated from the address fields.[6][7][8]

Many operating systems perform gratuitous ARP during startup. That helps to resolve problems which would otherwise occur if, for example, a network card was recently changed (changing the IP-address-to-MAC-address mapping) and other hosts still have the old mapping in their ARP caches.

Gratuitous ARP is also used by some interface drivers to provide load balancing for incoming traffic. In a team of network cards, it is used to announce a different MAC address within the team that should receive incoming packets.

ARP announcements can be used to defend link-local IP addresses in the Zeroconf protocol (RFC 3927), and for IP address takeover within high-availability clusters.

ARP mediation

ARP mediation refers to the process of resolving Layer 2 addresses when different resolution protocols are used on multiple connected circuits, e.g., ATM on one end and Ethernet on the others. A proposed standard for ARP mediation is currently in draft status under the Internet Engineering Task Force.[9]

Inverse ARP and Reverse ARP

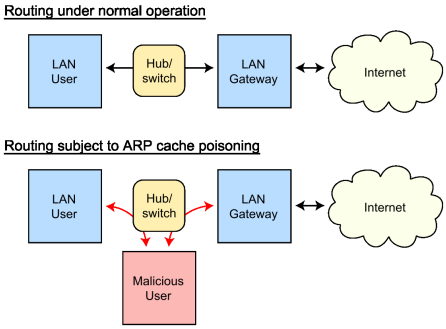
Inverse Address Resolution Protocol (Inverse ARP or InARP) is used to obtain Network Layer addresses (for example, IP addresses) of other nodes from Data Link Layer (Layer 2) addresses. It is primarily used in Frame Relay (DLCI) and ATM networks, in which Layer 2 addresses of virtual circuits are sometimes obtained from Layer 2 signaling, and the corresponding Layer 3 addresses must be available before those virtual circuits can be used.[10]

Since ARP translates Layer 3 addresses to Layer 2 addresses, InARP may be described as its inverse. In addition, InARP is implemented as a protocol extension to ARP: it uses the same packet format as ARP, but different operation codes.

The Reverse Address Resolution Protocol (Reverse ARP or RARP), like InARP, translates Layer 2 addresses to Layer 3 addresses. However, in InARP the requesting station queries the Layer 3 address of another node, whereas RARP is used to obtain the Layer 3 address of the requesting station itself for address configuration purposes. RARP is obsolete; it was replaced by BOOTP, which was later superseded by the Dynamic Host Configuration Protocol (DHCP).[11]

ARP spoofing and Proxy ARP

Because ARP does not provide methods for authenticating ARP replies on a network, ARP replies can come from systems other than the one with the required Layer 2 address. An ARP *proxy* is a system which answers the ARP request on behalf of another system for which it will forward traffic, normally as part of network design such as dialup internet service. By contrast in ARP *spoofing*, where the spoofer answers the ARP requests with the aim of interception. A malicious user may leverage ARP spoofing to perform a man-in-the-middle or denial-of-service attack on other users on the network. Various software exists to both detect and perform ARP spoofing attacks, though ARP itself does not provide any methods of protection from such attacks.[12]



A successful ARP spoofing attack allows an attacker to perform a man-in-the-middle attack.

Alternatives to ARP

Each computer maintains its own table of the mapping from Layer 3 addresses (e.g. IP addresses) to Layer 2 addresses (e.g. ethernet MAC addresses). In a modern computer this is maintained almost entirely by ARP packets on the local network and it thus often called the 'ARP cache' as opposed to 'Layer 2 address table'. In older computers, where broadcast packets were considered an expensive resource, other methods were used to maintain this table, such as static configuration files,[13] or centrally maintained lists. Since at least the 1980s[14] networked computers have had a command called *arp* for interrogating or manipulating this table, and practically all modern personal computers have a variant of this.[15][16][17][18]

ARP Stuffing

Embedded systems such as networked cameras[19] and networked power distribution devices,[20] which lack a user interface, can use so-called *ARP stuffing* to make an initial network connection, although this is a misnomer as there is no ARP *protocol* involved. This is a solution to an issue in network management of consumer devices, specifically the allocation of IP addresses of ethernet devices where 1) the user doesn't have the ability to control DHCP or similar address allocation protocols, 2) the device doesn't have a user interface to configure it, and 3) the user's computer can't communicate with it because it has no suitable IP address.

The solution adopted is as follows: the user's computer has an IP address *stuffed* manually into its address table (normally with the *arp* command with the MAC address taken from a label on the device) and then sends special packets to the device, typically a ping packet with a non-default size. The device then adopts this IP address, and the user then communicates with it by telnet or web protocols to complete the configuration. Such devices typically have a method to disable this process once the device is operating normally, as it is open to Denial of Service attack.

References

- Neighboring Subsystem by Rami Rosen [21]

[1] David C. Plummer (1982-11). "RFC 826, An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware" (http://tools.ietf.org/html/rfc826). Internet Engineering Task Force, Network Working Group. .

[2] IANA ARP - "Protocol Type" (http://www.iana.org/assignments/arp-parameters/)

[3] IANA - Ethertype values (http://www.iana.org/assignments/ethernet-numbers)

[4] RFC 5342

[5] "IANA ARP parameter assignments" (http://www.iana.org/assignments/arp-parameters/). IANA. 2009-04-24. .

[6] Gratuitous ARP in DHCP vs. IPv4 ACD Draft (http://www1.ietf.org/mail-archive/web/dhcwg/current/msg03797.html)

[7] RFC 2002 Section 4.6 (http://tools.ietf.org/html/rfc2002#section-4.6)

[8] RFC 2131 DHCP – Last lines of Section 4.4.1 (http://tools.ietf.org/html/rfc2131#section-4.4.1)

[9] Himanshu Shah, et. al. (2011-04-03). "ARP Mediation for IP Interworking of Layer 2 VPN" (http://tools.ietf.org/html/draft-ietf-l2vpn-arp-mediation-16). Internet Engineering Task Force. .

[10] T. Bradley, et. al. (1998-09). "RFC 2390 - Inverse Address Resolution Protocol" (http://tools.ietf.org/html/rfc2390). Internet Engineering Task Force. .

[11] Finlayson, Mann, Mogul, Theimer (1984-06). "RFC 903 - A Reverse Address Resolution Protocol" (http://tools.ietf.org/html/rfc903). Internet Engineering Task Force. .

[12] Steve Gibson (2005-12-11). "ARP Cache Poisoning" (http://www.grc.com/nat/arp.htm). GRC. .

[13] Sun Microsystems. "SunOS manual page for ethers(5) file" (http://www.freebsd.org/cgi/man.cgi?query=ethers&sektion=5&apropos=0&manpath=SunOS+4.1.3). . Retrieved 2011-09-28.

[14] University of California, Berkeley. "BSD manual page for arp(8C) command" (http://www.freebsd.org/cgi/man.cgi?query=arp&apropos=0&sektion=0&manpath=2.10+BSD&arch=default&format=html). . Retrieved 2011-09-28.

[15] Canonical. "Ubuntu manual page for arp(8) command" (http://manpages.ubuntu.com/manpages/lucid/man8/arp.8.html). . Retrieved 2011-09-28.

[16] Apple Computer. "Mac OSX manual page for arp(8) command" (http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man8/arp.8.html). . Retrieved 2011-09-28.

[17] Microsoft. "Windows help for arp command" (http://technet.microsoft.com/en-us/library/cc786759(WS.10).aspx). . Retrieved 2011-09-28.

[18] Cisco. "Cisco reference for 'show ip arp' command" (http://www.cisco.com/en/US/docs/ios/12_3/ipaddr/command/reference/ip1_s1g.html#wp1079902). . Retrieved 2011-09-28.

[19] Axis Communication. "Axis P13 Network Camera Series Installation Guide" (http://www.axis.com/files/manuals/ig_p13Series_38731_en_1006.pdf). . Retrieved 2011-09-28.

[20] American Power Corporation. "Switched Rack Power Distribution Unit Installation and Quick Start Manual" (http://www.apcmedia.com/salestools/ASTE-6Z6K56_R0_EN.pdf). . Retrieved 2011-09-28.

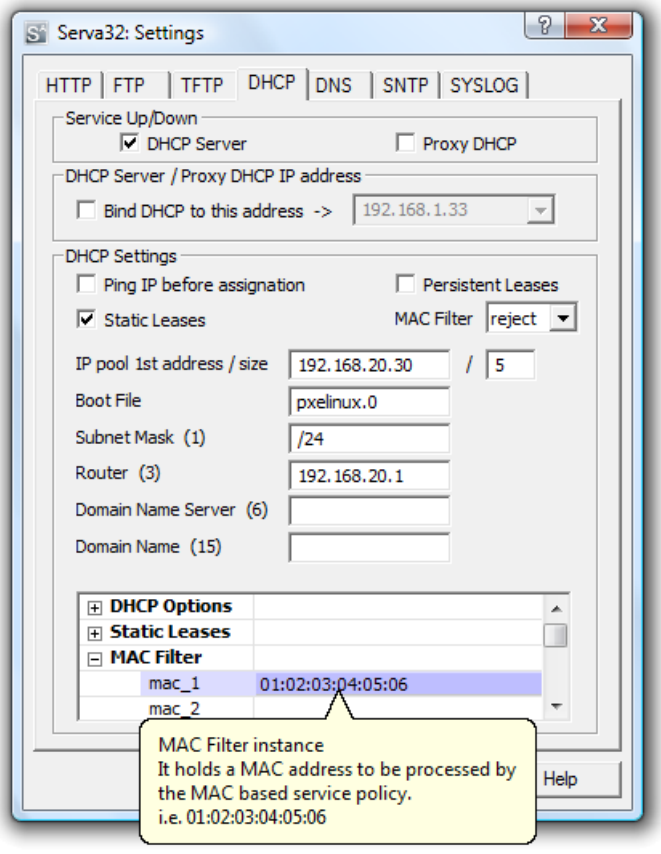
[21] https://www.linuxfoundation.org/collaborate/workgroups/networking/neighboringsubsystem

External links

- RFC 826 - Ethernet Address Resolution Protocol, Internet Standard STD 37.
- RFC 903 - Reverse Address Resolution Protocol, Internet Standard STD 38.
- RFC 2390 - Inverse Address Resolution Protocol, draft standard
- RFC 5227 - IPv4 Address Conflict Detection, proposed standard
- ArpON home page (http://arpon.sourceforge.net)
- ARP Sequence Diagram (pdf) (http://www.eventhelix.com/RealtimeMantra/Networking/Arp.pdf)
- Gratuitous ARP (http://wiki.wireshark.org/Gratuitous_ARP)
- Free ARP tools with source code (French) (http://www.authsecu.com/arpflood/)
- ARP-SK ARP traffic generation tools (http://sid.rstack.org/arp-sk/)
- Sample Capture file from WireSharkWiki (http://wiki.wireshark.org/SampleCaptures#head-2fb4a82886c1d8c722134b44461e22e5f7f54b32)

Dynamic Host Configuration Protocol

The **Dynamic Host Configuration Protocol** (**DHCP**) is a network protocol that is used to configure network devices so that they can communicate on an IP network. A DHCP client uses the DHCP protocol to acquire configuration information, such as an IP address, a default route and one or more DNS server addresses from a DHCP server. The DHCP client then uses this information to configure its host. Once the configuration process is complete, the host is able to communicate on the internet.



A DHCP Server settings tab

The DHCP server maintains a database of available IP addresses and configuration information. When it receives a request from a client, the DHCP server determines the network to which the DHCP client is connected, and then allocates an IP address or prefix that is appropriate for the client, and sends configuration information appropriate for that client.

Because the DHCP protocol must work correctly even before DHCP clients have been configured, the DHCP server and DHCP client must be connected to the same network link. In larger networks, this is not practical. On such networks, each network link contains one or more DHCP relay agents. These DHCP relay agents receive messages from DHCP clients and forward them to DHCP servers. DHCP servers send responses back to the relay agent, and the relay agent then sends these responses to the DHCP client on the local network link.

DHCP servers typically grant IP addresses to clients only for a limited interval. DHCP clients are responsible for renewing their IP address before that interval has expired, and must stop using the address once the interval has

expired, if they have not been able to renew it.

DHCP is used for IPv4 and IPv6. While both versions serve much the same purpose, the details of the protocol for IPv4 and IPv6 are sufficiently different that they may be considered separate protocols.[1]

Hosts that do not use DHCP for address configuration may still use it to obtain other configuration information. Alternatively, IPv6 hosts may use stateless address autoconfiguration. IPv4 hosts may use link-local addressing to achieve limited local connectivity.

History

DHCP was first defined as a standards track protocol in RFC 1531 in October 1993, as an extension to the Bootstrap Protocol (BOOTP). The motivation for extending BOOTP was that BOOTP required manual intervention to add configuration information for each client, and did not provide a mechanism for reclaiming disused IP addresses.

Many worked to clarify the protocol as it gained popularity, and in 1997 RFC 2131 was released, and remains as of 2011 the standard for IPv4 networks. DHCPv6 is documented in RFC 3315. RFC 3633 added a DHCPv6 mechanism for prefix delegation. DHCPv6 was further extended to provide configuration information to clients configured using stateless address autoconfiguration in RFC 3736.

The BOOTP protocol itself was first defined in RFC 951 as a replacement for the Reverse Address Resolution Protocol RARP. The primary motivation for replacing RARP with BOOTP was that RARP was a data link layer protocol. This made implementation difficult on many server platforms, and required that a server be present on each individual network link. BOOTP introduced the innovation of a *relay agent*, which allowed the forwarding of BOOTP packets off the local network using standard IP routing, thus one central BOOTP server could serve hosts on many IP subnets.[2]

Technical overview

Dynamic Host Configuration Protocol automates network-parameter assignment to network devices from one or more DHCP servers. Even in small networks, DHCP is useful because it makes it easy to add new machines to the network.

When a DHCP-configured client (a computer or any other network-aware device) connects to a network, the DHCP client sends a broadcast query requesting necessary information to a DHCP server. The DHCP server manages a pool of IP addresses and information about client configuration parameters such as default gateway, domain name, the name servers, other servers such as time servers, and so forth. On receiving a valid request, the server assigns the computer an IP address, a lease (length of time the allocation is valid), and other IP configuration parameters, such as the subnet mask and the default gateway. The query is typically initiated immediately after booting, and must complete before the client can initiate IP-based communication with other hosts. Upon disconnecting, the IP address is returned to the pool for use by another computer. This way, many other computers can use the same IP address within minutes of each other.

Depending on implementation, the DHCP server may have three methods of allocating IP-addresses:

- *dynamic allocation*: A network administrator assigns a range of IP addresses to DHCP, and each client computer on the LAN is configured to request an IP address from the DHCP server during network initialization. The request-and-grant process uses a lease concept with a controllable time period, allowing the DHCP server to reclaim (and then reallocate) IP addresses that are not renewed.
- *automatic allocation*: The DHCP server permanently assigns a free IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had.

- *static allocation*: The DHCP server allocates an IP address based on a table with MAC address/IP address pairs, which are manually filled in (perhaps by a network administrator). Only clients with a MAC address listed in this table will be allocated an IP address. This feature, which is not supported by all DHCP servers, is variously called *Static DHCP Assignment* by DD-WRT, *fixed-address* by the dhcpd documentation, *Address Reservation* by Netgear, *DHCP reservation* or *Static DHCP* by Cisco and Linksys, and *IP reservation* or *MAC/IP binding* by various other router manufacturers.

Technical details

DHCP uses the same two ports assigned by IANA for BOOTP: destination UDP port 67 for sending data to the server, and UDP port 68 for data to the client. DHCP communications are connectionless in nature.

DHCP operations fall into four basic phases: IP discovery, IP lease offer, IP request, and IP lease acknowledgement. These points are often abbreviated as DORA (Discovery, Offer, Request, Acknowledgement).

DHCP clients and servers on the same subnet communicate via UDP broadcasts, initially. If the client and server are on different subnets, a DHCP Helper or DHCP Relay Agent may be used. Clients requesting renewal of an existing lease may communicate directly via UDP unicast, since the client already has an established IP address at that point.

DHCP discovery

The client broadcasts messages on the physical subnet to discover available DHCP servers. Network administrators can configure a local router to forward DHCP packets to a DHCP server from a different subnet. This client-implementation creates a User Datagram Protocol (UDP) packet with the broadcast destination of 255.255.255.255 or the specific subnet broadcast address.

A DHCP client can also request its last-known IP address (in the example below, 192.168.1.100). If the client remains connected to a network for which this IP is valid, the server may grant the request. Otherwise, it depends whether the server is set up as authoritative or not. An authoritative server will deny the request, making the client ask for a new IP address immediately. A non-authoritative server simply ignores the request, leading to an implementation-dependent timeout for the client to give up on the request and ask for a new IP address.

DHCPDISCOVER

UDP Src=0.0.0.0 sPort=68 Dest=255.255.255.255 dPort=67			
OP	**HTYPE**	**HLEN**	**HOPS**
0x01	0x01	0x06	0x00
XID			
0x3903F326			
SECS		**FLAGS**	
0x0000		0x0000	
CIADDR (Client IP Address)			
0x00000000			
YIADDR (Your IP Address)			
0x00000000			
SIADDR (Server IP Address)			
0x00000000			
GIADDR (Gateway IP Address)			

0x00000000
CHADDR (Client Hardware Address)
0x00053C04
0x8D590000
0x00000000
0x00000000
192 octets of 0s, or overflow space for additional options. BOOTP legacy
Magic Cookie
0x63825363
DHCP Options
DHCP option 53: DHCP Discover
DHCP option 50: 192.168.1.100 requested
DHCP option 55: Parameter Request List: Request Subnet Mask (1), Router (3), Domain Name (15), Domain Name Server (6)

DHCP offer

When a DHCP server receives an IP lease request from a client, it reserves an IP address for the client and extends an IP lease offer by sending a DHCPOFFER message to the client. This message contains the client's MAC address, the IP address that the server is offering, the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

The server determines the configuration based on the client's hardware address as specified in the CHADDR (Client Hardware Address) field. Here the server, 192.168.1.1, specifies the IP address in the YIADDR (Your IP Address) field.

DHCPOFFER

UDP Src=192.168.1.1 sPort=67 Dest=255.255.255.255 dPort=68			
OP	**HTYPE**	**HLEN**	**HOPS**
0x02	0x01	0x06	0x00
0x00000000			
YIADDR (Your IP Address)			
0xC0A80164			
SIADDR (Server IP Address)			
0xC0A80101			
GIADDR (Gateway IP Address)			
0x00000000			
CHADDR (Client Hardware Address)			
0x00053C04			
0x8D590000			
0x00000000			
0x00000000			

192 octets of 0s. BOOTP legacy
Magic Cookie
0x63825363
DHCP Options
DHCP option 53: DHCP Offer
DHCP option 1: 255.255.255.0 subnet mask
DHCP option 3: 192.168.1.1 router
DHCP option 51: 86400s (1 day) IP lease time
DHCP option 54: 192.168.1.1 DHCP server
DHCP option 6: DNS servers 9.7.10.15, 9.7.10.16, 9.7.10.18

DHCP request

In response to the DHCP offer, the client replies with a DHCP request, multicast to the server, requesting the offered address. A client can receive DHCP offers from multiple servers, but it will accept only one DHCP offer. Based on the Transaction ID field in the request, servers are informed whose offer the client has accepted. When other DHCP servers receive this message, they withdraw any offers that they might have made to the client and return the offered address to the pool of available addresses. In some cases DHCP request message is broadcast, instead of being unicast to a particular DHCP server, because the DHCP client has still not received an IP address. Also, this way one message can let all other DHCP servers know that another server will be supplying the IP address without missing any of the servers with a series of unicast messages.

DHCPREQUEST

UDP Src=0.0.0.0 sPort=68 Dest=255.255.255.255 dPort=67			
OP	**HTYPE**	**HLEN**	**HOPS**
0x01	0x01	0x06	0x00
XID			
0x3903F326			
SECS		**FLAGS**	
0x0000		0x0000	
CIADDR (Client IP Address)			
0x00000000			
YIADDR (Your IP Address)			
0x00000000			
SIADDR (Server IP Address)			
0xC0A80101			
GIADDR (Gateway IP Address)			
0x00000000			
CHADDR (Client Hardware Address)			
0x00053C04			
0x8D590000			

0x00000000
0x00000000
192 octets of 0s. BOOTP legacy
Magic Cookie
0x63825363
DHCP Options
DHCP option 53: DHCP Request
DHCP option 50: 192.168.1.100 requested
DHCP option 54: 192.168.1.1 DHCP server.

DHCP acknowledgement

When the DHCP server receives the DHCPREQUEST message from the client, the configuration process enters its final phase. The acknowledgement phase involves sending a DHCPACK packet to the client. This packet includes the lease duration and any other configuration information that the client might have requested. At this point, the IP configuration process is completed.

The protocol expects the DHCP client to configure its network interface with the negotiated parameters.

DHCPACK

UDP Src=192.168.1.1 sPort=67 Dest=255.255.255.255 dPort=68				
OP	**HTYPE**	**HLEN**	**HOPS**	
0x02	0x01	0x06	0x00	
XID				
0x3903F326				
SECS			**FLAGS**	
0x0000		0x0000		
CIADDR (Client IP Address)				
0x00000000				
YIADDR (Your IP Address)				
0xC0A80164				
SIADDR (Server IP Address)				
0xC0A80101				
GIADDR (Gateway IP Address switched by relay)				
0x00000000				
CHADDR (Client Hardware Address)				
0x00053C04				
0x8D590000				
0x00000000				
0x00000000				
192 octets of 0s. BOOTP legacy				

Magic Cookie	
0x63825363	
DHCP Options	
DHCP option 53: DHCP ACK	
DHCP option 1: 255.255.255.0 subnet mask	
DHCP option 3: 192.168.1.1 router	
DHCP option 51: 86400s (1 day) IP lease time	
DHCP option 54: 192.168.1.1 DHCP server	
DHCP option 6: DNS servers 9.7.10.15, 9.7.10.16, 9.7.10.18	

After the client obtains an IP address, the client may use the Address Resolution Protocol (ARP) to prevent IP conflicts caused by overlapping address pools of DHCP servers.

DHCP information

A DHCP client may request more information than the server sent with the original DHCPOFFER. The client may also request repeat data for a particular application. For example, browsers use *DHCP Inform* to obtain web proxy settings via WPAD. Such queries do not cause the DHCP server to refresh the IP expiry time in its database.

DHCP releasing

The client sends a request to the DHCP server to release the DHCP information and the client deactivates its IP address. As client devices usually do not know when users may unplug them from the network, the protocol does not mandate the sending of *DHCP Release*.

Client configuration parameters in DHCP

A DHCP server can provide optional configuration parameters to the client. RFC 2132 describes the available DHCP options defined by Internet Assigned Numbers Authority (IANA) - DHCP and BOOTP PARAMETERS [3].

A DHCP client can select, manipulate and overwrite parameters provided by a DHCP server.[4]

DHCP options

The following tables list the available DHCP options, as stated in RFC2132.[5]

RFC1497 vendor extensions[6]

Code	Name	Length	Notes
0	Pad[7]	1 octet	Can be used to pad other options so that they are aligned to the word boundary
1	Subnet Mask[8]	4 octets	Must be sent after the router option (option 3) if both are included
2	Time Offset[9]	4 octets	
3	Router	multiples of 4 octets	Available routers, should be listed in order of preference
4	Time Server	multiples of 4 octets	Available time servers to synchronise with, should be listed in order of preference
5	Name Server	multiples of 4 octets	Available IEN116 name servers, should be listed in order of preference
6	Domain Name Server	multiples of 4 octets	Available DNS servers, should be listed in order of preference
7	Log Server	multiples of 4 octets	Available log servers, should be listed in order of preference.

8	Cookie Server	multiples of 4 octets	
9	LPR Server	multiples of 4 octets	
10	Impress Server	multiples of 4 octets	
11	Resource Location Server	multiples of 4 octets	
12	Host Name	minimum of 1 octet	
13	Boot File Size	2 octets	Length of the boot image in 4KiB blocks
14	Merit Dump File	minimum of 1 octet	Path where crash dumps should be stored
15	Domain Name	minimum of 1 octet	
16	Swap Server	4 octets	
17	Root Path	minimum of 1 octet	
18	Extensions Path	minimum of 1 octet	
255	End	1 octet	Used to mark the end of the vendor option field

IP Layer Parameters per Host[10]

Code	Name	Length	Notes
19	IP Forwarding Enable/Disable	1 octet	
20	Non-Local Source Routing Enable/Disable	1 octet	
21	Policy Filter	multiples of 8 octets	
22	Maximum Datagram Reassembly Size	2 octets	
23	Default IP Time-to-live	1 octet	
24	Path MTU Aging Timeout	4 octets	
25	Path MTU Plateau Table	multiples of 2 octets	

IP Layer Parameters per Interface[11]

Code	Name	Length	Notes
26	Interface MTU	2 octets	
27	All Subnets are Local	1 octet	
28	Broadcast Address	4 octets	
29	Perform Mask Discovery	1 octet	
30	Mask Supplier	1 octet	
31	Perform Router Discovery	1 octet	
32	Router Solicitation Address	4 octets	
33	Static Route	multiples of 8 octets	A list of destination/router pairs

Link Layer Parameters per Interface[12]

Code	Name	Length	Notes
34	Trailer Encapsulation Option	1 octet	
35	ARP Cache Timeout	4 octets	
36	Ethernet Encapsulation	1 octet	

TCP Parameters[13]

Code	Name	Length	Notes
37	TCP Default TTL	1 octet	
38	TCP Keepalive Interval	4 octets	
39	TCP Keepalive Garbage	1 octet	

Application and Service Parameters[14]

Code	Name	Length	Notes
40	Network Information Service Domain	minimum of 1 octet	
41	Network Information Servers	multiples of 4 octets	
42	Network Time Protocol Servers	multiples of 4 octets	
43	Vendor Specific Information	minimum of 1 octets	
44	NetBIOS over TCP/IP Name Server	multiples of 4 octets	
45	NetBIOS over TCP/IP Datagram Distribution Server	multiples of 4 octets	
46	NetBIOS over TCP/IP Node Type	1 octet	
47	NetBIOS over TCP/IP Scope	minimum of 1 octet	
48	X Window System Font Server	multiples of 4 octets	
49	X Window System Display Manager	multiples of 4 octets	
64	Network Information Service+ Domain	minimum of 1 octet	
65	Network Information Service+ Servers	multiples of 4 octets	
68	Mobile IP Home Agent	multiples of 4 octets	
69	Simple Mail Transport Protocol (SMTP) Server	multiples of 4 octets	
70	Post Office Protocol (POP3) Server	multiples of 4 octets	
71	Network News Transport Protocol (NNTP) Server	multiples of 4 octets	
72	Default World Wide Web (WWW) Server)	multiples of 4 octets	
73	Default Finger Server	multiples of 4 octets	
74	Default Internet Relay Chat (IRC) Server	multiples of 4 octets	
75	StreetTalk Server	multiples of 4 octets	
76	StreetTalk Directory Assistance (STDA) Server	multiples of 4 octets	

DHCP Extensions[15]

Code	Name	Length	Notes
50	Requested IP Address	4 octets	
51	IP Address Lease Time	4 octets	
52	Option Overload	1 octet	
66	TFTP server name	minimum of 1 octet	
67	Bootfile name	minimum of 1 octet	
53	DHCP Message Type	1 octet	
54	Server Identifier	4 octets	
55	Parameter Request List	minimum of 1 octet	
56	Message	minimum of 1 octet	
57	Maximum DHCP Message Size	2 octets	
58	Renewal (T1) Time Value	4 octets	
59	Rebinding (T2) Time Value	4 octets	
60	Vendor class identifier	minimum of 1 octet	
61	Client-identifier	minimum of 2 octets	

Vendor identification

An option exists to identify the vendor and functionality of a DHCP client. The information is a variable-length string of characters or octets which has a meaning specified by the vendor of the DHCP client. One method that a DHCP client can utilize to communicate to the server that it is using a certain type of hardware or firmware is to set a value in its DHCP requests called the Vendor Class Identifier (VCI) (Option 60). This method allows a DHCP server to differentiate between the two kinds of client machines and process the requests from the two types of modems appropriately. Some types of set-top boxes also set the VCI (Option 60) to inform the DHCP server about the hardware type and functionality of the device. The value that this option is set to give the DHCP server a hint about any required extra information that this client needs in a DHCP response.

DHCP relaying

In small networks, where only one IP subnet is being managed, DHCP clients communicate directly with DHCP servers. However, DHCP servers can also provide IP addresses for multiple subnets. In this case, a DHCP client that has not yet acquired an IP address cannot communicate directly with the DHCP server using IP routing, because it doesn't have a routable IP address, nor does it know the IP address of a router. In order to allow DHCP clients on subnets not directly served by DHCP servers to communicate with DHCP servers, DHCP relay agents can be installed on these subnets. The DHCP client broadcasts on the local link; the relay agent receives the broadcast and transmits it to one or more DHCP servers using unicast. The relay agent stores its own IP address in the GIADDR field of the DHCP packet. The DHCP server uses the GIADDR to determine the subnet on which the relay agent received the broadcast, and allocates an IP address on that subnet. When the DHCP server replies to the client, it sends the reply to the GIADDR address, again using unicast. The relay agent then retransmits the response on the local network.

Reliability

The DHCP protocol provides reliability in several ways: periodic renewal, rebinding, and failover. DHCP clients are allocated leases that last for some period of time. Clients begin to attempt to renew their leases once half the lease interval has expired. They do this by sending a unicast DHCPREQUEST message to the DHCP server that granted the original lease. If that server is down or unreachable, it will fail to respond to the DHCPREQUEST. However, the DHCPREQUEST will be repeated by the client from time to time, so when the DHCP server comes back up or becomes reachable again, the DHCP client will succeed in contacting it, and renew its lease.

If the DHCP server is unreachable for an extended period of time, the DHCP client will attempt to rebind, by broadcasting its DHCPREQUEST rather than unicasting it. Because it is broadcast, the DHCPREQUEST message will reach all available DHCP servers. If some other DHCP server is able to renew the lease, it will do so at this time.

In order for rebinding to work, when the client successfully contacts a backup DHCP server, that server must have accurate information about the client's binding. Maintaining accurate binding information between two servers is a complicated problem; if both servers are able to update the same lease database, there must be a mechanism to avoid conflicts between updates on the independent servers. A standard for implementing fault-tolerant DHCP servers was developed at the Internet Engineering Task Force.[16][17]

If rebinding fails, the lease will eventually expire. When the lease expires, the client must stop using the IP address granted to it in its lease. At that time, it will restart the DHCP process from the beginning by broadcasting a DHCPDISCOVER message. Since its lease has expired, it will accept any IP address offered to it. Once it has a new IP address, presumably from a different DHCP server, it will once again be able to use the network. However, since its IP address has changed, any ongoing connections will be broken.

Security

The base DHCP protocol does not include any mechanism for authentication.[18] Because of this, it is vulnerable to a variety of attacks. These attacks fall into three main categories:

- Unauthorized DHCP servers providing false information to clients.[19]
- Unauthorized clients gaining access to resources.[19]
- Resource exhaustion attacks from malicious DHCP clients.[19]

Because the client has no way to validate the identity of a DHCP server, unauthorized DHCP servers can be operated on networks, providing incorrect information to DHCP clients. This can serve either as a denial-of-service attack, preventing the client from gaining access to network connectivity, or as a man-in-the-middle attack. Because the DHCP server provides the DHCP client with server IP addresses, such as the IP address of one or more DNS servers,[19] an attacker can convince a DHCP client to do its DNS lookups through its own DNS server, and can therefore provide its own answers to DNS queries from the client.[20] This in turn allows the attacker to redirect network traffic through itself, allowing it to eavesdrop on connections between the client and network servers it contacts, or to simply replace those network servers with its own.[20]

Because the DHCP server has no secure mechanism for authenticating the client, clients can gain unauthorized access to IP addresses by presenting credentials, such as client identifiers, that belong to other DHCP clients. This also allows DHCP clients to exhaust the DHCP server's store of IP addresses—by presenting new credentials each time it asks for an address, the client can consume all the available IP addresses on a particular network link, preventing other DHCP clients from getting service.

DHCP does provide some mechanisms for mitigating these problems. The Relay Agent Information Option protocol extension (RFC 3046) allows network operators to attach tags to DHCP messages as these messages arrive on the network operator's trusted network. This tag is then used as an authorization token to control the client's access to network resources. Because the client has no access to the network upstream of the relay agent, the lack of

authentication does not prevent the DHCP server operator from relying on the authorization token.[18]

Another extension, Authentication for DHCP Messages (RFC 3118), provides a mechanism for authenticating DHCP messages. Unfortunately RFC 3118 has not seen widespread adoption because of the problems of managing keys for large numbers of DHCP clients.[21]

Notes

[1] Ralph Droms; Ted Lemon (2003). *The DHCP Handbook*. SAMS Publishing. p. 436. ISBN 0-672-32327-3.

[2] Bill Croft; John Gilmore (September 1985). "RFC 951 - Bootstrap Protocol" (http://tools.ietf.org/html/rfc951#section-6). *Network Working Group*. .

[3] http://www.iana.org/assignments/bootp-dhcp-parameters

[4] In Unix-like systems this client-level refinement typically takes place according to the values in a `/etc/dhclient.conf` configuration file.

[5] Alexander, Steve; Droms, Ralph (March 1997). *DHCP Options and BOOTP Vendor Extensions* (https://tools.ietf.org/html/rfc2132). IETF. RFC 2132. . Retrieved June 10, 2012.

[6] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-3). IETF. Section 3: RFC 1497 vendor extensions. . Retrieved 2012-07-26.

[7] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-3.1). IETF. Section 3.1: Pad Option. . Retrieved 2012-07-26.

[8] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-3.3). IETF. Section 3.3: Subnet Mask. . Retrieved 2012-07-26.

[9] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-3.4). IETF. Section 3.4: Time Offset. . Retrieved 2012-07-26.

[10] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-4). IETF. Section 4: IP Layer Parameters per Host. . Retrieved 2012-07-26.

[11] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-5). IETF. Section 5: IP Layer Parameters per Interface. . Retrieved 2012-07-26.

[12] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-6). IETF. Section 6: Link Layer Parameters per Interface. . Retrieved 2012-07-26.

[13] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-7). IETF. Section 7: TCP Parameters. . Retrieved 2012-07-26.

[14] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-8). IETF. Section 8: Application and Service Parameters. . Retrieved 2012-07-26.

[15] Alexander, Steve; Droms, Ralph (March 1997). "RFC 2132: DHCP Options and BOOTP Vendor Extensions" (http://tools.ietf.org/html/rfc2132#section-9). IETF. Section 9: DHCP Extensions. . Retrieved 2012-07-26.

[16] Droms, Ralph; Kinnear, Kim; Stapp, Mark; Volz, Bernie; Gonczi, Steve; Rabil, Greg; Dooley, Michael; Kapur, Arun (March 2003). *DHCP Failover Protocol* (https://tools.ietf.org/html/draft-ietf-dhc-failover-12). IETF. I-D draft-ietf-dhc-failover-12. . Retrieved May 09, 2010.

[17] The IETF proposal provided a mechanism whereby two servers could remain loosely in sync with each other in such a way that even in the event of a total failure of one server, the other server could recover the lease database and continue operating. Due to the length and complexity of the specification, it was never published as a standard; however, the techniques described in the specification are in wide use, with one open source implementation in the ISC DHCP server as well as several commercial implementations.

[18] Michael Patrick (January 2001). "RFC 3046 - DHCP Relay Agent Information Option" (http://tools.ietf.org/html/rfc3046#section-7). *Network Working Group*. .

[19] Ralph Droms (March 1997). "RFC 2131 - Dynamic Host Configuration Protocol" (http://tools.ietf.org/html/rfc2131#section-7). *Network Working Group*. .

[20] Sergey Golovanov (Kaspersky Labs) (June 2011). "TDSS loader now got "legs"" (http://www.securelist.com/en/blog/208188095/TDSS_loader_now_got_legs). .

[21] Ted Lemon (April 2002). "Implementation of RFC 3118" (http://www.ietf.org/mail-archive/web/dhcwg/current/msg00876.html). .

References

External links

- RFC 2131 - Dynamic Host Configuration Protocol
- RFC 2132 - DHCP Options and BOOTP Vendor Extensions
- RFC 3046 - DHCP Relay Agent Information Option
- RFC 3942 - Reclassifying Dynamic Host Configuration Protocol Version Four (DHCPv4) Options
- RFC 4242 - Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6
- RFC 4361 - Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)
- RFC 4436 - Detecting Network Attachment in IPv4 (DNAv4)

Domain Name System

The **Domain Name System** (**DNS**) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. A **Domain Name Service** resolves queries for these names into IP addresses for the purpose of locating computer services and devices worldwide. By providing a worldwide, distributed keyword-based redirection service, the Domain Name System is an essential component of the functionality of the Internet.

An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. For example, the domain name www.example.com translates to the addresses `192.0.43.10` (IPv4) and `2620:0:2d0:200::10` (IPv6). Unlike a phone book, however, DNS can be quickly updated and these updates distributed, allowing a service's location on the network to change without affecting the end users, who continue to use the same hostname. Users take advantage of this when they recite meaningful Uniform Resource Locators (URLs) and e-mail addresses without having to know how the computer actually locates the services.

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. This mechanism has made the DNS distributed and fault tolerant and has helped avoid the need for a single central register to be continually consulted and updated. Additionally, the responsibility for maintaining and updating the master record for the domains is spread among many domain name registrars, who compete for the end-user's, domain-owner's, business. Domains can be moved from registrar to registrar at any time.

The Domain Name System also specifies the technical functionality of this database service. It defines the DNS protocol, a detailed specification of the data structures and communication exchanges used in DNS, as part of the Internet Protocol Suite.

The Internet maintains two principal namespaces, the domain name hierarchy[1] and the Internet Protocol (IP) address spaces.[2] The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System.[3] A DNS name server is a server that stores the DNS records for a domain name, such as address (A) records, name server (NS) records, and mail exchanger (MX) records (see also list of DNS record types); a DNS name server responds with answers to queries against its database.

History

The practice of using a name as a simpler, more memorable abstraction of a host's numerical address on a network dates back to the ARPANET era. Before the DNS was invented in 1982, each computer on the network retrieved a file called *HOSTS.TXT* from a computer at SRI (now SRI International).[4][5] The HOSTS.TXT file mapped names to numerical addresses. A hosts file still exists on most modern operating systems by default and generally contains a mapping of "localhost" to the IP address 127.0.0.1. Many operating systems use name resolution logic that allows the administrator to configure selection priorities for available name resolution methods.

The rapid growth of the network made a centrally maintained, hand-crafted HOSTS.TXT file unsustainable; it became necessary to implement a more scalable system capable of automatically disseminating the requisite information.

At the request of Jon Postel, Paul Mockapetris invented the Domain Name System in 1983 and wrote the first implementation. The original specifications were published by the Internet Engineering Task Force in RFC 882 and RFC 883, which were superseded in November 1987 by RFC 1034[1] and RFC 1035.[3] Several additional Request for Comments have proposed various extensions to the core DNS protocols.

In 1984, four Berkeley students—Douglas Terry, Mark Painter, David Riggle, and Songnian Zhou—wrote the first Unix implementation, called The Berkeley Internet Name Domain (BIND) Server.[6] In 1985, Kevin Dunlap of DEC significantly re-wrote the DNS implementation. Mike Karels, Phil Almquist, and Paul Vixie have maintained BIND since then. BIND was ported to the Windows NT platform in the early 1990s.

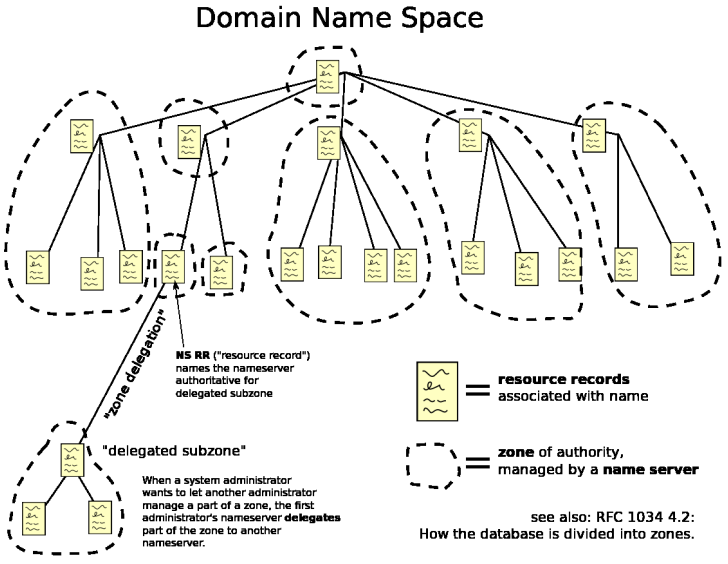
BIND was widely distributed, especially on Unix systems, and is the dominant DNS software in use on the Internet.[7] With the heavy use and resulting scrutiny of its open-source code, as well as increasingly more sophisticated attack methods, many security flaws were discovered in BIND. This contributed to the development of a number of alternative name server and resolver programs. BIND version 9 was written from scratch and now has a security record comparable to other modern DNS software.

Structure

Domain name space

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more *resource records*, which hold information associated with the domain name. The tree sub-divides into *zones* beginning at the root zone. A DNS zone may consist of only one domain, or may consist of many domains and sub-domains, depending on the administrative authority delegated to the manager.

Administrative responsibility over any zone may be divided by creating additional zones. Authority is said to be *delegated* for a portion of the old space, usually in the form of sub-domains, to another nameserver and administrative entity. The old zone ceases to be authoritative for the new zone.



The hierarchical Domain Name System, organized into zones, each served by a name server

Domain name syntax

The definitive descriptions of the rules for forming domain names appear in RFC 1035, RFC 1123, and RFC 2181. A domain name consists of one or more parts, technically called *labels*, that are conventionally concatenated, and delimited by dots, such as `example.com`.

- The right-most label conveys the top-level domain; for example, the domain name `www.example.com` belongs to the top-level domain `com`.
- The hierarchy of domains descends from right to left; each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label `example` specifies a subdomain of the `com` domain, and `www` is a sub domain of `example.com`. This tree of subdivisions may have up to 127 levels.
- Each label may contain up to 63 characters. The full domain name may not exceed a total length of 253 characters in its external dotted-label specification.[8] In the internal binary representation of the DNS the maximum length requires 255 octets of storage.[1] In practice, some domain registries may have shorter limits.
- DNS names may technically consist of any character representable in an octet. However, the allowed formulation of domain names in the DNS root zone, and most other sub domains, uses a preferred format and character set. The characters allowed in a label are a subset of the ASCII character set, and includes the characters *a* through *z*, *A* through *Z*, digits *0* through *9*, and the hyphen. This rule is known as the *LDH rule* (letters, digits, hyphen). Domain names are interpreted in case-independent manner.[9] Labels may not start or end with a hyphen.[10]
- A hostname is a domain name that has at least one IP address associated. For example, the domain names `www.example.com` and `example.com` are also hostnames, whereas the `com` domain is not.

Internationalized domain names

The permitted character set of the DNS prevented the representation of names and words of many languages in their native alphabets or scripts. ICANN has approved the Internationalizing Domain Names in Applications (IDNA) system, which maps Unicode strings into the valid DNS character set using Punycode. In 2009 ICANN approved the installation of IDN country code top-level domains. In addition, many registries of the existing top level domain names (TLD)s have adopted IDNA.

Name servers

The Domain Name System is maintained by a distributed database system, which uses the client-server model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root nameservers, the servers to query when looking up (*resolving*) a TLD.

Authoritative name server

An *authoritative* name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator.

An authoritative name server can either be a *master* server or a *slave* server. A master server is a server that stores the original (*master*) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records.

Every DNS zone must be assigned a set of authoritative name servers that are installed in NS records in the parent zone, and should be installed (to be authoritative records) as self-referential NS records on the authoritative name servers.

When domain names are registered with a domain name registrar, their installation at the domain registry of a top level domain requires the assignment of a *primary* name server and at least one *secondary* name server. The requirement of multiple name servers aims to make the domain still functional even if one name server becomes inaccessible or inoperable.[11] The designation of a primary name server is solely determined by the priority given to the domain name registrar. For this purpose, generally only the fully qualified domain name of the name server is required, unless the servers are contained in the registered domain, in which case the corresponding IP address is needed as well.

Primary name servers are often master name servers, while secondary name server may be implemented as slave servers.

An authoritative server indicates its status of supplying definitive answers, deemed *authoritative*, by setting a software flag (a protocol structure bit), called the *Authoritative Answer* (*AA*) bit in its responses.[3] This flag is usually reproduced prominently in the output of DNS administration query tools (such as dig) to indicate *that the responding name server is an authority for the domain name in question.*[3]

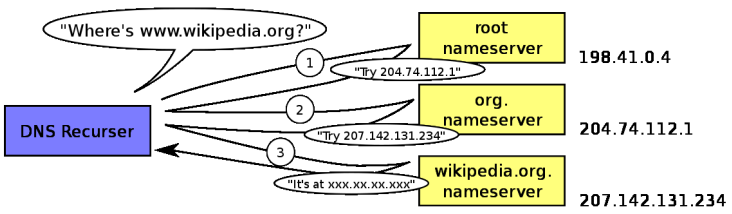
Operation

Address resolution mechanism

Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.

The process entails:

1. A network host is configured with an initial cache (so called *hints*) of the known addresses of the root nameservers. Such a *hint file* is updated periodically by an administrator from a reliable source.
2. A query to one of the root servers to find the server authoritative for the top-level domain.
3. A query to the obtained TLD server for the address of a DNS server authoritative for the second-level domain.
4. Repetition of the previous step to process each domain name label in sequence, until the final step which returns the IP address of the host sought.



A DNS recursor consults three nameservers to resolve the address www.wikipedia.org.

The diagram illustrates this process for the host www.wikipedia.org.

The mechanism in this simple form would place a large operating burden on the root servers, with every search for an address starting by querying one of them. Being as critical as they are to the overall function of the system, such heavy use would create an insurmountable bottleneck for trillions of queries placed every day. In practice caching is used in DNS servers to overcome this problem, and as a result, root nameservers actually are involved with very little of the total traffic.

Recursive and caching name server

In principle, authoritative name servers are sufficient for the operation of the Internet. However, with only authoritative name servers operating, every DNS query must start with recursive queries at the root zone of the Domain Name System and each user system must implement resolver software capable of recursive operation.

To improve efficiency, reduce DNS traffic across the Internet, and increase performance in end-user applications, the Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question. Typically, such *caching* DNS servers, also called *DNS caches*, also implement the recursive algorithm necessary to resolve a given name starting with the DNS root through to the authoritative name servers of the queried domain. With this function implemented in the name server, user applications gain efficiency in design and operation.

The combination of DNS caching and recursive functions in a name server is not mandatory; the functions can be implemented independently in servers for special purposes.

Internet service providers typically provide recursive and caching name servers for their customers. In addition, many home networking routers implement DNS caches and recursors to improve efficiency in the local network.

DNS resolvers

The client-side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.

A DNS query may be either a non-recursive query or a recursive query:

- A *non-recursive query* is one in which the DNS server provides a record for a domain for which it is authoritative itself, or it provides a partial result without querying other servers.
- A *recursive query* is one for which the DNS server will fully answer the query (or give an error) by querying other name servers as needed. DNS servers are not required to support recursive queries.

The resolver, or another DNS server acting recursively on behalf of the resolver, negotiates use of recursive service using bits in the query headers.

Resolving usually entails iterating through several name servers to find the needed information. However, some resolvers function more simply by communicating only with a single name server. These simple resolvers (called "stub resolvers") rely on a recursive name server to perform the work of finding information for them.

Circular dependencies and glue records

Name servers in delegations are identified by name, rather than by IP address. This means that a resolving name server must issue another DNS request to find out the IP address of the server to which it has been referred. If the name given in the delegation is a subdomain of the domain for which the delegation is being provided, there is a circular dependency. In this case the nameserver providing the delegation must also provide one or more IP addresses for the authoritative nameserver mentioned in the delegation. This information is called *glue*. The delegating name server provides this glue in the form of records in the *additional section* of the DNS response, and provides the delegation in the *answer section* of the response.

For example, if the authoritative name server for `example.org` is `ns1.example.org`, a computer trying to resolve `www.example.org` first resolves `ns1.example.org`. Since `ns1` is contained in `example.org`, this requires resolving `example.org` first, which presents a circular dependency. To break the dependency, the nameserver for the `org` top level domain includes glue along with the delegation for `example.org`. The glue records are address records that provide IP addresses for `ns1.example.org`. The resolver uses one or more of these IP addresses to query one of the domain's authoritative servers, which allows it to complete the DNS query.

Record caching

The DNS Resolution Process reduces the load on individual servers by *caching* DNS request records for a period of time after a response. This entails the local recording and subsequent consultation of the copy instead of initiating a new request upstream. The time for which a resolver caches a DNS response is determined by a value called the time to live (TTL) associated with every record. The TTL is set by the administrator of the DNS server handing out the authoritative response. The period of validity may vary from just seconds to days or even weeks.

As a noteworthy consequence of this distributed and caching architecture, changes to DNS records do not propagate throughout the network immediately, but require all caches to expire and refresh after the TTL. RFC 1912 conveys basic rules for determining appropriate TTL values.

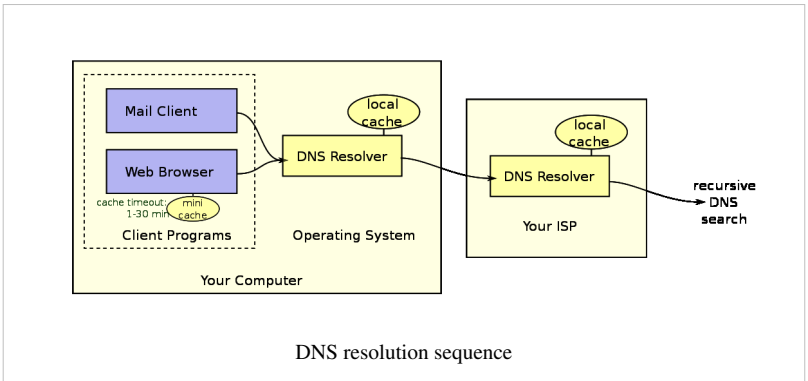
Some resolvers may override TTL values, as the protocol supports caching for up to 68 years or no caching at all. Negative caching, i.e. the caching of the fact of non-existence of a record, is determined by name servers authoritative for a zone which must include the Start of Authority (SOA) record when reporting no data of the requested type exists. The value of the *MINIMUM* field of the SOA record and the TTL of the SOA itself is used to establish the TTL for the negative answer.

Reverse lookup

A reverse lookup is a query of the DNS for domain names when the IP address is known. Multiple domain names may be associated with an IP address. The DNS stores IP addresses in the form of domain names as specially formatted names in pointer (`PTR`) records within the infrastructure top-level domain arpa. For IPv4, the domain is `in-addr.arpa`. For IPv6, the reverse lookup domain is `ip6.arpa`. The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.

When performing a reverse lookup, the DNS client converts the address into these formats, and then queries the name for a PTR record following the delegation chain as for any DNS query. For example, assume the IPv4 address `208.80.152.2` is assigned to Wikimedia. It is represented as a DNS name in reverse order like this: `2.152.80.208.in-addr.arpa`. When the DNS resolver gets a PTR (reverse-lookup) request, it begins by querying the root servers (which point to The American Registry For Internet Numbers' (ARIN [12]'s) servers for the `208.in-addr.arpa` zone). On ARIN's servers, `152.80.208.in-addr.arpa` is assigned to Wikimedia, so the resolver sends another query to the Wikimedia nameserver for `2.152.80.208.in-addr.arpa`, which results in an authoritative response.

Client lookup



DNS resolution sequence

Users generally do not communicate directly with a DNS resolver. Instead DNS resolution takes place transparently in applications such as web browsers, e-mail clients, and other Internet applications. When an application makes a request that requires a domain name lookup, such programs send a resolution request to the DNS resolver in the local operating system, which in turn handles the communications required.

The DNS resolver will almost invariably have a cache (see above) containing recent lookups. If the cache can provide the answer to the request, the resolver will return the value in the cache to the program that made the request. If the cache does not contain the answer, the resolver will send the request to one or more designated DNS servers. In the case of most home users, the Internet service provider to which the machine connects will usually supply this DNS server: such a user will either have configured that server's address manually or allowed DHCP to set it; however, where systems administrators have configured systems to use their own DNS servers, their DNS resolvers point to separately maintained nameservers of the organization. In any event, the name server thus queried will follow the process outlined above, until it either successfully finds a result or does not. It then returns its results to the DNS resolver; assuming it has found a result, the resolver duly caches that result for future use, and hands the result back to the software which initiated the request.

Broken resolvers

An additional level of complexity emerges when resolvers violate the rules of the DNS protocol. A number of large ISPs have configured their DNS servers to violate rules (presumably to allow them to run on less-expensive hardware than a fully compliant resolver), such as by disobeying TTLs, or by indicating that a domain name does not exist just because one of its name servers does not respond.[13]

As a final level of complexity, some applications (such as web-browsers) also have their own DNS cache, in order to reduce the use of the DNS resolver library itself. This practice can add extra difficulty when debugging DNS issues, as it obscures the freshness of data, and/or what data comes from which cache. These caches typically use very short caching times—on the order of one minute.[14]

Internet Explorer represents a notable exception: versions up to IE 3.x cache DNS records for 24 hours by default. Internet Explorer 4.x and later versions (up to IE 8) decrease the default time out value to half an hour, which may be changed in corresponding registry keys.[15]

Other applications

The system outlined above provides a somewhat simplified scenario. The Domain Name System includes several other functions:

- Hostnames and IP addresses do not necessarily match on a one-to-one basis. Multiple hostnames may correspond to a single IP address: combined with virtual hosting, this allows a single machine to serve many web sites. Alternatively, a single hostname may correspond to many IP addresses: this can facilitate fault tolerance and load distribution, and also allows a site to move physical locations seamlessly.
- There are many uses of DNS besides translating names to IP addresses. For instance, Mail transfer agents use DNS to find out where to deliver e-mail for a particular address. The domain to mail exchanger mapping provided by MX records accommodates another layer of fault tolerance and load distribution on top of the name to IP address mapping.
- E-mail Blacklists: The DNS is used for efficient storage and distribution of IP addresses of blacklisted e-mail hosts. The usual method is putting the IP address of the subject host into the sub-domain of a higher level domain name, and resolve that name to different records to indicate a positive or a negative. Here is a hypothetical example blacklist:
  - 102.3.4.5 is blacklisted => Creates 5.4.3.102.blacklist.example and resolves to 127.0.0.1
  - 102.3.4.6 is not => 6.4.3.102.blacklist.example is not found, or default to 127.0.0.2
  - E-mail servers can then query blacklist.example through the DNS mechanism to find out if a specific host connecting to them is in the blacklist. Today many of such blacklists, either free or subscription-based, are available mainly for use by email administrators and anti-spam software.
- Sender Policy Framework and DomainKeys, instead of creating their own record types, were designed to take advantage of another DNS record type, the TXT record.
- To provide resilience in the event of computer failure, multiple DNS servers are usually provided for coverage of each domain, and at the top level, thirteen very powerful root servers exist, with additional "copies" of several of them distributed worldwide via Anycast.
- Dynamic DNS (sometimes called DDNS) allows clients to update their DNS entry as their IP address changes, as it does, for example, when moving between ISPs or mobile hot spots.

Protocol details

DNS primarily uses User Datagram Protocol (UDP) on port number 53 to serve requests.[3] DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server. The Transmission Control Protocol (TCP) is used when the response data size exceeds 512 bytes, or for tasks such as zone transfers. Some resolver implementations use TCP for all queries.

DNS resource records

A Resource Record (RR) is the basic data element in the domain name system. Each record has a type (A, MX, etc.), an expiration time limit, a class, and some type-specific data. Resource records of the same type define a resource record set (RRset). The order of resource records in a set, returned by a resolver to an application, is undefined, but often servers implement round-robin ordering to achieve Global Server Load Balancing. DNSSEC, however, works on complete resource record sets in a canonical order.

When sent over an IP network, all records use the common format specified in RFC 1035:[16]

RR (Resource record) fields

Field	Description	Length (octets)
NAME	Name of the node to which this record pertains	(variable)
TYPE	Type of RR in numeric form (e.g. 15 for MX RRs)	2
CLASS	Class code	2
TTL	Count of seconds that the RR stays valid (The maximum is 2^{31}-1, which is about 68 years.)	4
RDLENGTH	Length of RDATA field	2
RDATA	Additional RR-specific data	(variable)

NAME is the fully qualified domain name of the node in the tree. On the wire, the name may be shortened using label compression where ends of domain names mentioned earlier in the packet can be substituted for the end of the current domain name. A free standing @ is used to denote the current origin.

TYPE is the record type. It indicates the format of the data and it gives a hint of its intended use. For example, the *A* record is used to translate from a domain name to an IPv4 address, the *NS* record lists which name servers can answer lookups on a DNS zone, and the *MX* record specifies the mail server used to handle mail for a domain specified in an e-mail address (see also List of DNS record types).

RDATA is data of type-specific relevance, such as the IP address for address records, or the priority and hostname for MX records. Well known record types may use label compression in the RDATA field, but "unknown" record types must not (RFC 3597).

The *CLASS* of a record is set to `IN` (for *Internet*) for common DNS records involving Internet hostnames, servers, or IP addresses. In addition, the classes Chaos (`CH`) and Hesiod (`HS`) exist.[17] Each class is an independent name space with potentially different delegations of DNS zones.

In addition to resource records defined in a zone file, the domain name system also defines several request types that are used only in communication with other DNS nodes (*on the wire*), such as when performing zone transfers (AXFR/IXFR) or for EDNS (OPT).

Wildcard DNS records

The domain name system supports *wildcard domain names* which are names that start with the *asterisk label*, '*', e.g., `*.example`.[1][18] DNS records belonging to wildcard domain names specify rules for generating resource records within a single DNS zone by substituting whole labels with matching components of the query name, including any specified descendants. For example, in the DNS zone *x.example*, the following configuration specifies that all subdomains (including subdomains of subdomains) of *x.example* use the mail exchanger *a.x.example*. The records for *a.x.example* are needed to specify the mail exchanger. As this has the result of excluding this domain name and its subdomains from the wildcard matches, all subdomains of *a.x.example* must be defined in a separate wildcard statement.

The role of wildcard records was refined in RFC 4592, because the original definition in RFC 1034 was incomplete and resulted in misinterpretations by implementers.[18]

Protocol extensions

The original DNS protocol had limited provisions for extension with new features. In 1999, Paul Vixie published in RFC 2671 an extension mechanism, called Extension mechanisms for DNS (EDNS) that introduced optional protocol elements without increasing overhead when not in use. This was accomplished through the `OPT` pseudo-resource record that only exists in wire transmissions of the protocol, but not in any zone files. Initial extensions were also suggested (EDNS0), such as increasing the DNS message size in UDP datagrams.

Dynamic zone updates

Dynamic DNS updates use the `UPDATE` DNS opcode to add or remove resource records dynamically from a zone data base maintained on an authoritative DNS server. The feature is described in RFC 2136. This facility is useful to register network clients into the DNS when they boot or become otherwise available on the network. Since a booting client may be assigned a different IP address each time from a DHCP server, it is not possible to provide static DNS assignments for such clients.

Security issues

Originally, security concerns were not major design considerations for DNS software or any software for deployment on the early Internet, as the network was not open for participation by the general public. However, the expansion of the Internet into the commercial sector in the 1990s changed the requirements for security measures to protect data integrity and user authentication.

Several vulnerability issues were discovered and exploited by malicious users. One such issue is DNS cache poisoning, in which data is distributed to caching resolvers under the pretense of being an authoritative origin server, thereby polluting the data store with potentially false information and long expiration times (time-to-live). Subsequently, legitimate application requests may be redirected to network hosts operated with malicious intent.

DNS responses are traditionally not cryptographically signed, leading to many attack possibilities; the Domain Name System Security Extensions (DNSSEC) modify DNS to add support for cryptographically signed responses. Several extensions have been devised to secure zone transfers as well.

Some domain names may be used to achieve spoofing effects. For example, paypal.com and paypa1.com are different names, yet users may be unable to distinguish them in a graphical user interface depending on the user's chosen typeface. In many fonts the letter *l* and the numeral *1* look very similar or even identical. This problem is acute in systems that support internationalized domain names, since many character codes in ISO 10646, may appear identical on typical computer screens. This vulnerability is occasionally exploited in phishing.[19]

Techniques such as forward-confirmed reverse DNS can also be used to help validate DNS results.

Domain name registration

The right to use a domain name is delegated by domain name registrars which are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN), the organization charged with overseeing the name and number systems of the Internet. In addition to ICANN, each top-level domain (TLD) is maintained and serviced technically by an administrative organization, operating a registry. A registry is responsible for maintaining the database of names registered within the TLD it administers. The registry receives registration information from each domain name registrar authorized to assign names in the corresponding TLD and publishes the information using a special service, the whois protocol.

ICANN publishes the complete list of TLD registries and domain name registrars. Registrant information associated with domain names is maintained in an online database accessible with the WHOIS service. For most of the more than 240 country code top-level domains (ccTLDs), the domain registries maintain the WHOIS (Registrant, name servers, expiration dates, etc.) information. For instance, DENIC, Germany NIC, holds the DE domain data. Since about 2001, most gTLD registries have adopted this so-called *thick* registry approach, i.e. keeping the WHOIS data in central registries instead of registrar databases.

For `COM` and `NET` domain names, a *thin* registry model is used: the domain registry (e.g. VeriSign) holds basic WHOIS (registrar and name servers, etc.) data. One can find the detailed WHOIS (registrant, name servers, expiry dates, etc.) at the registrars.

Some domain name registries, often called *network information centers* (NIC), also function as registrars to end-users. The major generic top-level domain registries, such as for the `COM`, `NET`, `ORG`, `INFO` domains, use a registry-registrar model consisting of many domain name registrars.[20][21] In this method of management, the registry only manages the domain name database and the relationship with the registrars. The *registrants* (users of a domain name) are customers of the registrar, in some cases through additional layers of resellers.

Internet standards

The Domain Name System is defined by Request for Comments (RFC) documents published by the Internet Engineering Task Force (Internet standards). The following is a list of RFCs that define the DNS protocol.

- RFC 920, *Domain Requirements* – Specified original top-level domains
- RFC 1032, *Domain Administrators Guide*
- RFC 1033, *Domain Administrators Operations Guide*
- RFC 1034, *Domain Names - Concepts and Facilities*
- RFC 1035, *Domain Names - Implementation and Specification*
- RFC 1101, *DNS Encodings of Network Names and Other Types*
- RFC 1123, *Requirements for Internet Hosts—Application and Support*
- RFC 1178, *Choosing a Name for Your Computer* (FYI 5)
- RFC 1183, *New DNS RR Definitions*
- RFC 1591, *Domain Name System Structure and Delegation* (Informational)
- RFC 1912, *Common DNS Operational and Configuration Errors*
- RFC 1995, *Incremental Zone Transfer in DNS*
- RFC 1996, *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*
- RFC 2100, *The Naming of Hosts* (Informational)
- RFC 2136, *Dynamic Updates in the domain name system (DNS UPDATE)*
- RFC 2181, *Clarifications to the DNS Specification*
- RFC 2182, *Selection and Operation of Secondary DNS Servers*
- RFC 2308, *Negative Caching of DNS Queries (DNS NCACHE)*
- RFC 2317, *Classless IN-ADDR.ARPA delegation* (BCP 20)

- RFC 2671, *Extension Mechanisms for DNS (EDNS0)*
- RFC 2672, *Non-Terminal DNS Name Redirection*
- RFC 2845, *Secret Key Transaction Authentication for DNS (TSIG)*
- RFC 3225, *Indicating Resolver Support of DNSSEC*
- RFC 3226, *DNSSEC and IPv6 A6 aware server/resolver message size requirements*
- RFC 3597, *Handling of Unknown DNS Resource Record (RR) Types*
- RFC 3696, *Application Techniques for Checking and Transformation of Names* (Informational)
- RFC 4343, *Domain Name System (DNS) Case Insensitivity Clarification*
- RFC 4592, *The Role of Wildcards in the Domain Name System*
- RFC 4635, *HMAC SHA TSIG Algorithm Identifiers*
- RFC 4892, *Requirements for a Mechanism Identifying a Name Server Instance* (Informational)
- RFC 5001, *DNS Name Server Identifier (NSID) Option*
- RFC 5452, *Measures for Making DNS More Resilient against Forged Answers*
- RFC 5625, *DNS Proxy Implementation Guidelines* (BCP 152)
- RFC 5890, *Internationalized Domain Names for Applications (IDNA):Definitions and Document Framework*
- RFC 5891, *Internationalized Domain Names in Applications (IDNA): Protocol*
- RFC 5892, *The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)*
- RFC 5893, *Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)*
- RFC 5894, *Internationalized Domain Names for Applications (IDNA):Background, Explanation, and Rationale* (Informational)
- RFC 5895, *Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008* (Informational)
- RFC 5966, *DNS Transport over TCP - Implementation Requirements*
- RFC 6195, *Domain Name System (DNS) IANA Considerations* (BCP 42)

Security

- RFC 4033, *DNS Security Introduction and Requirements*
- RFC 4034, *Resource Records for the DNS Security Extensions*
- RFC 4035, *Protocol Modifications for the DNS Security Extensions*
- RFC 4509, *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records*
- RFC 4470, *Minimally Covering NSEC Records and DNSSEC On-line Signing*
- RFC 5011, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*
- RFC 5155, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*
- RFC 5702, *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC*
- RFC 5910, *Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)*
- RFC 5933, *Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC*

References

[1] RFC 1034, *Domain Names - Concepts and Facilities*, P. Mockapetris, The Internet Society (November 1987)

[2] RFC 781, *Internet Protocol - DARPA Internet Program Protocol Specification*, Information Sciences Institute, J. Postel (Ed.), The Internet Society (September 1981)

[3] RFC 1035, *Domain Names - Implementation and Specification*, P. Mockapetris, The Internet Society (November 1987)

[4] RFC 3467, *Role of the Domain Name System (DNS)*, J.C. Klensin, J. Klensin (February 2003)

[5] Cricket Liu, Paul Albitz (2006). *DNS and BIND* (http://oreilly.com/catalog/9780596100575) (5th ed.). O'Reilly. p. 3. .

[6] Douglas Brian Terry, Mark Painter, David W. Riggle and Songnian Zhou, *The Berkeley Internet Name Domain Server* (http://www.eecs.berkeley.edu/Pubs/TechRpts/1984/5957.html), Proceedings USENIX Summer Conference, Salt Lake City, Utah, June 1984, pages 23–31.

[7] "DNS Server Survey" (http://mydns.bboy.net/survey/). .

[8] RFC 2181, *Clarifications to the DNS Specification*, R. Elz, R. Bush (July 1997)

[9] Network Working Group of the IETF, January 2006, RFC 4343: Domain Name System (DNS) Case Insensitivity Clarification

[10] RFC 3696, *Application Techniques for Checking and Transformation of Names*, J.C. Klensin, J. Klensin

[11] "Name Server definition at techterms.com" (http://www.techterms.com/definition/nameserver). .

[12] https://www.arin.net/

[13] "Providers ignoring DNS TTL?" (http://ask.slashdot.org/story/05/04/18/198259/providers-ignoring-dns-ttl). Slashdot. 2005. . Retrieved 2012-04-07.

[14] Ben Anderson: Why Web Browser DNS Caching Can Be A Bad Thing (http://dyn.com/web-browser-dns-caching-bad-thing/)

[15] "How Internet Explorer uses the cache for DNS host entries" (http://support.microsoft.com/default.aspx?scid=KB;en-us;263558). Microsoft Corporation. 2004. . Retrieved 2010-07-25.

[16] RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), Section 3

[17] RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), p. 11

[18] RFC 4592, *The Role of Wildcards in the Domain Name System*, E. Lewis (July 2006)

[19] APWG. "Global Phishing Survey: Domain Name Use and Trends in 1H2010." 10/15/2010 apwg.org (http://www.apwg.org/reports/APWG_GlobalPhishingSurvey_1H2010.pdf.)

[20] ICANN accredited registrars (http://www.icann.org/registrars/accredited-list.html)

[21] VeriSign COM and NET registry (http://www.verisign.com/information-services/naming-services/com-net-registry/page_002166.html)

External links

- Vixie, Paul (2007-05-04). "DNS Complexity – Although it contains just a few simple rules, DNS has grown into an enormously complex system." (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=481). ACM Queue.
- Zytrax.com (http://www.zytrax.com/books/dns/), Open Source Guide – DNS for Rocket Scientists, an on-line technical.
- Create a zone file (http://www.zonefile.org/?lang=en) on zonefile.org
- Domain Name System (http://www.microsoft.com/dns) on Microsoft TechNet

IPv4

Internet Protocol version 4 (**IPv4**) is the fourth revision in the development of the Internet Protocol (IP) and the first version of the protocol to be widely deployed. Together with IPv6, it is at the core of standards-based internetworking methods of the Internet. As of 2012 IPv4 is still the most widely deployed Internet Layer protocol.

IPv4 is described in IETF publication RFC 791 (September 1981), replacing an earlier definition (RFC 760, January 1980).

IPv4 is a connectionless protocol for use on packet-switched Link Layer networks (e.g., Ethernet). It operates on a best effort delivery model, in that it does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol, such as the Transmission Control Protocol (TCP).

Addressing

IPv4 uses 32-bit (four-byte) addresses, which limits the address space to 4294967296 (2^{32}) addresses. Addresses were assigned to users, and the number of unassigned addresses decreased. IPv4 address exhaustion occurred on February 3, 2011. It had been significantly delayed by address changes such as classful network design, Classless Inter-Domain Routing, and network address translation (NAT).

This limitation of IPv4 stimulated the development of IPv6 in the 1990s, which has been in commercial deployment since 2006.

IPv4 reserves special address blocks for private networks (~18 million addresses) and multicast addresses (~270 million addresses).

Address representations

IPv4 addresses may be written in any notation expressing a 32-bit integer value, but for human convenience, they are most often written in the dot-decimal notation, which consists of four octets of the address expressed individually in decimal and separated by periods.

The following table shows several representation formats:

Notation	Value	Conversion from dot-decimal
Dotted decimal	192.0.2.235	N/A
Dotted hexadecimal[1]	0xC0.0x00.0x02.0xEB	Each octet is individually converted to hexadecimal form
Dotted octal[1]	0300.0000.0002.0353	Each octet is individually converted into octal
Hexadecimal	0xC00002EB	Concatenation of the octets from the dotted hexadecimal
Decimal	3221226219	The 32-bit number expressed in decimal
Octal	030000001353	The 32-bit number expressed in octal

Allocation

Originally, an IP address was divided into two parts: the network identifier was the most significant (highest order) octet of the address, and the host identifier was the rest of the address. The latter was therefore also called the *rest field*. This enabled the creation of a maximum of 256 networks. This was quickly found to be inadequate.

To overcome this limit, the high order octet of the addresses was redefined to create a set of *classes* of networks, in a system which later became known as classful networking. The system defined five classes, Class A, B, C, D, and E. The Classes A, B, and C had different bit lengths for the new network identification. The rest of an address was used as previously to identify a host within a network, which meant that each network class had a different capacity to address hosts. Class D was allocated for multicast addressing and Class E was reserved for future applications.

Starting around 1985, people devised methods to subdivide IP networks. One flexible method was the *variable-length subnet mask* (VLSM).[2][3]

Around 1993, this system of classes was officially replaced with Classless Inter-Domain Routing (CIDR), and the class-based scheme was dubbed *classful*, by contrast. CIDR was designed to permit repartitioning of any address space so that smaller or larger blocks of addresses could be allocated to users. The hierarchical structure created by CIDR is managed by the Internet Assigned Numbers Authority (IANA) and the regional Internet registries (RIRs). Each RIR maintains a publicly searchable WHOIS database that provides information about IP address assignments.

Special-use addresses

Reserved address blocks

Range	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 1700
10.0.0.0/8	Private network	RFC 1918
100.64.0.0/10	Shared Address Space	RFC 6598
127.0.0.0/8	Loopback	RFC 5735
169.254.0.0/16	Link-local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	Reserved (IANA)	RFC 5735
192.0.2.0/24	TEST-NET-1, documentation and examples	RFC 5735
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, documentation and examples	RFC 5737
224.0.0.0/4	IP multicast (former Class D network)	RFC 5771
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

Private networks

Of the approximately four billion addresses allowed in IPv4, three ranges of address are reserved for use in private networks. These ranges are not routable outside of private networks, and private machines cannot directly communicate with public networks. They can, however, do so through network address translation.

The following are the three ranges reserved for private networks (RFC 1918):

Name	Address range	Number of addresses	*Classful* description	Largest CIDR block
24-bit block	10.0.0.0–10.255.255.255	16777216	Single Class A	10.0.0.0/8
20-bit block	172.16.0.0–172.31.255.255	1048576	Contiguous range of 16 Class B blocks	172.16.0.0/12
16-bit block	192.168.0.0–192.168.255.255	65536	Contiguous range of 256 Class C blocks	192.168.0.0/16

Virtual private networks

Packets with a private destination address are ignored by all public routers. Two private networks (e.g., two branch offices) cannot communicate via the public internet, unless they use an IP tunnel or a virtual private network (VPN). When one private network wants to send a packet to another private network, the first private network encapsulates the packet in a protocol layer so that the packet can travel through the public network. Then the packet travels through the public network. When the packet reaches the other private network, its protocol layer is removed, and the packet travels to its destination.

Optionally, encapsulated packets may be encrypted to secure the data while it travels over the public network.

Link-local addressing

RFC 5735 defines the special address block 169.254.0.0/16 for link-local addressing. These addresses are only valid on links (such as a local network segment or point-to-point connection) connected to a host. These addresses are not routable. Like private addresses, these addresses cannot be the source or destination of packets traversing the internet. These addresses are primarily used for address autoconfiguration (Zeroconf) when a host cannot obtain an IP address from a DHCP server or other internal configuration methods.

When the address block was reserved, no standards existed for address autoconfiguration. Microsoft created an implementation called Automatic Private IP Addressing (APIPA), which was deployed on millions of machines and became a de facto standard. Many years later, in May 2005, the IETF defined a formal standard in RFC 3927, entitled *Dynamic Configuration of IPv4 Link-Local Addresses*.

Loopback

The class A network 127.0.0.0 (classless network 127.0.0.0/8) is reserved for loopback. IP packets which source addresses belong to this network should never appear outside a host. The modus operandi of this network expands upon that of a loopback interface:

- IP packets which source and destination addresses belong to the network (or subnetwork) of the same loopback interface are returned back to that interface;
- IP packets which source and destination addresses belong to networks (or subnetworks) of different interfaces of the same host, one of them being a loopback interface, are forwarded regularly.

Addresses ending in 0 or 255

Networks with subnet masks of at least 24 bits, i.e. Class C networks in classful networking, and networks with CIDR prefixes /24 to /32 (255.255.255.0–255.255.255.255) may not have an address ending in 0 or 255.

Classful addressing prescribed only three possible subnet masks: Class A, 255.0.0.0 or /8; Class B, 255.255.0.0 or /16; and Class C, 255.255.255.0 or /24. For example, in the subnet 192.168.5.0/255.255.255.0 (192.168.5.0/24) the identifier 192.168.5.0 commonly is used to refer to the entire subnet. To avoid ambiguity in representation, the address ending in the octet *0* is reserved.

A broadcast address is an address that allows information to be sent to all interfaces in a given subnet, rather than a specific machine. Generally, the broadcast address is found by obtaining the bit complement of the subnet mask and performing a bitwise OR operation with the network identifier. In other words, the broadcast address is the last address in the address range of the subnet. For example, the broadcast address for the network 192.168.5.0 is 192.168.5.255. For networks of size /24 or larger, the broadcast address always ends in 255.

However, this does not mean that every address ending in 0 or 255 cannot be used as a host address. For example, consider a /16 subnet 192.168.0.0/255.255.0.0, which is equivalent to the address range 192.168.0.0–192.168.255.255. The broadcast address is 192.168.255.255. One can use the following addresses for hosts, even though they end with 255: 192.168.1.255, 192.168.2.255, etc. Also, 192.168.0.0 is the network identifier and must not be used for a host.[4] One can use the following addresses for hosts, even though they end with 0: 192.168.1.0, 192.168.2.0, etc.

In the past, conflict between network addresses and broadcast addresses arose because some software used non-standard broadcast addresses with zeros instead of ones.[5]

In networks smaller than /24, broadcast addresses do not necessarily end with 255. For example, a CIDR subnet 203.0.113.16/28 has the broadcast address 203.0.113.31.

Address resolution

Hosts on the Internet are usually known by names, e.g., www.example.com, not primarily by their IP address, which is used for routing and network interface identification. The use of domain names requires translating, called *resolving*, them to addresses and vice versa. This is analogous to looking up a phone number in a phone book using the recipient's name.

The translation between addresses and domain names is performed by the Domain Name System (DNS), a hierarchical, distributed naming system which allows for subdelegation of name spaces to other DNS servers. DNS is often described in analogy to the telephone system directory information systems in which subscriber names are translated to telephone numbers.

Address space exhaustion

Since the 1980s, it was apparent that the pool of available IPv4 addresses was being depleted at a rate that was not initially anticipated in the original design of the network address system.[6] The threat of exhaustion was the motivation for remedial technologies, such as classful networks, Classless Inter-Domain Routing (CIDR) methods, and network address translation (NAT). Eventually, IPv6 was created, which has many more addresses available.

Several market forces accelerated IPv4 address exhaustion:

- Rapidly growing number of Internet users
- Always-on devices — ADSL modems, cable modems
- Mobile devices — laptop computers, PDAs, mobile phones

Some technologies mitigated IPv4 address exhaustion:

- Network address translation (NAT) is a technology that allows a private network to use one public IP address. It permits private addresses in the private network.
- Use of private networks
- Dynamic Host Configuration Protocol (DHCP)
- Name-based virtual hosting of web sites
- Tighter control by regional Internet registries over the allocation of addresses to local Internet registries
- Network renumbering to reclaim large blocks of address space allocated in the early days of the Internet

The primary address pool of the Internet, maintained by IANA, was exhausted on 3 February 2011, when the last 5 blocks were allocated to the 5 RIRs.[7][8] APNIC was the first RIR to exhaust its regional pool on 15 April 2011, except for a small amount of address space reserved for the transition to IPv6, which will be allocated under a much more restricted policy.[9]

The accepted and standard solution is to use Internet Protocol Version 6. The address size was increased in IPv6 to 128 bits, providing a vastly increased address space that also allows improved route aggregation across the Internet and offers large subnetwork allocations of a minimum of 2^{64} host addresses to end-users. Migration to IPv6 is in progress but completion is expected to take considerable time.

Packet structure

An IP packet consists of a header section and a data section.

Header

The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional (red background in table) and aptly named: options. The fields in the header are packed with the most significant byte first (big endian), and for the diagram and discussion, the most significant bits are considered to come first (MSB 0 bit numbering). The most significant bit is numbered 0, so the version field is actually found in the four most significant bits of the first byte, for example.

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

Version

The first header field in an IP packet is the four-bit version field. For IPv4, this has a value of 4 (hence the name IPv4).

Internet Header Length (IHL)

The second field (4 bits) is the Internet Header Length (IHL), which is the number of 32-bit words in the header. Since an IPv4 header may contain a variable number of options, this field specifies the size of the header (this also coincides with the offset to the data). The minimum value for this field is 5 (RFC 791), which is a length of 5×32 = 160 bits = 20 bytes. Being a 4-bit value, the maximum length is 15 words (15×32 bits) or

480 bits = 60 bytes.

Differentiated Services Code Point (DSCP)

Originally defined as the Type of Service field, this field is now defined by RFC 2474 for Differentiated services (DiffServ). New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange.

Explicit Congestion Notification (ECN)

This field is defined in RFC 3168 and allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.

Total Length

This 16-bit field defines the entire packet (fragment) size, including header and data, in bytes. The minimum-length packet is 20 bytes (20-byte header + 0 bytes data) and the maximum is 65,535 bytes — the maximum value of a 16-bit word. The largest datagram that any host is required to be able to reassemble is 576 bytes, but most modern hosts handle much larger packets. Sometimes subnetworks impose further restrictions on the packet size, in which case datagrams must be fragmented. Fragmentation is handled in either the host or router in IPv4.

Identification

This field is an identification field and is primarily used for uniquely identifying fragments of an original IP datagram. Some experimental work has suggested using the ID field for other purposes, such as for adding packet-tracing information to help trace datagrams with spoofed source addresses.[10]

Flags

A three-bit field follows and is used to control or identify fragments. They are (in order, from high order to low order):

- bit 0: Reserved; must be zero.[11]
- bit 1: Don't Fragment (DF)
- bit 2: More Fragments (MF)

If the DF flag is set, and fragmentation is required to route the packet, then the packet is dropped. This can be used when sending packets to a host that does not have sufficient resources to handle fragmentation. It can also be used for Path MTU Discovery, either automatically by the host IP software, or manually using diagnostic tools such as ping or traceroute.

For unfragmented packets, the MF flag is cleared. For fragmented packets, all fragments except the last have the MF flag set. The last fragment has a non-zero Fragment Offset field, differentiating it from an unfragmented packet.

Fragment Offset

The fragment offset field, measured in units of eight-byte blocks, is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of $(2^{13} - 1) \times 8 = 65{,}528$ bytes, which would exceed the maximum IP packet length of 65,535 bytes with the header length included (65,528 + 20 = 65,548 bytes).

Time To Live (TTL)

An eight-bit time to live field helps prevent datagrams from persisting (e.g. going in circles) on an internet. This field limits a datagram's lifetime. It is specified in seconds, but time intervals less than 1 second are rounded up to 1. In practice, the field has become a hop count—when the datagram arrives at a router, the router decrements the TTL field by one. When the TTL field hits zero, the router discards the packet and typically sends a ICMP Time Exceeded message to the sender.

The program traceroute uses these ICMP Time Exceeded messages to print the routers used by packets to go from the source to the destination.

Protocol

This field defines the protocol used in the data portion of the IP datagram. The Internet Assigned Numbers Authority maintains a list of IP protocol numbers which was originally defined in RFC 790.

Header Checksum

The 16-bit checksum field is used for error-checking of the header. When a packet arrives at a router, the router calculates the checksum of the header and compares it to the checksum field. If the values do not match, the router discards the packet. Errors in the data field must be handled by the encapsulated protocol. Both UDP and TCP have checksum fields.

When a packet arrives at a router, the router decreases the TTL field. Consequently, the router must calculate a new checksum. RFC 1071 defines the checksum calculation:

The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

For example, consider Hex 450000304422400080060000 8c7c19acae241e2b (20 bytes IP header):

Step 1) 4500 + 0030 + 4422 + 4000 + 8006 + 0000 + 8c7c + 19ac + ae24 + 1e2b = 2BBCF (16-bit sum)

Step 2) 2 + BBCF = BBD1 = 1011101111010001 (1's complement 16-bit sum)

Step 3) ~BBD1 = 0100010000101110 = 442E (1's complement of 1's complement 16-bit sum)

To validate a header's checksum the same algorithm may be used - the checksum of a header which contains a correct checksum field is a word containing all zeros (value 0):

2BBCF + 442E = 2FFFD. 2 + FFFD = FFFF. the 1'S of FFFF = 0.

Source address

This field is the IPv4 address of the sender of the packet. Note that this address may be changed in transit by a network address translation device.

Destination address

This field is the IPv4 address of the receiver of the packet. As with the source address, this may be changed in transit by a network address translation device.

Options

The options field is not often used. Note that the value in the IHL field must include enough extra 32-bit words to hold all the options (plus any padding needed to ensure that the header contains an integral number of 32-bit words). The list of options may be terminated with an EOL (End of Options List, 0x00) option; this is only necessary if the end of the options would not otherwise coincide with the end of the header. The possible options that can be put in the header are as follows:

Field	Size (bits)	Description
Copied	1	Set to 1 if the options need to be copied into all fragments of a fragmented packet.
Option Class	2	A general options category. 0 is for "*control*" options, and 2 is for "*debugging and measurement*". 1, and 3 are reserved.
Option Number	5	Specifies an option.
Option Length	8	Indicates the size of the entire option (including this field). This field may not exist for simple options.
Option Data	Variable	Option-specific data. This field may not exist for simple options.

- Note: If the header length is greater than 5, i.e. it is from 6 to 15, it means that the options field is present and must be considered.
- Note: Copied, Option Class, and Option Number are sometimes referred to as a single eight-bit field - the *Option Type*.

The following two options are discouraged because they create security concerns: Loose Source and Record Route (LSRR) and Strict Source and Record Route (SSRR). Many routers block packets containing these options.[12]

Data

The data portion of the packet is not included in the packet checksum. Its contents are interpreted based on the value of the Protocol header field.

In a typical IP implementation, standard protocols such as TCP and UDP are implemented in the OS kernel, for performance reasons. Other protocols such as ICMP may be partially implemented by the kernel, or implemented purely in user software. Protocols not implemented in-kernel, and not exposed by standard APIs such as BSD sockets, are typically implemented using a 'raw socket' API.

Some of the common protocols for the data portion are listed below:

Protocol Number	Protocol Name	Abbreviation
1	Internet Control Message Protocol	ICMP
2	Internet Group Management Protocol	IGMP
6	Transmission Control Protocol	TCP
17	User Datagram Protocol	UDP
41	IPv6 encapsulation	ENCAP
89	Open Shortest Path First	OSPF
132	Stream Control Transmission Protocol	SCTP

See List of IP protocol numbers for a complete list.

Fragmentation and reassembly

The Internet Protocol enables networks to communicate with one another. The design accommodates networks of diverse physical nature; it is independent of the underlying transmission technology used in the Link Layer. Networks with different hardware usually vary not only in transmission speed, but also in the maximum transmission unit (MTU). When one network wants to transmit datagrams to a network with a smaller MTU, it may fragment its datagrams. In IPv4, this function was placed at the Internet Layer, and is performed in IPv4 routers, which thus only

require this layer as the highest one implemented in their design.

In contrast, IPv6, the next generation of the Internet Protocol, does not allow routers to perform fragmentation; hosts must determine the path MTU before sending datagrams.

Fragmentation

When a router receives a packet, it examines the destination address and determines the outgoing interface to use. The interface has an MTU. If the packet size is bigger than the MTU, the router may fragment the packet.

The router divides the packet into segments. The max size of each segment is the MTU minus the IP header size (20 bytes minimum; 60 bytes maximum). The router puts each segment into its own packet, each fragment packet having following changes:

- The *total length* field is the segment size.
- The *more fragments* (MF) flag is set for all segments except the last one, which is set to 0.
- The *fragment offset* field is set, based on the offset of the segment in the original data payload. This is measured in units of eight-byte blocks.
- The *header checksum* field is recomputed.

For example, for an MTU of 1,500 bytes and a header size of 20 bytes, the fragment offsets would be multiples of $(1500 - 20)/8 = 185$. These multiples are 0, 185, 370, 555, 740, ...

It is possible for a packet to be fragmented at one router, and for the fragments to be fragmented at another router. For example, consider a packet with a data size of 4,500 bytes, no options, and a header size of 20 bytes. So the packet size is 4,520 bytes. Assume that the packet travels over a link with an MTU of 2,500 bytes. Then it will become two fragments:

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (bytes)
1	2500	20	2480	1	0
2	2040	20	2020	0	310

Note that the fragments preserve the data size: 2480 + 2020 = 4500.

Note how we get the offsets from the data sizes:

- 0.
- 0 + 2480/8 = 310.

Assume that these fragments reach a link with an MTU of 1,500 bytes. Each fragment will become two fragments:

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (bytes)
1	1500	20	1480	1	0
2	1020	20	1000	1	185
3	1500	20	1480	1	310
4	560	20	540	0	495

Note that the fragments preserve the data size: 1480 + 1000 = 2480, and 1480 + 540 = 2020.

Note how we get the offsets from the data sizes:

- 0.
- 0 + 1480/8 = 185
- 185 + 1000/8 = 310
- 310 + 1480/8 = 495

We can use the last offset and last data size to calculate the total data size: 495*8 + 540 = 3960 + 540 = 4500.

Reassembly

A receiver knows that a packet is a fragment if at least one of the following conditions is true:

- The "more fragments" flag is set. (This is true for all fragments except the last.)
- The "fragment offset" field is nonzero. (This is true for all fragments except the first.)

The receiver identifies matching fragments using the identification field. The receiver will reassemble the data from fragments with the same identification field using both the fragment offset and the more fragments flag. When the receiver receives the last fragment (which has the "more fragments" flag set to 0), it can calculate the length of the original data payload, by multiplying the last fragment's offset by eight, and adding the last fragment's data size. In the example above, this calculation was 495*8 + 540 = 4500 bytes.

When the receiver has all the fragments, it can put them in the correct order, by using their offsets. It can then pass their data up the stack for further processing.

Assistive protocols

The Internet Protocol is the protocol that defines and enables internetworking at the Internet Layer and thus forms the Internet. It uses a logical addressing system. IP addresses are not tied in any permanent manner to hardware identifications and, indeed, a network interface can have multiple IP addresses. Hosts and routers need additional mechanisms to identify the relationship between device interfaces and IP addresses, in order to properly deliver an IP packet to the destination host on a link. The Address Resolution Protocol (ARP) performs this IP-address-to-hardware-address translation for IPv4. (A hardware address is also called a MAC address.) In addition, the reverse correlation is often necessary. For example, when an IP host is booted or connected to a network it needs to determine its IP address, unless an address is preconfigured by an administrator. Protocols for such inverse correlations exist in the Internet Protocol Suite. Currently used methods are Dynamic Host Configuration Protocol (DHCP), Bootstrap Protocol (BOOTP) and, infrequently, reverse ARP.

Notes

[1] "INET(3) man page" (http://www.unix.com/man-page/Linux/3/inet_addr/). . Retrieved 2010-11-28.

[2] "Planning Classless Routing: TCP/IP" (http://technet.microsoft.com/en-us/library/cc779089(WS.10).aspx). Technet.microsoft.com. 2003-03-28. . Retrieved 2012-01-20.

[3] "HP Networking: switches, routers, wired, wireless, HP TippingPoint Security" (http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf). 3com.com. . Retrieved 2012-01-20.

[4] Robert Braden (October 1989). "Requirements for Internet Hosts -- Communication Layers" (http://tools.ietf.org/html/rfc1122#page-31). IETF. p. 31. RFC 1122. .

[5] Robert Braden (October 1989). "Requirements for Internet Hosts -- Communication Layers" (http://tools.ietf.org/html/rfc1122#page-66). IETF. p. 66. RFC 1122. .

[6] "World 'running out of Internet addresses'" (http://technology.inquirer.net/infotech/infotech/view/20110121-315808/World-running-out-of-Internet-addresses). . Retrieved 2011-01-23.

[7] Smith, Lucie; Lipner, Ian (3 February 2011). "Free Pool of IPv4 Address Space Depleted" (http://www.nro.net/news/ipv4-free-pool-depleted). Number Resource Organization. . Retrieved 3 February 2011.

[8] ICANN,nanog mailing list. "Five /8s allocated to RIRs - no unallocated IPv4 unicast /8s remain" (http://mailman.nanog.org/pipermail/nanog/2011-February/032107.html). .

[9] Asia-Pacific Network Information Centre (15 April 2011). "APNIC IPv4 Address Pool Reaches Final /8" (http://www.apnic.net/publications/news/2011/final-8). . Retrieved 15 April 2011.

[10] Savage, Stefan. "Practical network support for IP traceback" (http://portal.acm.org/citation.cfm?id=347057.347560). . Retrieved 2010-09-06.

[11] As an April Fools' joke, proposed for use in RFC 3514 as the "Evil bit".

[12] "Cisco unofficial FAQ" (http://www.faqs.org/faqs/cisco-networking-faq/section-23.html). . Retrieved 2012-05-10.

References

External links

- RFC 791 — Internet Protocol
- http://www.iana.org — Internet Assigned Numbers Authority (IANA)
- http://www.networksorcery.com/enp/protocol/ip.htm — IP Header Breakdown, including specific options
- RFC 3344 — IPv4 Mobility
- IPv6 vs. carrier-grade NAT/squeezing more out of IPv4 (http://www.networkworld.com/news/2010/060710-tech-argument-ipv6-nat.html)

Address exhaustion:

- RIPE report on address consumption as of October 2003 (http://www.ripe.net/rs/news/ipv4-ncc-20031030.html)
- Official current state of IPv4 /8 allocations, as maintained by IANA (http://www.iana.org/assignments/ipv4-address-space)
- Dynamically generated graphs of IPv4 address consumption with predictions of exhaustion dates — Geoff Huston (http://www.potaroo.net/tools/ipv4/index.html)
- IP addressing in China and the myth of address shortage (http://www.apnic.net/community/about-the-internet-community/internet-governance/articles/ip-addressing-in-china-2004)
- Countdown of remaining IPv4 available addresses (http://www.inetcore.com/project/ipv4ec/index_en.html) (estimated)

IPv6

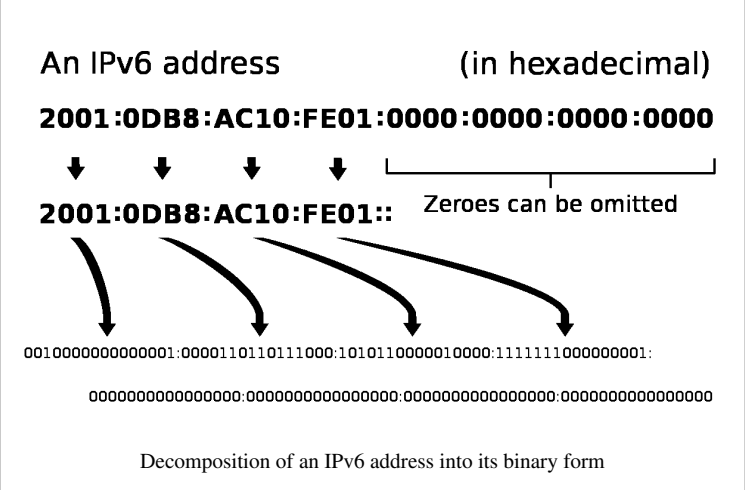
IPv6 (**Internet Protocol version 6**) is the latest revision of the Internet Protocol (IP), the primary communications protocol upon which the entire Internet is built. IPv6 is intended to replace the older IPv4, which is still employed for the vast majority of Internet traffic as of 2012.[1] IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 running out of addresses. IPv6 implements a new IP address system that allows for far more addresses to be assigned than is possible with IPv4, but as a result the two protocols are not compatible, complicating the transition to IPv6.

Each device on the Internet, such as a computer or mobile telephone, must be assigned an IP address in order to communicate with other devices. With the ever-increasing number of new devices being connected to the Internet, there is a need for more addresses than IPv4 can accommodate. IPv6 uses 128-bit addresses, allowing for 2^{128}, or approximately 34×10^{38} addresses — more than 79×10^{28} times as many as IPv4, which uses 32-bit addresses. IPv4 allows for only 4,294,967,296 unique addresses worldwide (or less than one address per person alive in 2012), but IPv6 allows for around 48×10^{28} addresses per person — a number unlikely ever to run out.

IPv6 addresses, as commonly displayed to users, consist of eight groups of four hexadecimal digits separated by colons, for example `2001:0db8:85a3:0042:0000:8a2e:0370:7334`.

The deployment of IPv6 is accelerating, with a World IPv6 Launch having taken place on 6 June 2012, in which major internet service providers, especially in countries that had been lagging in IPv6 adoption, deployed IPv6 addresses to portions of their users.[2] Data from Arbor Networks showed a peak of 0.2% of Internet traffic on IPv6 during the launch.[3]

Technical definition



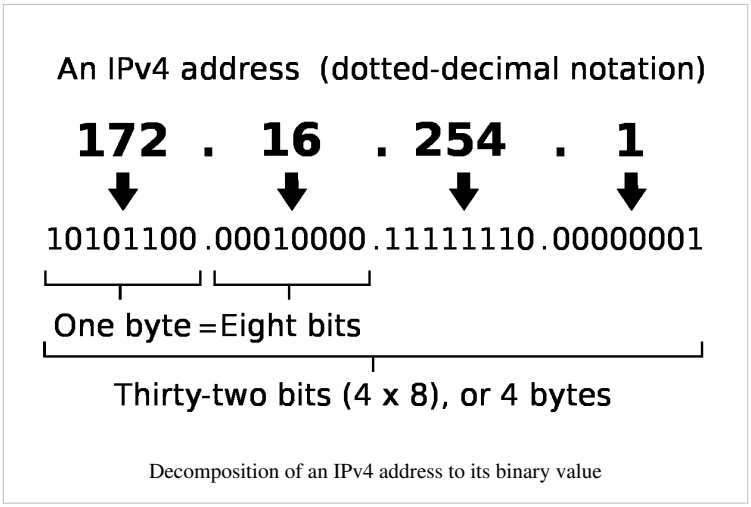
Decomposition of an IPv6 address into its binary form

IPv6, like the most commonly used IPv4 (as of 2012), is an Internet-layer protocol for packet-switched internetworking and provides end-to-end datagram transmission across multiple IP networks. It is described in Internet standard document RFC 2460, published in December 1998.[4] In addition to offering more addresses, IPv6 also implements features not present in IPv4. It simplifies aspects of address assignment (stateless address autoconfiguration), network renumbering and router announcements when changing network connectivity providers. The IPv6 subnet size has been standardized by fixing the size of the host identifier portion of an address to 64 bits to facilitate an automatic mechanism for forming the host identifier from link-layer media addressing information (MAC address). Network security is also integrated into the design of the IPv6 architecture, including the option of IPsec.

IPv6 does not implement interoperability features with IPv4, but essentially creates a parallel, independent network. Exchanging traffic between the two networks requires special translator gateways, but this is not generally required, since most computer operating systems and software implement both protocols for transparent access to both networks, either natively or using a tunneling protocol like 6to4, 6in4, or Teredo.

Motivation and origin

IPv4



Decomposition of an IPv4 address to its binary value

Internet Protocol Version 4 (IPv4) was the first publicly used version of the Internet Protocol. IPv4 addresses are typically displayed as four numbers, each in the range 0 to 255, or 8 bits per number, for a total of 32 bits. Thus IPv4 provides an addressing capability of 2^{32} or approximately 4.3 billion addresses. Address exhaustion was not initially a concern in IPv4 as this version was originally presumed to be an internal test within ARPA, and not intended for public use.

> The decision to put a 32-bit address space on there was the result of a year's battle among a bunch of engineers who couldn't make up their minds about 32, 128, or variable-length. And after a year of fighting, I said—I'm now at ARPA, I'm running the program, I'm paying for this stuff, I'm using American tax dollars, and I wanted some progress because we didn't know if this was going to work. So I said: OK, it's 32-bits. That's enough for an experiment; it's 4.3 billion terminations. Even the Defense Department doesn't need 4.3 billion of everything and couldn't afford to buy 4.3 billion edge devices to do a test anyway. So at the time I thought we were doing an experiment to

> prove the technology and that if it worked we'd have opportunity to do a production version of it. Well, it just escaped! It got out and people started to use it, and then it became a commercial thing. So this [IPv6] is the production attempt at making the network scalable.
>
> —Vint Cerf, Google IPv6 Conference 2008[5]

During the first decade of operation of the Internet (by the late 1980s), it became apparent that methods had to be developed to conserve address space. In the early 1990s, even after the redesign of the addressing system using a classless network model, it became clear that this would not suffice to prevent IPv4 address exhaustion, and that further changes to the Internet infrastructure were needed.[6]

The last available top-level (`/8`) blocks of 16 million IPv4 addresses were assigned in February 2011 by the Internet Assigned Numbers Authority (IANA) to the five Regional Internet registries (RIRs). However, many free addresses still remain within most assigned blocks, and each RIR will continue with standard address allocation policy until it is at its last `/8` block. After that, only blocks of 1024 addresses (a `/22`) will be made available from the RIR to each Local Internet registry (LIR). As of September 2012, both the Asia-Pacific Network Information Centre (APNIC) and the Réseaux IP Européens Network Coordination Centre (RIPE_NCC) had reached this stage.[7][8]

Working-group proposal

By the beginning of 1992, several proposals appeared and by the end of 1992, the IETF announced a call for white papers.[9] In September 1993, the IETF created a temporary, ad-hoc *IP Next Generation* (IPng) area to deal specifically with IPng issues. The new area was led by Allison Mankin and Scott Bradner, and had a directorate with 15 engineers from diverse backgrounds for direction-setting and preliminary document review:[6][10] The working-group members were J. Allard (Microsoft), Steve Bellovin (AT&T), Jim Bound (Digital Equipment Corporation), Ross Callon (Wellfleet), Brian Carpenter (CERN), Dave Clark (MIT), John Curran (NEARNET), Steve Deering (Xerox), Dino Farinacci (Cisco), Paul Francis (NTT), Eric Fleischmann (Boeing), Mark Knopper (Ameritech), Greg Minshall (Novell), Rob Ullmann (Lotus), and Lixia Zhang (Xerox).[11]

Logo for IPv6 released by the Internet Society

The Internet Engineering Task Force adopted the IPng model on 25 July 1994, with the formation of several IPng working groups.[6] By 1996, a series of RFCs was released defining Internet Protocol version 6 (IPv6), starting with RFC 1883. (Version 5 was used by the experimental Internet Stream Protocol.)

It is widely expected that the Internet will use IPv4 alongside IPv6 for the foreseeable future. IPv4-only and IPv6-only nodes cannot communicate directly, and need assistance from an intermediary gateway or must use other transition mechanisms.

Exhaustion of IPv4 addresses

On 3 February 2011, in a ceremony in Miami, the Internet Assigned Numbers Authority (IANA) assigned the last batch of five /8 address blocks to the Regional Internet Registries,[12] officially depleting the global pool of completely fresh blocks of addresses.[13] Each /8 address block represents approximately 16.7 million possible addresses, for a total of over 80 million potential addresses combined.

At the time, it was anticipated that these addresses could well be fully consumed within three to six months at then-current rates of allocation.[14] APNIC was the first RIR to exhaust its regional pool on 15 April 2011, except for a small amount of address space reserved for the transition to IPv6, which will be allocated in a much more restricted way.[15]

In 2003, the director of Asia-Pacific Network Information Centre (APNIC), Paul Wilson, stated that, based on then-current rates of deployment, the available space would last for one or two decades.[16] In September 2005, a report by Cisco Systems suggested that the pool of available addresses would exhaust in as little as 4 to 5 years.[17] In 2008, a policy process started for the end-game and post-exhaustion era.[18] In 2010, a daily updated report projected the global address pool exhaustion by the first quarter of 2011, and depletion at the five regional Internet registries before the end of 2011.[19]

Comparison to IPv4

IPv6 specifies a new packet format, designed to minimize packet header processing by routers.[4][20] Because the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable. However, in most respects, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP and NTPv3, where the new address format may cause conflicts with existing protocol syntax.

Larger address space

The main advantage of IPv6 over IPv4 is its larger address space. The length of an IPv6 address is 128 bits, compared to 32 bits in IPv4.[4] The address space therefore has 2^{128} or approximately 34×10^{38} addresses. By comparison, this amounts to approximately 48×10^{28} addresses for each of the seven billion people alive in 2011.[21] In addition, the IPv4 address space is poorly allocated, with approximately 14% of all available addresses utilized.[22] While these numbers are large, it wasn't the intent of the designers of the IPv6 address space to assure geographical saturation with usable addresses. Rather, the longer addresses simplify allocation of addresses, enable efficient route aggregation, and allow implementation of special addressing features. In IPv4, complex Classless Inter-Domain Routing (CIDR) methods were developed to make the best use of the small address space. The standard size of a subnet in IPv6 is 2^{64} addresses, the square of the size of the entire IPv4 address space. Thus, actual address space utilization rates will be small in IPv6, but network management and routing efficiency is improved by the large subnet space and hierarchical route aggregation.

Renumbering an existing network for a new connectivity provider with different routing prefixes is a major effort with IPv4.[23][24] With IPv6, however, changing the prefix announced by a few routers can in principle renumber an entire network, since the host identifiers (the least-significant 64 bits of an address) can be independently self-configured by a host.[25]

Multicasting

Multicasting, the transmission of a packet to multiple destinations in a single send operation, is part of the base specification in IPv6. In IPv4 this is an optional although commonly implemented feature.[26] IPv6 multicast addressing shares common features and protocols with IPv4 multicast, but also provides changes and improvements by eliminating the need for certain protocols. IPv6 does not implement traditional IP broadcast, i.e. the transmission of a packet to all hosts on the attached link using a special *broadcast address*, and therefore does not define broadcast addresses. In IPv6, the same result can be achieved by sending a packet to the link-local *all nodes* multicast group at address `ff02::1`, which is analogous to IPv4 multicast to address `224.0.0.1`. IPv6 also provides for new multicast implementations, including embedding rendezvous point addresses in an IPv6 multicast group address, which simplifies the deployment of inter-domain solutions.[27]

In IPv4 it is very difficult for an organization to get even one globally routable multicast group assignment, and the implementation of inter-domain solutions is very arcane.[28] Unicast address assignments by a local Internet registry for IPv6 have at least a 64-bit routing prefix, yielding the smallest subnet size available in IPv6 (also 64 bits). With such an assignment it is possible to embed the unicast address prefix into the IPv6 multicast address format, while still providing a 32-bit block, the least significant bits of the address, or approximately 4.2 billion multicast group identifiers. Thus each user of an IPv6 subnet automatically has available a set of globally routable source-specific multicast groups for multicast applications.[29]

Stateless address autoconfiguration (SLAAC)

IPv6 hosts can configure themselves automatically when connected to a routed IPv6 network using the Neighbor Discovery Protocol via Internet Control Message Protocol version 6 (ICMPv6) router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; if configured suitably, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters.[25]

If IPv6 stateless address autoconfiguration is unsuitable for an application, a network may use stateful configuration with the Dynamic Host Configuration Protocol version 6 (DHCPv6) or hosts may be configured statically.

Routers present a special case of requirements for address configuration, as they often are sources for autoconfiguration information, such as router and prefix advertisements. Stateless configuration for routers can be achieved with a special router renumbering protocol.[30]

Network-layer security

Internet Protocol Security (IPsec) was originally developed for IPv6, but found widespread deployment first in IPv4, into which it was back-engineered. Earlier, IPsec was an integral part of the base IPv6 protocol suite,[4][31] but has since been made optional.[32]

Simplified processing by routers

In IPv6, the packet header and the process of packet forwarding have been simplified. Although IPv6 packet headers are at least twice the size of IPv4 packet headers, packet processing by routers is generally more efficient,[4][20] thereby extending the end-to-end principle of Internet design. Specifically:

- The packet header in IPv6 is simpler than that used in IPv4, with many rarely used fields moved to separate optional header extensions.
- IPv6 routers do not perform fragmentation. IPv6 hosts are required to either perform path MTU discovery, perform end-to-end fragmentation, or to send packets no larger than the IPv6 default minimum MTU size of 1280 octets.

- The IPv6 header is not protected by a checksum; integrity protection is assumed to be assured by both link-layer and higher-layer (TCP, UDP, etc.) error detection. UDP/IPv4 may actually have a checksum of 0, indicating no checksum; IPv6 requires UDP to have its own checksum. Therefore, IPv6 routers do not need to recompute a checksum when header fields (such as the time to live (TTL) or hop count) change. This improvement may have been made less necessary by the development of routers that perform checksum computation at link speed using dedicated hardware, but it is still relevant for software-based routers.
- The *TTL* field of IPv4 has been renamed to *Hop Limit*, reflecting the fact that routers are no longer expected to compute the time a packet has spent in a queue.

Mobility

Unlike mobile IPv4, mobile IPv6 avoids triangular routing and is therefore as efficient as native IPv6. IPv6 routers may also allow entire subnets to move to a new router connection point without renumbering.[33]

Options extensibility

The IPv6 protocol header has a fixed size (40 octets). Options are implemented as additional extension headers after the IPv6 header, which limits their size only by the size of an entire packet. The extension header mechanism makes the protocol extensible in that it allows future services for quality of service, security, mobility, and others to be added without redesign of the basic protocol.[4]

Jumbograms

IPv4 limits packets to 65535 ($2^{16}-1$) octets of payload. An IPv6 node can optionally handle packets over this limit, referred to as jumbograms, which can be as large as 4294967295 ($2^{32}-1$) octets. The use of jumbograms may improve performance over high-MTU links. The use of jumbograms is indicated by the Jumbo Payload Option header.[34]

Privacy

Like IPv4, IPv6 supports globally unique static IP addresses, which can be used to track a single device's Internet activity. Most devices are used by a single user, so a device's activity is often assumed to be equivalent to a user's activity. This causes privacy concerns in the same way that cookies can also track a user's navigation through sites.

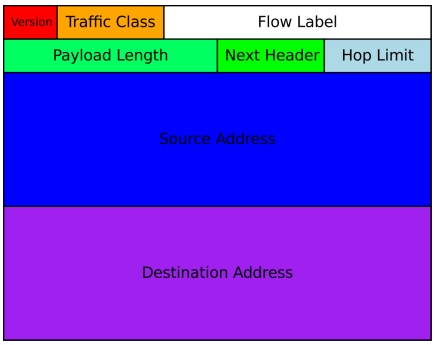
The privacy enhancements in IPv6 have been mostly developed in response to a misunderstanding.[35] Interfaces can have addresses based on the MAC address of the machine (the EUI-64 format), but this is not a requirement. Even when an address is not based on the MAC address though, the interface's address is (contrary to IPv4) usually global instead of local, which makes it much easier to identify a single user through the IP address.

Privacy extensions for IPv6 have been defined to address these privacy concerns.[36] When privacy extensions are enabled, the operating system generates ephemeral IP addresses by concatenating a randomly generated host identifier with the assigned network prefix. These ephemeral addresses, instead of trackable static IP addresses, are used to communicate with remote hosts. The use of ephemeral addresses makes it difficult to accurately track a user's Internet activity by scanning activity streams for a single IPv6 address.[37]

Privacy extensions are enabled by default in Windows, Mac OS X (since 10.7), and iOS (since version 4.3).[38] Some Linux distributions have enabled privacy extensions as well.[39]

Privacy extensions do not protect the user from other forms of activity tracking, such as tracking cookies. Privacy extensions do little to protect the user from tracking if only one or two hosts are using a given network prefix, and the activity tracker is privy to this information. In this scenario, the network prefix is the unique identifier for tracking. Network prefix tracking is less of a concern if the user's ISP assigns a dynamic network prefix via DHCP.[40][41]

Packet format



IPv6 packet header

An IPv6 packet has two parts: a header and payload.

The header consists of a fixed portion with minimal functionality required for all packets and may be followed by optional extensions to implement special features.

The fixed header occupies the first 40 octets (320 bits) of the IPv6 packet. It contains the source and destination addresses, traffic classification options, a hop counter, and the type of the optional extension or payload which follows the header. This *Next Header* field tells the receiver how to interpret the data which follows the header. If the packet contains options, this field contains the option type of the next option. The "Next Header" field of the last option, points to the upper-layer protocol that is carried in the packet's payload.

Extension headers carry options that are used for special treatment of a packet in the network, e.g., for routing, fragmentation, and for security using the IPsec framework.

Without special options, a payload must be less than 64kB. With a Jumbo Payload option (in a *Hop-By-Hop Options* extension header), the payload must be less than 4 GB.

Unlike in IPv4, routers never fragment a packet. Hosts are expected to use Path MTU Discovery to make their packets small enough to reach the destination without needing to be fragmented. See IPv6 Packet#Fragmentation.

Addressing

Compared to IPv4, the most obvious advantage of IPv6 is its larger address space. IPv4 addresses are 32 bits long and number about 43×10^9 (4.3 billion).[42] IPv6 addresses are 128 bits long and number about 34×10^{38} (340 Undecillion). IPv6's addresses are deemed enough for the foreseeable future.[43]

IPv6 addresses are written in eight groups of four hexadecimal digits separated by colons, such as `2001:0db8:85a3:0000:0000:8a2e:0370:7334`. IPv6 unicast addresses other than those that start with binary 000 are logically divided into two parts: a 64-bit (sub-)network prefix, and a 64-bit interface identifier.[44]

For stateless address autoconfiguration (SLAAC) to work, subnets require a /64 address block, as defined in RFC 4291 section 2.5.1. Local Internet registries get assigned at least /32 blocks, which they divide among ISPs.[45] The obsolete RFC 3177 recommended the assignment of a /48 to end-consumer sites. This was replaced by RFC 6177, which "recommends giving home sites significantly more than a single /64, but does not recommend that every home site be given a /48 either". /56s are specifically considered. It remains to be seen if ISPs will honor this recommendation; for example, during initial trials, Comcast customers were given a single /64 network.[46]

IPv6 addresses are classified by three types of networking methodologies: unicast addresses identify each network interface, anycast addresses identify a group of interfaces, usually at different locations of which the nearest one is automatically selected, and multicast addresses are used to deliver one packet to many interfaces. The broadcast method is not implemented in IPv6. Each IPv6 address has a scope, which specifies in which part of the network it is valid and unique. Some addresses are unique only on the local (sub-)network. Others are globally unique.

Some IPv6 addresses are reserved for special purposes, such as loopback, 6to4 tunneling, and Teredo tunneling. See RFC 5156. Also, some address ranges are considered special, such as link-local addresses for use on the local link only, Unique Local addresses (ULA) as described in RFC 4193, and solicited-node multicast addresses used in the Neighbor Discovery Protocol.

IPv6 in the Domain Name System

In the Domain Name System, hostnames are mapped to IPv6 addresses by *AAAA* resource records, so-called *quad-A* records. For reverse resolution, the IETF reserved the domain `ip6.arpa`, where the name space is hierarchically divided by the 1-digit hexadecimal representation of nibble units (4 bits) of the IPv6 address. This scheme is defined in RFC 3596.

Address format

An IPv6 address is represented by 8 groups of 16-bit values, each group represented as 4 hexadecimal digits and separated by colons (:). For example:

`2001:0db8:0000:0000:0000:ff00:0042:8329`

The hexadecimal digits are not case-sensitive; e.g., the groups `0DB8` and `0db8` are equivalent.

An IPv6 address may be abbreviated by using one or more of the following rules:

1. Remove one or more leading zeroes from one or more groups of hexadecimal digits; this is usually done to either all or none of the leading zeroes. (For example, convert the group `0042` to `42`.)
2. Omit one or more consecutive sections of zeroes, using a double colon (::) to denote the omitted sections. The double colon may only be used once in any given address, as the address would be indeterminate if the double colon was used multiple times. (For example, `2001:db8::1:2` is valid, but `2001:db8::1::2` is not permitted.)

Below is an example of these rules:

Address	2001	:	0db8	:	0000	:	0000	:	0000	:	ff00	:	0042	:	8329
With Rule 1 applied to its fullest extent (all leading zeroes removed)	2001	:	db8	:	0	:	0	:	0	:	ff00	:	42	:	8329
With Rule 2 applied to its fullest extent (the most consecutive sections of zeroes omitted)	2001	:	0db8	:						:	ff00	:	0042	:	8329
With the above 2 actions combined	2001	:	db8	:						:	ff00	:	42	:	8329

Below are the text representations of these addresses:

Initial address: `2001:0db8:0000:0000:0000:ff00:0042:8329`

After removing all leading zeroes: `2001:db8:0:0:0:ff00:42:8329`

After omitting consecutive sections of zeroes: `2001:0db8::ff00:0042:8329`

After doing both: `2001:db8::ff00:42:8329`

Another example is the loopback address, which can be abbreviated to `::1` by using both rules above:[42]

Initial address: `0000:0000:0000:0000:0000:0000:0000:0001`

After removing all leading zeroes: `0:0:0:0:0:0:0:1`

After omitting consecutive sections of zeroes: `::0001`

After doing both: `::1`

As IPv6 addresses may have more than one representation, which can lead to confusion; there is a proposed standard for representing them in text.[47]

Transition mechanisms

Until IPv6 completely supplants IPv4, a number of transition mechanisms[48] are needed to enable IPv6-only hosts to reach IPv4 services and to allow isolated IPv6 hosts and networks to reach each-other over IPv4-only infrastructure.

Many of these transition mechanisms use tunneling to encapsulate IPv6 traffic within IPv4 networks. This is an imperfect solution, which may increase latency and cause problems with Path MTU Discovery.[49] Tunneling protocols are a temporary solution for networks that do not support native dual-stack, where both IPv6 and IPv4 run independently.

Dual IP stack implementation

Dual-stack (or "native dual-stack") refers to side-by-side implementation of IPv4 and IPv6. That is, both protocols run on the same network infrastructure, and there's no need to encapsulate IPv6 inside IPv4 (using tunneling) or vice-versa. Dual-stack is defined in RFC 4213.[50]

Although this is the most desired IPv6 implementation, as it avoids the complexities and pitfalls of tunneling, it is not always possible, since outdated network equipment may not support IPv6. A good example is cable TV-based internet access. In modern cable TV networks, the core of the HFC network (such as large core routers) are likely to support IPv6. However, other network equipment (such as a CMTS) or customer equipment (like cable modems) may require software updates or hardware upgrades to support IPv6. This means cable network operators must resort to tunneling until the backbone equipment supports native dual-stack.

Tunneling

Because not all networks support dual-stack, tunneling is used for IPv4 networks to talk to IPv6 networks (and vice-versa). Many current internet users do not have IPv6 dual-stack support, and thus cannot reach IPv6 sites directly. Instead, they must use IPv4 infrastructure to carry IPv6 packets. This is done using a technique known as *tunneling*, which encapsulates IPv6 packets within IPv4, in effect using IPv4 as a link layer for IPv6.

IP protocol 41 indicates IPv4 packets which encapsulate IPv6 datagrams. Some routers or network address translation devices may block protocol 41. To pass through these devices, you might use UDP packets to encapsulate IPv6 datagrams. Other encapsulation schemes, such as AYIYA or Generic Routing Encapsulation, are also popular.

Conversely, on IPv6-only internet links, when access to IPv4 network facilities is needed, tunneling of IPv4 over IPv6 protocol occurs, using the IPv6 as a link layer for IPv4.

Automatic tunneling

Automatic tunneling refers to a technique where the routing infrastructure automatically determines the tunnel endpoints. Some automatic tunneling techniques are below.

6to4 is recommended by RFC 3056. It uses protocol 41 encapsulation.[51] Tunnel endpoints are determined by using a well-known IPv4 anycast address on the remote side, and embedding IPv4 address information within IPv6 addresses on the local side. 6to4 is the most common tunnel protocol currently deployed.

Teredo is an automatic tunneling technique that uses UDP encapsulation and can allegedly cross multiple NAT boxes.[52] IPv6, including 6to4 and Teredo tunneling, are enabled by default in Windows Vista[53] and Windows 7. Most Unix systems implement only 6to4, but Teredo can be provided by third-party software such as Miredo.

ISATAP[54] treats the IPv4 network as a virtual IPv6 local link, with mappings from each IPv4 address to a link-local IPv6 address. Unlike 6to4 and Teredo, which are *inter-site* tunnelling mechanisms, ISATAP is an *intra-site* mechanism, meaning that it is designed to provide IPv6 connectivity between nodes within a single organisation.

Configured and automated tunneling (6in4)

In *configured tunneling*, the tunnel endpoints are explicitly configured, either by an administrator manually or the operating system's configuration mechanisms, or by an automatic service known as a tunnel broker;[55] this is also referred to as *automated tunneling*. Configured tunneling is usually more deterministic and easier to debug than automatic tunneling, and is therefore recommended for large, well-administered networks. Automated tunneling provides a compromise between the ease of use of automatic tunneling and the deterministic behaviour of configured tunneling.

Raw encapsulation of IPv6 packets using IPv4 protocol number 41 is recommended for configured tunneling; this is sometimes known as 6in4 tunneling. As with automatic tunneling, encapsulation within UDP may be used in order to cross NAT boxes and firewalls.

Proxying and translation for IPv6-only hosts

After the regional Internet registries have exhausted their pools of available IPv4 addresses, it is likely that hosts newly added to the Internet might only have IPv6 connectivity. For these clients to have backward-compatible connectivity to existing IPv4-only resources, suitable IPv6 transition mechanisms must be deployed.

One form of address translation is the use of a dual-stack application-layer proxy server, for example a web proxy.

NAT-like techniques for application-agnostic translation at the lower layers in routers and gateways have been proposed. The NAT-PT standard was dropped due to a number of criticisms,[56] however more recently the continued low adoption of IPv6 has prompted a new standardization effort under the name NAT64.

IPv6 readiness

Compatibility with IPv6 networking is mainly a software or firmware issue. However, much of the older hardware that could in principle be upgraded is likely to be replaced instead. The American Registry for Internet Numbers (ARIN) suggested that all Internet servers be prepared to serve IPv6-only clients by January 2012.[57] Sites will only be accessible over NAT64 if they do not use IPv4 literals as well.

Software

Applications can be IPv4 only, IPv6 only, dual-stack, or hybrid dual-stack. Most personal computers running recent operating system versions are IPv6-ready. Many popular applications with network capabilities are ready, and most others could be easily upgraded with help from the developers.

Some software transitioning mechanisms are outlined in RFC 4038, RFC 3493, and RFC 3542.

IPv4-mapped IPv6 addresses

Hybrid dual-stack IPv6/IPv4 implementations recognize a special class of addresses, the IPv4-mapped IPv6 addresses. In these addresses, the first 80 bits are zero, the next 16 bits are one, and the remaining 32 bits are the IPv4 address. You may see these addresses with the first 96 bits written in the standard IPv6 format, and the remaining 32 bits written in the customary dot-decimal notation of IPv4. For example, `::ffff:192.0.2.128` represents the IPv4 address `192.0.2.128`. A deprecated format for IPv4-compatible IPv6 addresses was `::192.0.2.128`.[58]

Because of the significant internal differences between IPv4 and IPv6, some of the lower-level functionality available to programmers in the IPv6 stack does not work identically with IPv4-mapped addresses. Some common IPv6 stacks do not implement the IPv4-mapped address feature, either because the IPv6 and IPv4 stacks are separate implementations (e.g., Microsoft Windows 2000, XP, and Server 2003), or because of security concerns (OpenBSD).[59] On these operating systems, a program must open a separate socket for each IP protocol it uses. On some systems, e.g., the Linux kernel, NetBSD, and FreeBSD, this feature is controlled by the socket option

`IPV6_V6ONLY`, as specified in RFC 3493.[60]

Hardware and embedded systems

Low-level equipment such as network adapters and network switches may not be affected by the change, since they transmit link-layer frames without inspecting the contents. However, networking devices that obtain IP addresses or perform routing of IP packets do need to understand IPv6.

Most equipment would be IPv6 capable with a software or firmware update if the device has sufficient storage and memory space for the new IPv6 stack. However, manufacturers may be reluctant to spend on software development costs for hardware they have already sold when they are poised for new sales from IPv6-ready equipment.

In some cases, non-compliant equipment needs to be replaced because the manufacturer no longer exists or software updates are not possible, for example, because the network stack is implemented in permanent read-only memory.

The CableLabs consortium published the 160 Mbit/s DOCSIS 3.0 IPv6-ready specification for cable modems in August 2006. The widely used DOCSIS 2.0 does not support IPv6. The new 'DOCSIS 2.0 + IPv6' standard supports IPv6, which may on the cable modem side require only a firmware upgrade.[61][62] It is expected that only 60% of cable modems' servers and 40% of cable modems will be DOCSIS 3.0 by 2011.[63] However, most ISPs that support DOCSIS 3.0 do not support IPv6 across their networks.

Other equipment which is typically not IPv6-ready ranges from Voice over Internet Protocol devices to laboratory equipment and printers.

Deployment

The introduction of Classless Inter-Domain Routing (CIDR) in the Internet routing and IP address allocation methods in 1993 and the extensive use of network address translation (NAT) delayed the inevitable IPv4 address exhaustion, but the final phase of exhaustion started on 3 February 2011.[19] However, despite a decade long development and implementation history as a Standards Track protocol, general worldwide deployment is still in its infancy. As of October 2011, about 3% of domain names and 12% of the networks on the internet have IPv6 protocol support.[64]

IPv6 has been implemented on all major operating systems in use in commercial, business, and home consumer environments. Since 2008, the domain name system can be used in IPv6. IPv6 was first used in a major world event during the 2008 Summer Olympic Games,[65] the largest showcase of IPv6 technology since the inception of IPv6.[66] Countries like China and the Federal U.S. Government are also starting to require IPv6 capability on their equipment.

In 2010, Verizon mandated IPv6 operation and deprecated IPv4 as an optional capability for cellular hardware.[67] T-Mobile USA followed suit. As of June 2012, T-Mobile supports external IPv6 access.[68]

References

[1] David Frost (20 April 2011). "Ipv6 traffic volumes going backwards" (http://www.itwire.com/business-it-news/networking/46689-ipv6-traffic-volumes-going-backwards). iTWire. . Retrieved 19 February 2012.

[2] Goldman, David. "The Internet now has 340 trillion trillion trillion addresses" (http://money.cnn.com/2012/06/06/technology/ipv6/index.htm). CNN. . Retrieved 23 June 2012.

[3] Graph of IPv6 share from May 15th to June 14th, 2012 (http://ddos.arbornetworks.com/uploads/2012/06/IJ13.jpg) at arbornetworks.com

[4] RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden (December 1998)

[5] *Google IPv6 Conference 2008: What will the IPv6 Internet look like?* (http://www.youtube.com/watch?v=mZo69JQoLb8). Event occurs at 13:35. .

[6] RFC 1752 *The Recommendation for the IP Next Generation Protocol*, S. Bradner, A. Mankin, January 1995.

[7] Rashid, Fahmida. "IPv4 Address Exhaustion Not Instant Cause for Concern with IPv6 in Wings" (http://www.eweek.com/c/a/IT-Infrastructure/IPv4-Address-Exhaustion-Not-Instant-Cause-for-Concern-with-IPv6-in-Wings-287643/). eWeek. . Retrieved 23 June 2012.

[8] Ward, Mark. "Europe hits old internet address limits" (http://www.bbc.co.uk/news/technology-19600718). BBC. . Retrieved 15 September 2012.
[9] RFC 1550, *IP: Next Generation (IPng) White Paper Solicitation*, S. Bradner, A. Mankin (December 1993)
[10] "History of the IPng Effort" (http://playground.sun.com/ipv6/doc/history.html). Sun. .
[11] The Internet Engineering Task Force. Apendix B. http://tools.ietf.org/html/rfc1752#appendix-B
[12] "River of IPv4 addresses officially runs dry" (http://arstechnica.com/tech-policy/news/2011/02/river-of-ipv4-addresses-officially-runs-dry). arsttechnica.com. .
[13] Rashid, Fahmida Y. (3 February 2011). "IPv4 Address Depletion Adds Momentum to IPv6 Transition" (http://www.eweek.com/c/a/IT-Infrastructure/IPv4-Address-Depletion-Adds-Momentum-to-IPv6-Transition-875751/). eWeek.com. . Retrieved 3 February 2011.
[14] "Two /8s allocated to APNIC from IANA" (http://www.apnic.net/publications/news/2011/delegation). APNIC. 1 January 2010. . Retrieved 3 February 2011.
[15] Asia-Pacific Network Information Centre (15 April 2011). "APNIC IPv4 Address Pool Reaches Final /8" (http://www.apnic.net/publications/news/2011/final-8). . Retrieved 15 April 2011.
[16] Exec: No shortage of Net addresses By John Lui, CNETAsia (http://news.zdnet.com/2100-1009_22-1020653.html)
[17] "A Pragmatic Report on IPv4 Address Space Consumption by Tony Hain, Cisco Systems" (http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-3/ipv4.html). Cisco.com. 1 July 2005. . Retrieved 19 February 2012.
[18] Proposed Global Policy for the Allocation of the Remaining IPv4 Address Space (http://www.ripe.net/ripe/policies/proposals/2008-03.html)
[19] "IPv4 Address Report" (http://www.potaroo.net/tools/ipv4/). Potaroo.net. . Retrieved 20 January 2012.
[20] RFC 1726, *Technical Criteria for Choosing IP The Next Generation (IPng)*, Partridge C., Kastenholz F. (December 1994)
[21] "U.S. Census Bureau" (http://www.census.gov/main/www/popclock.html). Census.gov. . Retrieved 20 January 2012.
[22] "Moving to IPv6: Now for the hard part (FAQ)" (http://news.cnet.com/8301-30685_3-20030482-264.html). *Deep Tech*. CNET News. . Retrieved 3 February 2011.
[23] RFC 2071, *Network Renumbering Overview: Why would I want it and what is it anyway?*, P. Ferguson, H. Berkowitz (January 1997)
[24] RFC 2072, *Router Renumbering Guide*, H. Berkowitz (January 1997)
[25] RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thomson, T. Narten, T. Jinmei (September 2007)
[26] RFC 1112, *Host extensions for IP multicasting*, S. Deering (August 1989)
[27] RFC 3956, *Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address*, P. Savola, B. Haberman (November 2004)
[28] RFC 2908, *The Internet Multicast Address Allocation Architecture*, D. Thaler, M. Handley, D. Estrin (September 2000)
[29] RFC 3306, *Unicast-Prefix-based IPv6 Multicast Addresses*, B. Haberman, D. Thaler (August 2002)
[30] RFC 2894, *Router Renumbering for IPv6*, M. Crawford, August 2000.
[31] RFC 4301, *IPv6 Node Requirements", J. Loughney (April 2006)*
[32] RFC 6434, "IPv6 Node Requirements", E. Jankiewicz, J. Loughney, T. Narten (December 2011)
[33] RFC 3963, *Network Mobility (NEMO) Basic Protocol Support*, V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert (January 2005)
[34] RFC 2675, *IPv6 Jumbograms*, D. Borman, S. Deering, R. Hinden (August 1999)
[35] IPv6 Essentials by Silvia Hagen, p. 28, chapter 3.5.
[36] T. Narten, R. Draves (2001-01). "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" (http://www.ietf.org/rfc/rfc3041.txt). .
[37] Privacy Extensions (IPv6) (http://www.elektronik-kompendium.de/sites/net/1601271.htm), Elektronik Kompendium.
[38] IPv6: Privacy Extensions einschalten (http://www.heise.de/netze/artikel/IPv6-Privacy-Extensions-einschalten-1204783.html), Reiko Kaps, 13 April 2011
[39] "Comment #61 : Bug #176125 : Bugs: "procps" package: Ubuntu" (https://bugs.launchpad.net/ubuntu/+source/procps/+bug/176125/comments/61). Bugs.launchpad.net. . Retrieved 19 February 2012.
[40] Statement on IPv6 Address Privacy (ftp://ftp.cuhk.edu.hk/pub/doc/ipng/html/ipv6-address-privacy.html), Steve Deering & Bob Hinden, Co-Chairs of the IETF's IP Next Generation Working Group, 6 November 1999.
[41] "Neues Internet-Protokoll erschwert anonymes Surfen" (http://www.spiegel.de/netzwelt/web/0,1518,729340,00.html). Spiegel.de. . Retrieved 19 February 2012.
[42] RFC 4291 *IP Version 6 Addressing Architecture*, R. Hinden, S. Deering (February 2006)
[43] "The sheer size of IPv6" (http://pthree.org/2009/03/08/the-sheer-size-of-ipv6/). Pthree.org. 8 March 2009. . Retrieved 20 January 2012.
[44] RFC 4291 p. 9
[45] "IPv6 Address Allocation and Assignment Policy" (http://www.ripe.net/ripe/docs/ripe-512). RIPE NCC. 8 February 2011. . Retrieved 27 March 2011.
[46] "Comcast Activates First Users With IPv6 Native Dual Stack Over DOCSIS" (http://blog.comcast.com/2011/01/comcast-activates-first-users-with-ipv6-native-dual-stack-over-docsis.html). Comcast. 31 January 2011. .
[47] RFC 5952, *A Recommendation for IPv6 Address Text Representation*, S. Kawamura (August 2010)
[48] "IPv6 Transition Mechanism / Tunneling Comparison" (http://www.sixxs.net/faq/connectivity/?faq=comparison). Sixxs.net. . Retrieved 20 January 2012.
[49] "RFC 6343 – Advisory Guidelines for 6to4 Deployment" (http://tools.ietf.org/html/rfc6343). Tools.ietf.org. . Retrieved 20 August 2012.

[50] "RFC 4213 – Basic Transition Mechanisms for IPv6 Hosts and Routers" (http://tools.ietf.org/html/rfc4213). Tools.ietf.org. . Retrieved 20 August 2012.
[51] RFC 3056 *Connection of IPv6 Domains via IPv4 Clouds*, B. Carpenter, Februari 2001.
[52] RFC 4380 *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*, C. Huitema, Februari 2006
[53] "The Windows Vista Developer Story: Application Compatibility Cookbook" (http://msdn2.microsoft.com/en-us/library/aa480152.aspx). Msdn2.microsoft.com. 24 April 2006. . Retrieved 20 January 2012.
[54] RFC 5214 *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*, F. Templin, T. Gleeson, D. Thaler, March 2008.
[55] RFC 3053, *IPv6 Tunnel Broker*, A. Durand, P. Fasano, I. Guardini, D. Lento (January 2001)
[56] RFC 4966 Reasons to Move the Network Address Translator – Protocol Translator (NAT-PT) to Historic Status
[57] *Web sites must support IPv6 by 2012, expert warns* (http://www.networkworld.com/news/2010/012110-ipv6-warning.html). Network World. 21 January 2010. . Retrieved 30 September 2010.
[58] "RFC4291" (http://tools.ietf.org/html/rfc4291). Tools.ietf.org. . Retrieved 20 January 2012.
[59] "OpenBSD inet6(4) manual page" (http://www.openbsd.org/cgi-bin/man.cgi?query=inet6&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html#PROTOCOLS). Openbsd.org. 13 December 2009. . Retrieved 20 January 2012.
[60] "RFC 3493 – Basic Socket Interface Extensions for IPv6" (http://tools.ietf.org/html/rfc3493#page-22). Tools.ietf.org. . Retrieved 20 January 2012.
[61] "DOCSIS 2.0 Interface" (http://www.cablemodem.com/specifications/specifications20.html). Cablemodem.com. 29 October 2007. . Retrieved 31 August 2009.
[62] "RMV6TF.org" (http://rmv6tf.org/2008-IPv6-Summit-Presentations/Dan Torbet - IPv6andCablev2.pdf) (PDF). . Retrieved 20 January 2012.
[63] "DOCSIS 3.0 Network Equipment Penetration to Reach 60% by 2011" (http://www.abiresearch.com/abiprdisplay.jsp?pressid=710) (Press release). ABI Research. 23 August 2007. . Retrieved 30 September 2007.
[64] Mike Leber (2 October 2010). "Global IPv6 Deployment Progress Report" (http://bgp.he.net/ipv6-progress-report.cgi). Hurricane Electric. . Retrieved 19 October 2011.
[65] "Beijing2008.cn leaps to next-generation Net" (http://en.beijing2008.cn/news/official/preparation/n214384681.shtml) (Press release). The Beijing Organizing Committee for the Games of the XXIX Olympiad. 30 May 2008. .
[66] Das, Kaushik (2008). "IPv6 and the 2008 Beijing Olympics" (http://ipv6.com/articles/general/IPv6-Olympics-2008.htm). *IPv6.com*. . Retrieved 15 August 2008. "As thousands of engineers, technologists have worked for a significant time to perfect this (IPv6) technology, there is no doubt, this technology brings considerable promises but this is for the first time that it will showcase its strength when in use for such a mega-event."
[67] Derek Morr (9 June 2009). "Verizon Mandates IPv6 Support for Next-Gen Cell Phones" (http://www.circleid.com/posts/20090609_verizon_mandates_ipv6_support_for_next_gen_cell_phones/). CircleID. .
[68] theipv6guy (31 July 2012). "T-Mobile USA Launches External IPv6" (https://support.t-mobile.com/thread/27171). T-Mobile. .

External links

- IPv6 (http://www.dmoz.org/Computers/Internet/Protocols/IP/IPv6/) at the Open Directory Project
- Why IPv6 matters to radio stations (http://radioworld.com/article/why-ipv6-matters-to-your-station/23533)
- Security implications of implementing IPv6 (https://cert.inteco.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_security_implications_ipv6.pdf)
- Free Pool of IPv4 Address Space Depleted (https://www.nro.net/news/ipv4-free-pool-depleted)
- An article about IPv6 in Linux by Rami Rosen (http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/IPv6-in-Linux/)
- An Introduction and Statistics about IPV6 (https://www.google.com/intl/en/ipv6/)
- Google IPv6 check (https://ipv6test.google.com/)
- IPv6 Introduction and Configuration (https://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4776.html?OpenDocument) by IBM Redbooks

Network address translation

In computer networking, **network address translation** (**NAT**) is the process of modifying IP address information in IP packet headers while in transit across a traffic routing device.

The simplest type of NAT provides a one-to-one translation of IP addresses. RFC 2663 refers to this type of NAT as **basic NAT**. It is often also referred to as **one-to-one NAT**. In this type of NAT only the IP addresses, IP header checksum and any higher level checksums that include the IP address need to be changed. The rest of the packet can be left untouched (at least for basic TCP/UDP functionality, some higher level protocols may need further translation). Basic NATs can be used when there is a requirement to interconnect two IP networks with incompatible addressing.

However it is common to hide an entire IP address space, usually consisting of private IP addresses, behind a single IP address (or in some cases a small group of IP addresses) in another (usually public) address space. To avoid ambiguity in the handling of returned packets, a one-to-many NAT must alter higher level information such as TCP/UDP ports in outgoing communications and must maintain a translation table so that return packets can be correctly translated back. RFC 2663 uses the term **NAPT** (**network address and port translation**) for this type of NAT. Other names include **PAT** (**port address translation**), **IP masquerading**, **NAT Overload** and **many-to-one NAT**. Since this is the most common type of NAT it is often referred to simply as NAT.

As described, the method enables communication through the router only when the conversation originates in the masqueraded network, since this establishes the translation tables. For example, a web browser in the masqueraded network can browse a website outside, but a web browser outside could not browse a web site in the masqueraded network. However, most NAT devices today allow the network administrator to configure translation table entries for permanent use. This feature is often referred to as "static NAT" or port forwarding and allows traffic originating in the "outside" network to reach designated hosts in the masqueraded network.

In the mid-1990s NAT became a popular tool for alleviating the consequences of IPv4 address exhaustion.[1] It has become a common, indispensable feature in routers for home and small-office Internet connections. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address.

Network address translation has serious drawbacks on the quality of Internet connectivity and requires careful attention to the details of its implementation. In particular, all types of NAT break the originally envisioned model of IP end-to-end connectivity across the Internet and NAPT makes it difficult for systems behind a NAT to accept incoming communications. As a result, NAT traversal methods have been devised to alleviate the issues encountered.

One-to-many NATs

The majority of NATs map multiple private hosts to one publicly exposed IP address. In a typical configuration, a local network uses one of the designated "private" IP address subnets (RFC 1918). A router on that network has a private address in that address space. The router is also connected to the Internet with a "public" address assigned by an Internet service provider. As traffic passes from the local network to the Internet, the source address in each packet is translated on the fly from a private address to the public address. The router tracks basic data about each active connection (particularly the destination address and port). When a reply returns to the router, it uses the connection tracking data it stored during the outbound phase to determine the private address on the internal network to which to forward the reply.

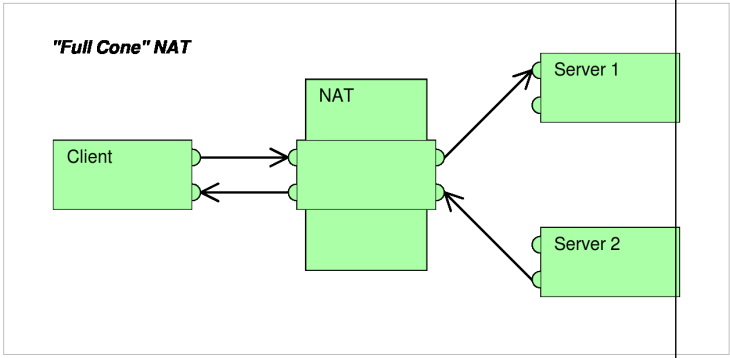
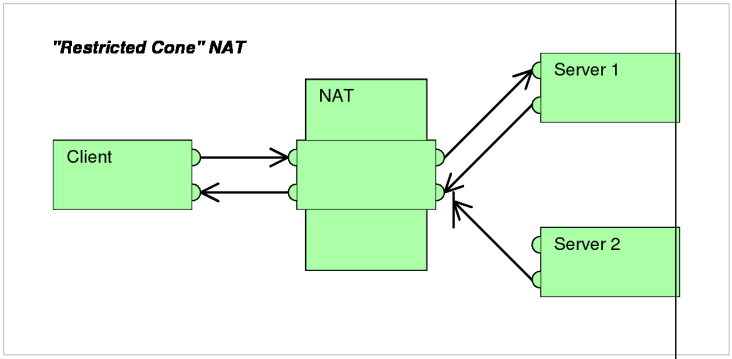
All Internet packets have a source IP address and a destination IP address. Typically packets passing from the private network to the public network will have their source address modified while packets passing from the public network back to the private network will have their destination address modified. More complex configurations are also

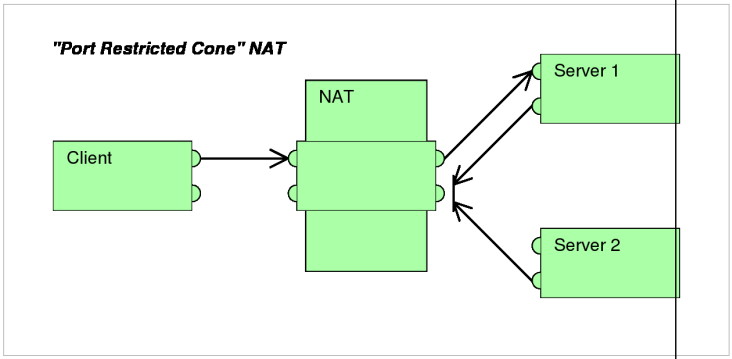
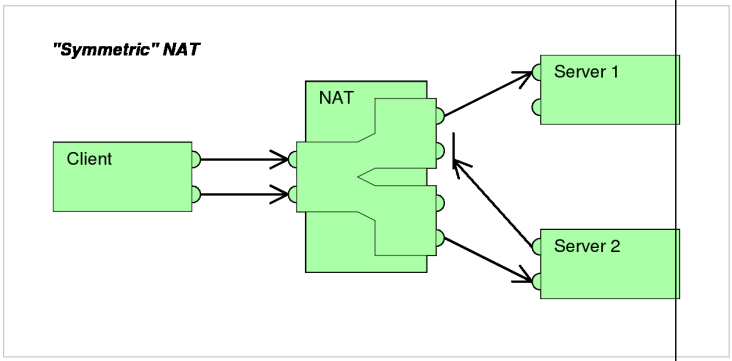
possible.

To avoid ambiguity in how to translate returned packets, further modifications to the packets are required. The vast bulk of Internet traffic is TCP and UDP packets, and for these protocols the port numbers are changed so that the combination of IP and port information on the returned packet can be unambiguously mapped to the corresponding private address and port information. Protocols not based on TCP or UDP require other translation techniques. ICMP packets typically relate to an existing connection and need to be mapped using the same IP and port mappings as that connection.

Methods of Port translation

There are several ways of implementing network address and port translation. In some application protocols that use IP address information, the application running on a node in the masqueraded network needs to determine the external address of the NAT, i.e., the address that its communication peers detect, and, furthermore, often needs to examine and categorize the type of mapping in use. Usually this is done because it is desired to set up a direct communications path (either to save the cost of taking the data via a server or to improve performance) between two clients both of which are behind separate NATs. For this purpose, the Simple traversal of UDP over NATs (STUN) protocol was developed (RFC 3489, March 2003). It classified NAT implementation as *full cone NAT*, *(address) restricted cone NAT*, *port restricted cone NAT* or *symmetric NAT* and proposed a methodology for testing a device accordingly. However, these procedures have since been deprecated from standards status, as the methods have proven faulty and inadequate to correctly assess many devices. New methods have been standardized in RFC 5389 (October 2008) and the STUN acronym now represents the new title of the specification: *Session Traversal Utilities for NAT*.

Full-cone NAT, also known as *one-to-one NAT* • Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort. • *Any external host* can send packets to iAddr:iPort by sending packets to eAddr:ePort.	
(Address) restricted cone NAT • Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort. • An external host (*hAddr:any*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously sent a packet to hAddr:*any*. "Any" means the port number doesn't matter.	

Port-restricted cone NAT Like an address restricted cone NAT, but the restriction includes port numbers. • Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort. • An external host (*hAddr:hPort*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously sent a packet to hAddr:hPort.	
Symmetric NAT • Each request from the same internal IP address and port to a specific destination IP address and port is mapped to a unique external source IP address and port, if the same internal host sends a packet even with the same source address and port but to a different destination, a different mapping is used. • Only an external host that receives a packet from an internal host can send a packet back.	

This terminology has been the source of much confusion, as it has proven inadequate at describing real-life NAT behavior.[2] Many NAT implementations combine these types, and it is therefore better to refer to specific individual NAT behaviors instead of using the Cone/Symmetric terminology. Especially, most NAT translators combine *symmetric NAT* for outgoing connections with *static port mapping*, where incoming packets to the external address and port are redirected to a specific internal address and port. Some products can redirect packets to several internal hosts, e.g. to divide the load between a few servers. However, this introduces problems with more sophisticated communications that have many interconnected packets, and thus is rarely used.

Type of NAT and NAT Traversal

The NAT traversal problem arises when two peers behind distinct NAT try to communicate. One way to solve this problem is to use port forwarding, another way is to use various NAT traversal techniques. The most popular technique for TCP NAT traversal is TCP hole punching, which requires the NAT to follow the *port preservation* design for TCP, as explained below.

Many NAT implementations follow the *port preservation* design especially for TCP, which is to say that they use the same values as internal and external port numbers. NAT *port preservation* for outgoing TCP connections is especially important for TCP NAT traversal, because programs usually bind distinct TCP sockets to ephemeral ports for distinct TCP connections, rendering NAT port prediction impossible for TCP.

On the other hand, for UDP, NATs do not need to have *port preservation* because applications usually reuse the same UDP socket to send packets to distinct hosts, making port prediction straightforward, as it is the same source port for each packet.

Furthermore, *port preservation* in NAT for TCP allows P2P protocols to offer less complexity and less latency because there is no need to use a third party to discover the NAT port since the application already knows the NAT port.[3]

However, if two internal hosts attempt to communicate with the same external host using the same port number, the external port number used by the second host will be chosen at random. Such NAT will be sometimes perceived as *(address) restricted cone NAT* and other times as *symmetric NAT*.

Recent studies have shown that roughly 70% of clients in P2P networks employ some form of NAT.[4]

Implementation

Establishing Two-Way Communication

Every TCP and UDP packet contains both a source IP address and source port number as well as a destination IP address and destination port number. The port address/IP address pair forms a socket. In particular, the source port and source IP address form the source socket.

For publicly accessible services such as web servers and mail servers the port number is important. For example, port 80 connects to the web server software and port 25 to a mail server's SMTP daemon. The IP address of a public server is also important, similar in global uniqueness to a postal address or telephone number. Both IP address and port must be correctly known by all hosts wishing to successfully communicate.

Private IP addresses as described in RFC 1918 are significant only on private networks where they are used, which is also true for host ports. Ports are unique endpoints of communication on a host, so a connection through the NAT device is maintained by the combined mapping of port and IP address.

PAT (Port Address Translation) resolves conflicts that would arise through two different hosts using the same source port number to establish unique connections at the same time.

An Analogy

A NAT device is similar to a phone system at an office that has one public telephone number and multiple extensions. Outbound phone calls made from the office all appear to come from the same telephone number. However, an incoming call that does not specify an extension cannot be transferred to an individual inside the office. In this scenario, the office is a private LAN, the main phone number is the public IP address, and the individual extensions are unique port numbers.[5]

Translation of the Endpoint

With NAT, all communication sent to external hosts actually contain the *external* IP address and port information of the NAT device instead of internal host IPs or port numbers.

- When a computer on the private (internal) network sends a packet to the external network, the NAT device replaces the internal IP address in the source field of the packet header (*sender's address*) with the external IP address of the NAT device. PAT may then assign the connection a port number from a pool of available ports, inserting this port number in the source port field (much like the *post office box number*), and forwards the packet to the external network. The NAT device then makes an entry in a translation table containing the internal IP address, original source port, and the translated source port. Subsequent packets from the same connection are translated to the same port number.
- The computer receiving a packet that has undergone NAT establishes a connection to the port and IP address specified in the altered packet, oblivious to the fact that the supplied address is being translated (analogous to using a *post office box number*).
- A packet coming from the external network is mapped to a corresponding internal IP address and port number from the translation table, replacing the external IP address and port number in the incoming packet header (similar to the translation from *post office box number* to *street address*). The packet is then forwarded over the inside network. Otherwise, if the destination port number of the incoming packet is not found in the translation table, the packet is dropped or rejected because the PAT device doesn't know where to send it.

NAT will only translate IP addresses and ports of its internal hosts, hiding the true endpoint of an internal host on a private network.

Visibility of Operation

NAT operation is typically transparent to both the internal and external hosts.

Typically the internal host is aware of the true IP address and TCP or UDP port of the external host. Typically the NAT device may function as the default gateway for the internal host. However the external host is only aware of the public IP address for the NAT device and the particular port being used to communicate on behalf of a specific internal host.

NAT and TCP/UDP

"Pure NAT", operating on IP alone, may or may not correctly parse protocols that are totally concerned with IP information, such as ICMP, depending on whether the payload is interpreted by a host on the "inside" or "outside" of translation. As soon as the protocol stack is traversed, even with such basic protocols as TCP and UDP, the protocols will break unless NAT takes action beyond the network layer.

IP packets have a checksum in each packet header, which provides error detection only for the header. IP datagrams may become fragmented and it is necessary for a NAT to reassemble these fragments to allow correct recalculation of higher-level checksums and correct tracking of which packets belong to which connection.

The major transport layer protocols, TCP and UDP, have a checksum that covers all the data they carry, as well as the TCP/UDP header, plus a "pseudo-header" that contains the source and destination IP addresses of the packet carrying the TCP/UDP header. For an originating NAT to pass TCP or UDP successfully, it must recompute the TCP/UDP header checksum based on the translated IP addresses, not the original ones, and put that checksum into the TCP/UDP header of the first packet of the fragmented set of packets. The receiving NAT must recompute the IP checksum on every packet it passes to the destination host, and also recognize and recompute the TCP/UDP header using the retranslated addresses and pseudo-header. This is not a completely solved problem. One solution is for the receiving NAT to reassemble the entire segment and then recompute a checksum calculated across all packets.

The originating host may perform Maximum transmission unit (MTU) path discovery to determine the packet size that can be transmitted without fragmentation, and then set the *don't fragment* (DF) bit in the appropriate packet header field.

Destination network address translation (DNAT)

DNAT is a technique for transparently changing the destination IP address of an en-route packet and performing the inverse function for any replies. Any router situated between two endpoints can perform this transformation of the packet.

DNAT is commonly used to publish a service located in a private network on a publicly accessible IP address. This use of DNAT is also called port forwarding, or DMZ when used on an entire server, which becomes exposed to the WAN, becoming analogous to an undefended military demilitarised zone (DMZ).

SNAT

The meaning of the term *SNAT* varies by vendor. Many vendors have proprietary definitions for *SNAT*. A common expansion is *source NAT*, the counterpart of *destination NAT* (*DNAT*). Microsoft uses the acronym for *Secure NAT*, in regard to the ISA Server. For Cisco Systems, *SNAT* means *stateful NAT*. For Watchguard Systems, *SNAT* means *static NAT*.

Microsoft's Secure network address translation (SNAT) is part of Microsoft's Internet Security and Acceleration Server and is an extension to the NAT driver built into Microsoft Windows Server. It provides connection tracking and filtering for the additional network connections needed for the FTP, ICMP, H.323, and PPTP protocols as well as the ability to configure a transparent HTTP proxy.

Secure network address translation

In computer networking, the process of network address translation done in a secure way involves rewriting the source and/or destination addresses of IP packets as they pass through a router or firewall.

Dynamic network address translation

Dynamic NAT, just like static NAT, is not common in smaller networks but is found within larger corporations with complex networks. The way dynamic NAT differs from static NAT is that where static NAT provides a one-to-one internal to public static IP address mapping, dynamic NAT doesn't make the mapping to the public IP address static and usually uses a group of available public IP addresses.

Applications affected by NAT

Some Application Layer protocols (such as FTP and SIP) send explicit network addresses within their application data. FTP in active mode, for example, uses separate connections for control traffic (commands) and for data traffic (file contents). When requesting a file transfer, the host making the request identifies the corresponding data connection by its network layer and transport layer addresses. If the host making the request lies behind a simple NAT firewall, the translation of the IP address and/or TCP port number makes the information received by the server invalid. The Session Initiation Protocol (SIP) controls many Voice over IP (VoIP) calls, and suffers the same problem. SIP and SDP may use multiple ports to set up a connection and transmit voice stream via RTP. IP addresses and port numbers are encoded in the payload data and must be known prior to the traversal of NATs. Without special techniques, such as STUN, NAT behavior is unpredictable and communications may fail.

Application layer gateway (ALG) software or hardware may correct these problems. An ALG software module running on a NAT firewall device updates any payload data made invalid by address translation. ALGs obviously need to understand the higher-layer protocol that they need to fix, and so each protocol with this problem requires a separate ALG. For example, on many Linux systems, there are kernel modules called *connection trackers* which serve to implement ALGs. However, ALG does not work if the control channel is encrypted (e.g. FTPS).

Another possible solution to this problem is to use NAT traversal techniques using protocols such as STUN or ICE, or proprietary approaches in a session border controller. NAT traversal is possible in both TCP- and UDP-based applications, but the UDP-based technique is simpler, more widely understood, and more compatible with legacy NATs. In either case, the high level protocol must be designed with NAT traversal in mind, and it does not work reliably across symmetric NATs or other poorly behaved legacy NATs.

Other possibilities are UPnP (Universal Plug and Play) or NAT-PMP (NAT Port Mapping Protocol), but these require the cooperation of the NAT device.

Most traditional client-server protocols (FTP being the main exception), however, do not send layer 3 contact information and therefore do not require any special treatment by NATs. In fact, avoiding NAT complications is

practically a requirement when designing new higher-layer protocols today (e.g. the use of SFTP instead of FTP).

NATs can also cause problems where IPsec encryption is applied and in cases where multiple devices such as SIP phones are located behind a NAT. Phones which encrypt their signaling with IPsec encapsulate the port information within an encrypted packet, meaning that NA(P)T devices cannot access and translate the port. In these cases the NA(P)T devices revert to simple NAT operation. This means that all traffic returning to the NAT will be mapped onto one client causing service to more than one client "behind" the NAT to fail. There are a couple of solutions to this problem: one is to use TLS, which operates at level 4 in the OSI Reference Model and therefore does not mask the port number; another is to encapsulate the IPsec within UDP - the latter being the solution chosen by TISPAN to achieve secure NAT traversal.

The DNS protocol vulnerability announced by Dan Kaminsky on July 8, 2008 is indirectly affected by NAT port mapping. To avoid DNS server cache poisoning, it is highly desirable to not translate UDP source port numbers of outgoing DNS requests from a DNS server which is behind a firewall which implements NAT. The recommended work-around for the DNS vulnerability is to make all caching DNS servers use randomized UDP source ports. If the NAT function de-randomizes the UDP source ports, the DNS server will be made vulnerable.

Advantages of PAT

In addition to the advantages provided by NAT:

- PAT (Port Address Translation) allows many internal hosts to share a single external IP address.
- Users who do not require support for inbound connections do not consume public IP addresses.

Drawbacks

The primary purpose of IP-masquerading NAT is that it has been a practical solution to the impending exhaustion of IPv4 address space. Even large networks can be connected to the Internet with as little as a single IP address. The more common arrangement is having machines that require end-to-end connectivity supplied with a routable IP address, while having machines that do not provide services to outside users behind NAT with only a few IP addresses used to enable Internet access, however, this brings some problems, outlined below.

Some[6] have also called this exact feature a major drawback, since it delays the need for the implementation of IPv6:

> "[...] it is possible that its [NAT's] widespread use will significantly delay the need to deploy IPv6. [...] It is probably safe to say that networks would be better off without NAT [...]"

Hosts behind NAT-enabled routers do not have end-to-end connectivity and cannot participate in some Internet protocols. Services that require the initiation of TCP connections from the outside network, or stateless protocols such as those using UDP, can be disrupted. Unless the NAT router makes a specific effort to support such protocols, incoming packets cannot reach their destination. Some protocols can accommodate one instance of NAT between participating hosts ("passive mode" FTP, for example), sometimes with the assistance of an application-level gateway (see below), but fail when both systems are separated from the Internet by NAT. Use of NAT also complicates tunneling protocols such as IPsec because NAT modifies values in the headers which interfere with the integrity checks done by IPsec and other tunneling protocols.

End-to-end connectivity has been a core principle of the Internet, supported for example by the Internet Architecture Board. Current Internet architectural documents observe that NAT is a violation of the End-to-End Principle, but that NAT does have a valid role in careful design.[7] There is considerably more concern with the use of IPv6 NAT, and many IPv6 architects believe IPv6 was intended to remove the need for NAT.[8]

Because of the short-lived nature of the stateful translation tables in NAT routers, devices on the internal network lose IP connectivity typically within a very short period of time unless they implement NAT keep-alive mechanisms by frequently accessing outside hosts. This dramatically shortens the power reserves on battery-operated hand-held

devices and has thwarted more widespread deployment of such IP-native Internet-enabled devices.

Some Internet service providers (ISPs), especially in India, Russia, parts of Asia and other "developing" regions provide their customers only with "local" IP addresses, due to a limited number of external IP addresses allocated to those entities. Thus, these customers must access services external to the ISP's network through NAT. As a result, the customers cannot achieve true end-to-end connectivity, in violation of the core principles of the Internet as laid out by the Internet Architecture Board.

- Scalability - An implementation that only tracks ports can be quickly depleted by internal applications that use multiple simultaneous connections (such as an HTTP request for a web page with many embedded objects). This problem can be mitigated by tracking the destination IP address in addition to the port (thus sharing a single local port with many remote hosts), at the expense of implementation complexity and CPU/memory resources of the translation device.
- Firewall complexity - Because the internal addresses are all disguised behind one publicly accessible address, it is impossible for external hosts to initiate a connection to a particular internal host without special configuration on the firewall to forward connections to a particular port. Applications such as VOIP, videoconferencing, and other peer-to-peer applications must use NAT traversal techniques to function.

Specifications

IEEE[9] Reverse Address and Port Translation (RAPT, or RAT) allows a host whose real IP address is changing from time to time to remain reachable as a server via a fixed home IP address. In principle, this should allow setting up servers on DHCP-run networks. While not a perfect mobility solution, RAPT together with upcoming protocols like DHCP-DDNS, it may end up becoming another useful tool in the network admin's arsenal.

IETF[10] *RAPT* (IP Reachability Using Twice Network Address and Port Translation) The RAT device maps an IP datagram to its associated CN and 0MN by using three additional fields: the IP protocol type number and the transport layer source and destination connection identifiers (e.g. TCP port number or ICMP echo request/reply ID field).

Cisco *RAPT* implementation is PAT (Port Address Translation) or NAT overloading, and maps multiple private IP addresses to a single public IP address. Multiple addresses can be mapped to a single address because each private address is tracked by a port number. PAT uses unique source port numbers on the inside global IP address to distinguish between translations. The port number is encoded in 16 bits. The total number of internal addresses that can be translated to one external address could theoretically be as high as 65,536 per IP address. Realistically, the number of ports that can be assigned a single IP address is around 4000. PAT will attempt to preserve the original source port. If this source port is already used, PAT will assign the first available port number starting from the beginning of the appropriate port group 0-511, 512-1023, or 1024-65535. When there are no more ports available and there is more than one external IP address configured, PAT moves to the next IP address to try to allocate the original source port again. This process continues until it runs out of available ports and external IP addresses.

Mapping of Address and Port is a Cisco proposal which combines A+P port address translation with tunneling of the IPv4 packets over an ISP provider's internal IPv6 network. In effect, it is an (almost) stateless alternative to Carrier Grade NAT and DS Lite that pushes the IPv4 IP address/port translation function (and therefore the maintenance of NAT state) entirely into the existing customer premises equipment NAT implementation. thus avoiding the NAT444 and statefulness problems of Carrier Grade NAT, and also provides a transition mechanism for the deployment of native IPv6 at the same time with very little added complexity.

3COM **U.S. Patent 6,055,236** [11] (Method and system for locating network services with distributed network address translation) Methods and system for locating network services with distributed network address translation. Digital certificates are created that allow an external network device on an external network, such as the Internet, to request a service from an internal network device on an internal distributed network address translation network, such as a stub local area network. The digital certificates include information obtained with a Port Allocation

Protocol used for distributed network address translation. The digital certificates are published on the internal network so they are accessible to external network devices. An external network device retrieves a digital certificate, extracts appropriate information, and sends a service request packet to an internal network device on an internal distributed network address translation network. The external network device is able to locate and request a service from an internal network device. An external network device can also request a security service, such as an Internet Protocol security ("IPsec") service from an internal network device. The external network device and the internal network device can establish a security service (e.g., Internet Key Exchange protocol service). The internal network device and external network device can then establish a Security Association using Security Parameter Indexes ("SPI") obtained using a distributed network address translation protocol. External network devices can request services, and security services on internal network devices on an internal distribute network address translation network that were previously unknown and unavailable to the external network devices.

Examples of NAT software

- Internet Connection Sharing (ICS): Windows NAT+DHCP since W98SE
- WinGate: like ICS plus lots of control
- iptables: the Linux packet filter and NAT (interface for NetFilter)
- IPFilter: Solaris, NetBSD, FreeBSD, xMach.
- PF (firewall): The OpenBSD Packet Filter.
- Netfilter Linux packet filter framework

References

[1] www.tcpipguide.com/free/t_IPNetworkAddressTranslationNATProtocol.htm (http://www.tcpipguide.com/free/t_IPNetworkAddressTranslationNATProtocol.htm)

[2] François Audet; and Cullen Jennings (January 2007) (text). *RFC 4787 Network Address Translation (NAT) Behavioral Requirements for Unicast UDP* (http://www.ietf.org/rfc/rfc4787.txt). IETF. . Retrieved 2007-08-29.

[3] "Characterization and Measurement of TCP Traversal through NATs and Firewalls" (http://nutss.gforge.cis.cornell.edu/pub/imc05-tcpnat/). December 2006. .

[4] "Illuminating the shadows: Opportunistic network and web measurement" (http://illuminati.coralcdn.org/stats/). December 2006. .

[5] "The Audio over IP Instant Expert Guide" (http://www.tieline.com/Downloads/Audio-over-IP-Instant-Expert-Guide-v1.pdf). Tieline. January 2010. . Retrieved 2011-08-19.

[6] Larry L. Peterson; and Bruce S. Davie; *Computer Networks: A Systems Approach*, Morgan Kaufmann, 2003, pp. 328-330, ISBN 1-55860-832-X

[7] R. Bush; and D. Meyer; RFC 3439, *Some Internet Architectural Guidelines and Philosophy* (http://www.ietf.org/rfc/rfc3439.txt), December 2002

[8] G. Van de Velde *et al.*; RFC 4864, *Local Network Protection for IPv6* (http://tools.ietf.org/rfc/rfc4864.txt), May 2007

[9] http://ieeexplore.ieee.org/iel4/6056/16183/00749275.pdf

[10] http://www3.ietf.org/proceedings/99nov/I-D/draft-ietf-nat-rnat-00.txt

[11] http://www.google.com/patents?vid=6055236

External links

- NAT-Traversal Test and results (http://nattest.net.in.tum.de)
- Characterization of different TCP NATs (http://nutss.net/pub/imc05-tcpnat/) – Paper discussing the different types of NAT
- Anatomy: A Look Inside Network Address Translators – Volume 7, Issue 3, September 2004 (http://www.cisco.com/en/US/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html)
- Jeff Tyson, HowStuffWorks: *How Network Address Translation Works* (http://computer.howstuffworks.com/nat.htm/printable)
- NAT traversal techniques in multimedia Networks (http://www.newport-networks.com/whitepapers/nat-traversal1.html) – White Paper from Newport Networks

- NAT traversal for IP Communications (http://www.voiptraversal.com/EyeballAnyfirewallWhitePaper.pdf) – White Paper from Eyeball Networks
- Peer-to-Peer Communication Across Network Address Translators (http://www.brynosaurus.com/pub/net/p2pnat/) (PDF) (http://www.brynosaurus.com/pub/net/p2pnat.pdf) – NAT traversal techniques for UDP and TCP
- http://www.zdnetasia.com/insight/network/0,39044847,39050002,00.htm
- RFCs
  - RFC 1631 (Status: Obsolete) - The IP Network Address Translator (NAT)
  - RFC 1918 - Address Allocation for Private Internets
  - RFC 3022 (Status: Informational) – Traditional IP Network Address Translator (Traditional NAT)
  - RFC 4008 (Status: Standards Track) – Definitions of Managed Objects for Network Address Translators (NAT)
  - RFC 5128 (Status: Informational) - State of Peer-to-Peer (P2P) Communications across Network Address Translators (NATs)
  - RFC 4966 (Status: Informational) - Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status
- *Speak Freely* End of Life Announcement (http://www.fourmilab.ch/speakfree/unix/) – John Walker's discussion of why he stopped developing a famous program for free Internet communication, part of which is directly related to NAT
- natd (http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-natd.html)
- SNAT, DNAT and OCS2007R2 (http://www.cainetworks.com/support/training/snat-dnat-ocs.html) – discussing the SNAT in Microsoft OCS 2007R2
- Alternative Taxonomy (Part of the documentation for the IBM iSeries)
  - Static NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwstatic.htm)
  - Dynamic NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwdynamic.htm)
  - Masquerade NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwaddmasq.htm)
- Network Address Translation - NAT (http://blog.ipexpert.com/2009/09/07/network-address-translation-nat/)
- Cisco Systems
  - Document ID 6450: How NAT Works (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml)
  - Document ID 26704: Network Address Translation (NAT) FAQ (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_q_and_a_item09186a00800e523b.shtml)
  - White Paper: Cisco IOS Network Address Translation Overview (http://www.cisco.com/en/US/technologies/tk648/tk361/tk438/technologies_white_paper09186a0080091cb9.html)
  - Cisco IOS NAT Commands Cisco IOS commands (http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/cs/csprtd/csprtd11/csnat.htm)
  - Animation Cisco NAT sample (http://www.cisco.com/image/gif/paws/6450/nat.swf)

Ebooks Dedicated to Richard Beckett

Article Sources and Contributors

Address Resolution Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=510461986 *Contributors*: 0x55534C, Abdull, Aldie, Alfio, Alvin-cs, Andareed, Arkrishna, Badams5115, Bcollins92131, Bernfarr, BillySharps, Bmaisonnier, Borgx, CarpeCerevisi, Casey Abell, Cburnett, CecilWard, Charivari, Chealer, Cv-natalay, Denisarona, Digimer, Dizon Ingens, Dkhydema, Dlohcierekim's sock, Dmccarty, DonDiego, Dorins, Doug.otis, Dysprosia, Edward, Edward Z. Yang, Eeekster, Enjoi4586, Enoent, Etienne.navarro, Euchiasmus, Favonian, Fireman biff, Fred Gandt, Fredrik, Gamera2, Gerixau, Giftlite, Glenn, Grawity, Guenthert, Guy Harris, Hkulkarni, Hulkster121382, Huwr, IanHarvey, Ibc111, Isnow, JTN, Jeltz, JesseW, Jim-j, Johncc330, Johnuniq, Jqgarcia, Karada, Karih, Karl-Henner, Kbrose, Kelly Martin, Kenyon, Kjj31337, Kku, Kondareddy, Kranthiswaroop, Ksn, Kvng, Leftturnsolutions, Lilac Soul, Lollerskates, Looxix, M. B., Jr., M4gnum0n, MER-C, Mandarax, Mange01, MeekMark, Mesoderm, MessiFCB, Mindmatrix, Misterfinsmiles, Ml198, Mokhov, Mouse Nightshirt, Mr Stephen, Muhandes, Music Sorter, Nageh, Nate Silva, Natersoz, Nealcardwell, Nixdorf, Nubiatech, Nuno Tavares, Olivier Debre, Ondenc, OsamaBinLogin, Papadopa, Pit, PolarYukon, QmunkE, Quarl, RB972, Rabarberski, Radiant chains, Rafostry, Raptor66, Rich Farmbrough, Richi, Rkinder, Rlevse, RobertKennedy, Roberta F., Rs junior, S4m1, SQL, SensuiShinobu1234, Sfisher, Sho Uemura, Spikey it, Stephan Leeds, Stormie, Suisui, Suruena, Swlee, Synergy, TYelliot, Tedernst, Template namespace initialisation script, The Anome, Tizio, Ttwaring, Uruiamme, Varlaam, Voidxor, Wbenton, Wesambhaya, Wk muriithi, Woohookitty, Yurik, Yyy, Zundark, 329 anonymous edits

Dynamic Host Configuration Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=513824819 *Contributors*: 78.26, 909078L, AStrunk, Abhayakara, Abhi3385, Abisys, Abune, Acroterion, AdamJudd, Adoniscik, Afiler, Aitias, Alaniaris, Alansohn, Aldie, Alex rosenberg35, Alfio, Alison22, AlistairMcMillan, Allstarecho, Almightylinuxgod, Alphachimp, Alvin-cs, Amaccuish, Ampsarus, Andareed, Andre Engels, Andrei Stroe, Animum, Anuragkrgupta, Apokrif, Arichnad, Arkrishna, Ashakiran, Ashwin, Atomician, BL, BW, Bardo, BartRoos, BenAveling, Billdorr, Bobby D. DS., Bogdangiusca, Bookbrad, BrianOfRugby, Brucefulton, CALR, Can't sleep, clown will eat me, Celarnor, Celinama, Centrx, CharlieEchoTango, Chealer, Chris 73, Chris the speller, Chrumps, ClanCC, Cleared as filed, Cliffb, Closedmouth, Coastalsteve984, Conversion script, Coutcin, Ctmt, DARTH SIDIOUS 2, DKEdwards, DStoykov, DVdm, Danger, Daniel.Cardenas, Davee, Davidkazuhiro, Davidoff, Davidstrauss, Dbu, DeadEyeArrow, Dee Jay Randall, Denelson83, Desiremore, Dgies, Dgw, DigitalSorceress, Dols, Dreadstar, Dremora, Dylan Lake, EagleOne, Earendur, Echosmoke, Edward, Egjose, Ehn, Electrified mocha chinchilla, Elwell, Enjoi4586, Enrique r25, ErikWarmelink, Evil Monkey, Ewlyahoocom, Feureau, Flash.killer, Fleung, Friday, FrostytheSnownoob, Fudoreaper, Furrykef, G Sisson, Gaius Cornelius, Gamera2, Gareth Owen, Gary King, Giftlite, Gimboid13, Grafen, Grenavitar, Guru cool, Gwalker nz, Hackingyourdna, Hadal, Hanjifi, Hannes Hirzel, HarisM, Harshbhai, Hazphi, Hazzamon, Hcberkowitz, Heathbar5477, Helix84, Hephaestos, Hide1713, Hughcharlesparker, Hughey, Infofarmer, Ironywrit, IsUsername, JTN, JaGa, Jasoltape, JasonWoof, Jasper Deng, Jimsve, Jobin RV, Joeinwap, Joerg Reiher, Josh Parris, Joy, Jrvz, Jude Rosario, Karora, Kartik Agaram, Kawanzaleroy, Kbrose, Kentyman, Kevinmon, Kgfleischmann, Kinema, King Toadsworth, Kingpin13, Kinu, Krellis, Ksn, Kubigula, Kurykh, Kvng, LAAFan, Labongo, Lacen, Lahiru k, Lightmouse, Lkinkade, Looxix, Loren.wilton, Lowellian, M4gnum0n, M7, MER-C, Mahewa, Makelelecba, Mam711, Mandarax, Mani1, Materialscientist, MattGiuca, Matthuxtable, Maximbo, Mcgonrya, Mechanical digger, Memming, Michael Devore, Mike Dillon, Mike Rosoft, Mindmatrix, Ministry of Truth, Mking30, Moeron, Moondyne, Mr Stephen, MrExplosive, Msabramo, Mwtoews, NCdave, Nachmore, Nafmosaved, Nastajus, Naved 12, Netsnipe, Ngriffeth, Nichtich, Nikai, Nimiew, Nixdorf, Njroberts, Nnp, Nubiatech, Nutster, Nv8200p, Ocker3, Ocram, Ojw, Omniplex, One, Ootachi, PCHS-NJROTC, PRBryson, PV=nRT, Pablicosta, Pagingmrherman, Pandemic, Parag2010, Parbatyogesh, Paris.butterfield, PeaceNT, Pedant17, Peng, Phatom87, Philcha, Piet Delport, Pinkadelica, Pion, Pokemonmegaman, Prohlep, Pseudomonas, QRX, Quarl, RJaguar3, Raanoo, Radiant chains, Raghav, RattleMan, Razimantv, Rchandra, RedWolf, Reub2000, Rfc1394, Rich Farmbrough, Richwales, Rikboven, Rjstott, Rjwilmsi, Rl201, Rob Hooft, Rodrigo.haus, Roris.solaris, Rrburke, Rursus, SJP, Sam Korn, Sameer.sujeet, Savantas83, Schaea, Scott.somohano, ScottSteiner, Sdrtirs, Sesshomaru, Shady69, Shields020, Shiftoften66, Shiryaev, Skyfiler, Slazenger, Snarius, SpK, Spasemunki, SpeedyGonsales, SportWagon, Srbanator, Srharriott, Stephan Leeds, Stevenm.ict, Suburbanslice, Suffusion of Yellow, Sunray, Superborsuk, Superm401, Suruena, Swatithorve, TYelliot, Taochen, Taoqirkhosa, Tcb, Teemuk, Tellyaddict, Template namespace initialisation script, Teraya, The Thing That Should Not Be, The.valiant.paladin, Theking17825, Thomas d stewart, Thumperward, Tide rolls, Tim.sylvester, Tothwolf, Totty3478, Tremilux, True Pagan Warrior, Trusilver, Tyler.szabo, Ummit, UncleBubba, Utcursch, VernoWhitney, Vijeshchandran, Vocaro, Waggers, Wasisnt, Wdrazo, Wereon, Wiki104, Wikisierracharlie, Wilson.canadian, Winterst, Wisco, Wisher, Wizzard2k, Wk muriithi, Wrs1864, XLiquidIceX, Xhienne, Xionbox, Xoxon1kk1, Yesteraeon, YordanGeorgiev, Yuckfoo, Zac439, Zedla, ^demon, 922 anonymous edits

Domain Name System *Source*: http://en.wikipedia.org/w/index.php?oldid=516801726 *Contributors*: (, 194.17.59.xxx, 28421u2232nfenfcenc, 2D, 4th-otaku, 777sms, A-moll9, ANONYMOUS COWARD0xC0DE, Abhayakara, Abhinav paulite, Adamantios, Adambiswanger1, Ahoerstemeier, Ahu, Aldie, Ale2006, AlexJP, Alfio, AlistairMcMillan, Allstarecho, Alphachimp, Alphax, Although, Alvestrand, AnK, Anastrophe, Andareed, Andrejj, Andrew Hampe, AndrewN, Andrewpmk, Anjaliiiiiiiiii, Anna Lincoln, Anon lynx, Anthony Appleyard, Anwar saadat, Apacheguru, Arcade, Ardenn, Arichnad, Arifin banjar, Arjayay, Art LaPella, Arvindn, Avoided, Axonizer, Ayla, Azaziel, Azraell, Bahaanet, Balajisarathi, Banej, Barek, Bart133, Bayerischermann, Beachy, Beard0, Bedekar.nilesh, Ben-Zin, BenM, Beno1000, Bevo, Bhadani, Bhairab.online, BigMoneyJim, Bigflavor, Binaryspiral, Blanchardb, Bloggeruk, Blue Dot, Bluezy, Boba5fett, Bobblewik, Bobo192, Bobrayner, Bocafan76, Borislav, Bovineone, Brandon, Breno, BrianTung, Brianga, Brichcja, Brucevdk, BryDye, Bryan Derksen, Btharper1221, Byronknoll, Cain Mosni, Calvin 1998, Calvinkrishy, Can't sleep, clown will eat me, CapitalR, Capricorn42, CardinalDan, Cate, Cbuckley, Centrx, Ceyockey, Chairman S., Chasindream, Chealer, Checco, Chenzw, Chipp, Chocolateboy, Chris fluppy, ChrisCork, Chrisdab, Christian List, Cjkporter, ClaretAsh, Cleared as filed, CobraA1, Codes02, Cohesion, Colfer2, Cometstyles, Conversion script, Coredesat, Corti, Cremepuff222, Cst17, Cyanoa Cylate, Cybjit, Cynric, DGMDGM, DNSstuff, Damian Yerrick, Dandorid, David Haslam, David.Monniaux, DavidCary, Davidshq, Davipo, Dbo789, Dcsutherland, Dee.tee, Delivery:435, Demitsu, Dennis Brown, DerHexer, Derat, Diannaa, Discospinster, Dithridge, Djkrajnik, Dman727, Dolda2000, DoomBringer, Doomdayx, Dougalf, Dr.queso, Draicone, Dreg mogul, Dripdk, Drj, Drjan, Drowne, Dtobias, Dwheeler, Dysprosia, EdJohnston, Editor plus, Edward K, Edward321, Egmontaz, Ehn, Elinruby, Eloquence, Elvey, Emdx, Epastore, Epbr123, Ephillips, Equendil, Eras-mus, Eric Kvaalen, ErikWarmelink, Evil Monkey, Explicit, Feezo, Fenitharbour, Feydey, Fifthnail, FisherQueen, FleetCommand, Fleminra, Flubbit, Forseti, Fortunecookie289, Frap, Fred Gandt, Fudoreaper, Funkysh, Funnyfarmofdoom, FvdP, Fæ, GFellows, Garas, Gary King, Gdr, Geoffspear, Gerbrant, Gerixau, Gilesmorant, Glane23, Glenn, Glrx, Gobbleswoggler, Gogo Dodo, GoingBatty, Graciella, GraemeL, Graham87, Grayshi, Greg.smith.tor.ca, Groogle, Guanxi, Gurch, Gutzalpus, Haakon, Hadal, HaeB, Hairy Dude, HalJor, Hamtechperson, Hardaker, Harris7, Hayabusa future, Hbatra, Hcberkowitz, Hdt83, Helix84, Helixblue, Henriwatson, Hilaryam89, Honta, Horsten, Htaccess, Hu12, IJMacD, Ibuzzo, Idont Havaname, Insanity Incarnate, Inter, InverseHypercube, Iohannes Animosus, Ispabierto, Iulianu, J.delanoy, JForget, JH-man, JSGould, JTN, Jackbrownwan, Jake Wartenberg, Jamyskis, Jarsyl, Jasynnash2, Jauerback, Javawizard, Jawsper, Jcw69, Jdthood, Jec, Jesant13, Jfpierce, Jgeer, Jh51681, Jhasai, Jhawkinson, Jiddisch, Jinlye, Jivecat, Jj137, Jlaidman, Jmcw37, Jnc, JoanneB, Joe Sewell, Jog1973, Johan Elisson, John Broughton, John Vandenberg, JohnSmith777, Johnarett, Jonah S., Joonas Govenius, Joseph Dwayne, Joseph Solis in Australia, Jpatokal, Jscbiss, JuJube, Jy, Jyoshimi, Jyoujjawal, KGasso, KLLvr283, Kaconk666, Karada, Karengpve, Kateshortforbob, Katieh5584, Kbrose, Kelly Martin, Kenguest, Kinema, Kingpin13, Kinneyboy90, Kitplane01, Kizor, Kjdunlap, Kjetil r, KnightRider, Kop, Kraftlos, Krawi, Krellis, Krooga, Kungfuadam, Kurkku, Kurumban, Kvng, Kwamikagami, Kwertii, Labongo, Lambtron, Leandrod, Lee Daniel Crocker, Lemming, Lentoyip, LeoNomis, Liamdaly87, Liftarn, Lightdarkness, Lightmouse, Limajean34983, LimoWreck, LionKimbro, Lod, Lotje, Ludde23, Luna Santin, Lupin, Lysdexia, MC MasterChef, MER-C, MWD27, Maestro25, Mahanga, Mam711, Mange01, Marc Mongenet, Marcel van b, Marcod'Itri, Marechal Ney, Mark Arsten, Mark Bergsma, Mark7-2, Materialscientist, Matilda, Matt Darby, Matt.smart, MattGiuca, Matthewvelie, Mattias.Campe, Mattrobinson78, Maury Markowitz, Max Cheung, Maximillion Pegasus, Mblumber, Mendel, Merlion444, Merovingian, MetsFan76, Mfwitten, Mgechert, Michael Hardy, Michel Jansen, Mike Rosoft, Mikeborschow, Mindmatrix, Minesweeper, Minghong, MonRa, Monty845, Moondyne, Moonriddengirl, Moskvax, MrJones, MrKris, Mrguyguy226, Mrzaius, Msantcroos, Msjadex, Mwtoews, Mysekurity, N0YKG, NTox, Nabber00, Nakon, Name99, Nanshu, Naraht, Narayanacharan, Natkeeran, NawlinWiki, Nburden, Nealcardwell, Neil916, Netesq, Netkinetic, NevilleDNZ, Nevlino, NewEnglandYankee, Niceguyedc, NickCT, NickW557, Nickmaine, Nilesh2293, Nimiew, Nishanth.fairfield, Niteowlneils, Nivix, Nneonneo, NotAnonymous0, Notheruser, Nubiatech, O, Oddity-, Olddocks, Oldwes, Oliver341, Omegagod, Omicronpersei8, Omiksemaj, Omniplex, Openhazel, Optimist on the run, Oshah, Oskar Sigvardsson, Ott2, Ottawahitech, Oxymoron83, Ozruta, Pagingmrherman, Pak21, Parsecboy, Patrick, Paul Stansifer, Paulson74, Pcb21, Pedant17, Peter Karlsen, Peterbrown1, Pgan002, Pharaoh of the Wizards, Phatom87, Philarete, Philip Trueman, PhilipMW, Piano non troppo, Pierre Monteux, Pilom, Pinethicket, Piotras, Pjrich, Pointer1, Poweroid, Pranav.ota, Puffin, Qasimalikhawaja, QmunkE, Quarl, Quincywalltech, Quintote, Qwertyca, Qwertyus, R4f, RAM, RachaelJetkins, Ragib, RainbowOfLight, Rameshbabu.itian, RandomStringOfCharacters, Rapomon, Rasbelin, Rasmus Faber, Ravedave, Rawlogic, Raybellis, Reach Out to the Truth, RedHillian, Reeser8, Regibox, ReiDx, RetiredUser2, Rfc1394, Riana, Rich Farmbrough, Rick Block, Rick Sidwell, Rikkus, Rjd0060, Rjgodoy, Rob Hooft, Robertb-dc, Roger Davies, Ron Ritzman, Ronz, Rottenblackberry, Roughana, RoyBoy, Rrius, Rror, Rshin, RussBlau, Ruwanindika, Rwashi, S3000, SDC, SJP, SMC89, ST47, SWAdair, Salokha, Samboy, Sanbio, Sanjiv swarup, Scarian, Sceptre, Schmloof, Schnolle, Science4sail, Scolobb, Scott.somohano, Scratchy, Seaphoto, Sen86, Serezniy, Serie, Shadowhillway, Shadowjams, Shalom Yechiel, Shams9598, Shenme, Shentino, Shii, Simigh, Simon Dodd, Simonjonzie, Sionus, SirPavlova, Skarebo, Sleigh, Smalljim, Smallpond, Smappy, Smashville, Smichae, Snowolf, Snoyes, Socialservice, Sokool, SolarisBigot, Solveforce, Some jerk on the Internet, Sonomadiver@gmail.com, Sonuahluwalia, SpaceFlight89, Sparkie82, Spellmaster, SperoSpes, Spinboy, Spishco, Sstrader, StaticGull, StealthFox, Stephan Leeds, Stephen Gilbert, Stib, Stwalkerster, SuperHamster, Suruena, SusanLarson, Sweet473, SyntaxError55, Syp, TAG.Odessa, Takis, Tardis, Taw, TedPavlic, Tedder, Teppic, The Anome, The King's Peach, The Thing That Should Not Be, The locster, Thefunkygibbon, Thumperward, Tide rolls, Tim1988, Timwi, Tirerim, Tobias Bergemann, Tobias Conradi, Tonsofpcs, Tony1, Torzsmokus, Totty3478, Towel401, Trantjd, Tree Biting Conspiracy, Tregoweth, Trusilver, Ttony21, Tualha, Turnstep, Twajn, Twinxor, Tylerdmace, Tyomitch, Ufirst, Unara, Uncle G, UncleBubba, Updatebjarni, UrsaFoot, Use Bombs, Valenciano, Vanished user 39948282, Vanisheduser12345, Vary, Vera Cruz, Versageek, VictorianMutant, Vivio Testarossa, Vldhcp, Vocaro, Vrenator, Vssun, W2bh, WRJ, Wasisnt, Wavelength, Wayward, Webmaster4india, Webzero, Weike9009, Wesley, Weyes, Wigren, Wiki alf, Wiki104, WikiReviewer.de, Wikibase, Wikibob, Wikibofh, Wikiborg, Wikijeff, Wikipedia.org@parkerpress.com, WikipediaEdit, Wikipelli, Will Beback Auto, Wizardist, Wk muriithi, Wknight94, Wmahan, Wrs1864, Xardion, XaroPoo, Xionbox, Yamamoto Ichiro, Yes-Chef?, Yogi de, Yurik, Yyy, ZSBoS, ZaffaNE, Zelphar, Zfr, Zidane2k1, Zro, Zundark, Zzuuzz, Пика Пика, 1779 anonymous edits

IPv4 *Source*: http://en.wikipedia.org/w/index.php?oldid=516844395 *Contributors*: -Majestic-, A.R., Abdull, Acather96, Adrian.benko, AlephGamma, Alexkon, AlistairMcMillan, Althena, AndreasWittenstein, Andrewmc123, Andypar, Angela, ArglebargleIV, Arjayay, Armando, Axelriv, Barro, Barryd815, Begoon, Bmpercy, Borgx, Breno, Brest, Bro1960, Brouhaha, C. A. Russell, CCFreak2K, CWii, Calcwatch, Calinou1, CaribDigita, Carlo.arenas, Cburnett, CecilWard, CesarB, Chris D Heath, Chrishmt0423, Christan80, Cm115, Cmichael, Conversion script, Coredesat, Corti, Crashdoom, Cwolfsheep, Cybjit, Cynical, DARTH SIDIOUS 2, DBigXray, DH85868993, Dan6hell66, Daniel Staal, Daniel.Cardenas, Danielbarnabas, DataWraith, DavidDelaune, DenisKrivosheev, Dmaftei, Dnas, Dotshuai, DreamGuy, Dsearls, Duffman, Dunganb, Dungkal, Ed g2s, EdC, Ekspiulo, El C, Electron9, Enjoi4586, Ericbarch, ErikWarmelink, EvilSS, Exallium, Faco, Favonian, Floydpink, Formulax, Fred Bradstadt, Fredrik, Fresheneesz, Gallando, Garo, Gehlers, General Wesc, Giftlite, Glrx, Graciella, Graham87, Graven69, Gthm159, Hairy Dude, Hcberkowitz, ILike2BeAnonymous, IRedRat, Ilario, Imroy, Indrek, Ironholds, JTN, Jasper Deng, Jbergste, Jbohac, JeroenMassar, Jesant13, Jgeer, Jk2q3jrklse, Jnc, Joelby, Johnnyboyshoots,

Johnteslade, Joseph Solis in Australia, Joshua, Jpyeron, Jwdonal, Kaal, Kaeso, Karada, Kasperl, Katieh5584, Kbrose, Kevin66, Kgfleischmann, Khr0n0s, Khvalamde, Kickboy, Kiore, Kmwiki, Krellis, Kvng, Kwamikagami, Kyng, Leuqarte, Liface, Lightmouse, Lph, Lukeritchie, M.O.X, MECU, Magioladitis, Magnus.de, Maimai009, Mange01, Marcoscm, Markrod, Melnakeeb, Mewashere1, MiNeEstasVortaristo, MichaelGoldshteyn, Milan Keršláger, Mindmatrix, Miss Saff, Mmtrebuchet, Mochi, Mokgen, Molerat, Morten, Mushroom, Nathan Hamblen, Nealmcb, Neelix, NewEnglandYankee, Nil Einne, Noldoaran, Nubiatech, Ojw, Omniplex, Onceler, Opelio, Outback the koala, Parent5446, Parkamark, Paul, PaulHanson, Pengo, Peterhoneyman, Phantomdj, Philip Trueman, Phoenix314, Phorque, Piano non troppo, Pmj, Pratikarun, Presto8, Ptmc2112, Raanoo, Ranto, Rantsroamer, Rblhjm, Rchandra, RevRagnarok, RexNL, Rjwilmsi, Robert Brockway, RoySmith, Rpwoodbu, Ruwolf, RxS, RyanWKeen, Sarafankit, Scjessey, Seaphoto, Shane kerr, Simon J Kissane, Sirmelle, SmilingBoy, Smurrayinchester, SpacemanSpiff, Spearhead, SpectatorRah, SpeedyGonsales, Stephan Leeds, StrangerInParadise, Suruena, Swellesley, Taestell, Teemuk, Tenretnieht, The Anome, The Thing That Should Not Be, TheGreyArea, Themonstergila, Thnidu, Tide rolls, Toolnut, Tyler.szabo, UU, Ultimus, Undeference, Versus22, Visiting1, Vivio Testarossa, Vkartiik, Wavelength, Winston Chuen-Shih Yang, Wolfsbane2k, Woohookitty, Wrs1864, Wyksztalcioch, XavierHager, Xibe, Yann Lejeune, Yyy, Zanetu, Zetawoof, Zfr, ^demon, 516 ,ירח אפל anonymous edits

IPv6 *Source*: http://en.wikipedia.org/w/index.php?oldid=516777238 *Contributors*: 09 F9 11 02 9D 74 E3 5B D8 41 56 C5 63 56 88 C0 - hack, 128.107.253.xxx, 1exec1, 2001:44B8:3178:E800:0:0:B000:B1E5, 2001:44B8:3178:E800:F5D0:5DED:E7AE:8192, 2001:470:890A:1:216:D4FF:FEEB:DF4B, 2001:db8, 2345us, 2A00:F480:4:134:8CD0:DE64:F766:A6EC, 2A00:F480:4:2A1:E02E:59C3:B36D:DB51, 49oxen, A3 nm, AJR, Abune, Accountholder, AceJohnny, Adam Conover, Adamthewebman, Adoniscik, Aeluwas, Aeons, Agent00111, Ahoerstemeier, Aigarius, Ajbool, Akamad, Alainkaa, Alansohn, Aldie, AlephGamma, Alex43223, Alexander UA, Alexkon, Alexwcovington, AlistairMcMillan, Allanlw, Allens, Amigan, Amybajwa, AnandKumria, Anastrophe, Andareed, Anders Feder, Andonic, Andrei Stroe, Andrew4u, AndyTheGrump, Annesville, AnonMoos, Antonis Christofides, Arbitrary username, Arichnad, ArnoldReinhold, Artemgy, Arthur Rubin, Aschwegmann, Ashley Y, Astronautics, Atakdoug, Auntof6, Avanu, Axelriv, Baa, Badcalculon, Badon, Balachanderk, Barri, Bazsa, Bbpen, Bdesham, Belamp, Beland, Benabik, Benandorsqueaks, Benbest, Bender235, Bentendo24, Beoba, Bertra g, Bgraabek, Big Brother 1984, BioTube, Blaynew, Blubber42, Bobblewik, Bobo192, Boothy m, Borgx, Bornapilot, Bousquf, Bovineone, Brandmeister, Breno, Brianmaddox, BrokenSegue, Brownsteve, Bruce404, Bsv109, Bugfood, BurnDownBabylon, C.S.Abishek, CKD, CKerr1, CableCat, Cal-linux, Calvin 1998, Cameltrader, Cananian, Canterel, Cap'n Refsmmat, Capek, Caper13, CapitalR, Captkirk, Carlosguitar, Centrx, CesarB, Cesarolvera, Cgdallen, ChaosR, Charliel 94, Charu, Charwinger21, Chaser, Chbarts, Chris 73, ChrisErbach, ChrisGualtieri, Chrisinspfld, Cinnamondouche, ClamDip, Closedmouth, Cloudmonkey, Cmdrjameson, Cmsb705, Coaxial, Cobi, Cometstyles, ComputinChuck, Confuciou, Conversion script, Corso84, Crazycomputers, Crd Alameda, Credema, Crispmuncher, Crl620, Crwth, Cst17, Cwolfsheep, Cyan, Cybercobra, Cybjit, Cyp, DARTH SIDIOUS 2, DBooth, DMahalko, DNSstuff, Daev, Dainomite, DalZot, Dale Arnett, Dan100, Dandorid, Daniel Luechtefeld, Daniel.Cardenas, DanielDeGraaf, Darguz Parsilvan, Darxus, Das7002, DataMatrix, Dave6, David Gerard, Davidhorman, Dawnseeker2000, Deepeedoyle, Defyant, DerekMorr, Dglynch, Diannaa, Dickguertin, Dicklyon, Dispenser, Djschaap, Dlabtot, Dlange13, Dnas, Dnevil, DocKrin, Dolda2000, Don4of4, Donreed, Dontopenyoureyes, DragonflySixtyseven, Drangon, Drearwig, Drewheasman, Droob, Dwandelt, Dwmalone, Dylan Lake, Dylan620, E94mli, EDUBLE, EEMIV, EdBever, EdC, EdoDodo, Edokter, Edward, Ehn, Ejumper, El C, Eldar, Emonk72, Endpoint, Enjoi4586, Enquire, Equazcion, Eradt11, Erc, ErikHaugen, EthanL, EugeneZelenko, Euphrosyne, Evil Monkey, Extropian314, FGont, FT2, Falcon8765, Falcon9x5, Farncombe, FatalError, Feedmecereal, Feureau, Fewaffles, Figz, Fildon, Firealwaysworks, FlieGerFaUstMe262, Fluffy 543, Fnagaton, Folajimi, Foobar, Fragglet, Frap, Fredrik, Freeaqingme, Fresheneesz, Frnkblk, Fsterry, Gaius Cornelius, Galmicmi, Geekening, GerardM, Ghane, Giftlite, Giraffedata, Glane23, Glen Hein, Glenn burdett, Glenndavies, Glrx, Gmaxwell, Gmedding, Gorffy, GovStuff, Graciella, Graham87, GrahamHardy, GreenReaper, Greg L, Gudeldar, Guiltyspark, Gunnala, Gurch, Gwernol, Gyrospsherus, HAl, Haakon, Hairy Dude, Haleyga, Ham Pastrami, Hankwang, Hannes.nz, HarryHenryGebel, Hashar, Hatu7, Hawaiian717, Hcberkowitz, Hdt83, Helix84, Hfastedge, Hires an editor, HobbesLeviathan, Holizz, Holmwood, HorsePunchKid, Hpa, Husky, Hvn0413, Hypersonic, II MusLiM HyBRiD II, IRedRat, Ihope127, Ilja Lorek, Iluvcapra, Imarsman, Imroy, Imzogelmo, Incnis Mrsi, Incompetence, Infofarmer, Insanity Incarnate, Int21h, InterMa, Intgr, Intrepion, Iromeister, IronGargoyle, Ironholds, Itojun, Ivolocy, J-D-Cronin, J128, JHunterJ, JTN, Jaizovic, Jamesday, JameySharp, Jan1nad, Jansenguus, Janto, Jarry1250, Jason Quinn, Jasper Deng, Jbossbarr, Jclemens, Jddahl, Jec, Jeff G., Jeff02, JeffMorriss, Jeffsw6, Jengelh, Jeremy Visser, JeroenMassar, Jfromcanada, Jhd, Jhwoodyatt, Jithrae, Jj137, Jnc, Jo3sampl, JodyB, JoeKearney, Joefromrandb, John Maynard Friedman, JohnOwens, Johnsmith4092, Johnuniq, JonDePlume, Jondel, Jonkerz, Jordandanford, Jordipalet, Joshua Scott, Jtg, Julesd, Justjohn45, K0rana, KDS4444, Kaldari, Karada, Kasperd, Kattmamma, Kaypoh, Kb966k, Kbolino, Kbrose, Kcordina, Keilana, KelleyCook, Kenyon, Kestasjk, Kgfleischmann, Kieff, Kinema, Klaver, Koavf, Konikofi, Kozuch, KrakatoaKatie, Krellis, Kroneage, Kthomas8, Kusma, Kvaks, Kvng, Kynan, LP-mn, LUUSAP, LVXN112, Lagesag, LakeHMM, Latifladid, Laug, Lawrence Cohen, Lightbulbcole, Likestheaction, Logan, Lord Chamberlain, the Renowned, Lotje, Lotu, Lousyd, Luigiacruz, M. B., Jr., M.O.X, M00dawg, Maalaoui, Macrakis, Madalibi, Madhav208, Madman91, Maduskis, Magioladitis, Mahtin, Maian, Majordragon, Manankanchu, Mange01, Mansoor.riz, Marcod'Itri, Marius p, Mark Bergsma, MarkMLl, Markus Kuhn, MarkusWanner, Martarius, Martyvis, Marudubshinki, Matchups, Mathboy965, MaxWilder, Maxim Masiutin, Mdd4696, Meand, Megan at ARIN, Meneth, Mentifisto, Mhd.ashraf, Michael B. Trausch, Michael miceli, Michael.davie, MichaelBillington, Michaelrayw2, Midnightcomm, MihalOrela, Mike Rosoft, Mike Schwartz, MikeWren, Mikeyman12345, Mikm, Mindmatrix, Minghong, Minimac, Mintleaf, Mitar, MithrasPriest, Mmxx, Mnudelman, Mobus, Molerat, Money23, Mordomo, Mormonsareloser, Mr Minchin, MrOllie, Mro, Msiebuhr, Muad, Mukkakukaku, Mulad, MureninC, N328KF, NYMets2000, Nacnud22032, Naff89, Nageh, NameIsRon, Napalm Llama, NapoliRoma, Narellec, Nasa-verve, Nashrul Hakiem, Nate Silva, Naveenpf, Nczempin, Nealmcb, NerdBoy1392, NerdyScienceDude, Netrangerrr, Nhandler, Nicd, Ninjagecko, Nixdorf, Nixeagle, No More TV, Noldoaran, Northgrove, Notbyworks, Nsaa, Nubiatech, Nurg, Nwbeeson, Nzseries1, OSborn, Od Mishehu, Ohnoitsjamie, Old Moonraker, Olipro, OlivierMehani, Oliwan, Omegatron, Omegium, Omicronpersei8, Omniplex, One half 3544, Oneliner, Orchistro, OriumX, Ost316, Ovideon, P.vishnu7, PabloStraub, Paine, Pankkake, Paolopal, Paradox2, Paranoid, Parsmutaf, Pascal666, PassportDude, Pathoschild, Patrickdavidson, Paul1337, PaulHanson, PaulTanenbaum, Peak, PedroPVZ, Pekaje, Pengo, Personman, PeterCScott, PeterKz, PeterStJohn, PhilHibbs, Philc 0780, Phoe6, Phoenix-forgotten, PierreAbbat, Pinkgothic, Pjrm, Planetscared, Plasticup, Plau, Pnm, Pointillist, Pontillo, Porttikivi, Powerlord, Prabhuhwiki, Prolog, Prunesqualer, Psheld, Psz, Ptmc2112, Public Menace, Puchiko, PuerExMachina, Quaestor23, Quantumobserver, Quebec99, Qwertytech, Qwertyus, RHaworth, Raffen, Raghith, Rait, Ralphcase, Random832, Rchandra, Rcloran, Rd232, Rdenis, RedWolf, Redalert2fan, Redlazer, Reimon88, Relativitydrive, Rememberway, ReyBrujo, Reza 2638, Rgaushell, Rich Farmbrough, Richard W.M. Jones, RichiH, Rick Sidwell, RickBeton, Rilesman, Rischmueller, Roadrunner7, Robaire5006, Robert Brockway, Robomaeyhem, Rocketman768, Ronz, Roy.wonder.cohen, RoyBoy, RoySmith, Rps, Rrius, Rwessel, Ryankearney, Rythie, Ryulong, Ryuzaki00, ST47, SWAdair, Sam Hocevar, Samantha of Cardyke, Sandeepsoman, Saran945, Sarutv, SaveTheRbtz, Scientus, Scj2315, Scott.somohano, Scottbadman, Scottywong, Scratchy, Seano1, Sebcastle, Sega381, SelfStudyBuddy, Setherson, Sfan00 IMG, Sgeeves, Shadowjams, ShakataGaNai, Shirishag75, Shultz IV, Sibi antony, Sick bug, Sigkill, SillyWilly, Simon J Kissane, Sirmelle, Sjmsteffann, Sjrosen, Skittleys, Skybon, Slahrdzhe, Smallpond, Smarter1, SmilingBoy, Smurfy, Smurrayinchester, Snaxe920, Snorgy, Sobec, Socialservice, SolarWind, Some jerk on the Internet, Sonicsuns, Sosodank, South Bay, Southen, Spearhead, Spicemines, SpuriousQ, Squids and Chips, SrMico, Stefan, Stephan Leeds, Stephenb, SteveDavidsen, StuartBrady, Stux, Stwalkerster, Subsurfer, Sukenjain, Suruena, Suseno, Svick, SymlynX, Synchrite, Sysy, TJJFV, TJRC, TYelliot, TankMiche, Team4Technologies, Teapeat, Teemu Maki, Teemuk, Template namespace initialisation script, Tenth Plague, Terillius, Terjepetersen, Tfl, ThG, The Anome, The Last Username I Could Think Of, The Nut, TheAnarcat, TheLight, Theant2000, Thehotelambush, Thewallowmaker, Thexchair, Thorpe, Thrindel, Thumperward, ThurnerRupert, Tim Capps, Titoxd, Tjh1234, Tmaufer, Tmh, To Serve Man, Tommy2010, Towel401, Tpbradbury, Trejrco, Trevor Johns, Tristanb, Trisweb, Trontonic, Trusilver, TwoHundredOk, Tylerl, Ukh, Uncle Milty, UncleDouggie, Unclevinny, Und1sk0, Unitacx, Universalcosmos, Uruiamme, Utility Knife, Vajrallan, Valenciano, Vannessa.beth, Vedantm, Vegaswikian, Verdy p, Veriodbg, Vertium, Vespristiano, Victor, Viniciustinti, Viperious, Visiting1, Voidxor, Voomoo, WJBscribe, Wahjava, Warren, WarthogDemon, Watain, Wavelength, WayneMokane, Weatherman1126, Weregerbil, Wetman, Weyes, Whatisipv6, Wiki13, Wikimoder, WikipedianYknOK, Wimt, Winston Chuen-Shih Yang, Wisco, Wk muriithi, Wmahan, Wonderstruck, Wrs1864, Ww, Wwoods, X-Fi6, Xandell, Xeltran, Xerces8, Xpclient, Yago, Yama, Yayay, Youssefsan, Yurik, Yurivict, Zachlipton, Zaphraud, Zbxgscqf, Zed5linux, Zeerak88, Zemyla, Zenmohit, Zhackwyatt, Zmding, ZorphDark, Обедающий философ, 1696 anonymous edits

Network address translation *Source*: http://en.wikipedia.org/w/index.php?oldid=517151024 *Contributors*: (, 2001:470:1F09:10D6:215:FF:FE77:FD85, 65.29.90.xxx, Aapo Laitinen, Aawc, Aelantha, Aitias, Ajo Mama, Alan U. Kennington, Alansohn, Aldie, Alex Smotrov, Alex.atkins, Alex.zeffertt, Alexhixon, AlistairMcMillan, Althena, Altrn8r, Andrew Hampe, Andrewpmk, Andrewriddell2, Aneah, Angela, Ap, ArséniureDeGallium, Ashwin, Asymmetric, Balajisarathi, Barek, Bbpen, Benoit rigaut, Bevo, Bos-Herz edit acct, Brion VIBBER, Brynosaurus, Cate, Cbarbry, Cburnett, CesarB, Cf. Hay, Cheung1303, Chowbok, Christian75, CommonsDelinker, Conversion script, Copsewood, Cotoco, Cpartsenidis, Crazycomputers, Crispmuncher, CrucifiedChrist, CyberSkull, Cybjit, D235, DARTH SIDIOUS 2, Daf, Damienivan, DanielEng, Daveg1k, Daveofthenewcity, Dawnseeker2000, Dcoetzee, DevastatorIIC, Dgtsyb, DiGiT, DisillusionedBitterAndKnackered, Droob, Drpixie, Drumzandspace2000, Dspradau, Dysprosia, EH74DK, Edcolins, Eddy264, Edward, Eolsson, Equendil, Ergy, Everyking, Evil Monkey, Excirial, Felipe1982, Fenix*NBK*, Fresheneesz, Gandaliter, Gareth Owen, Gary King, Garyvdm, Giftlite, Giraffedata, Glenn, Goatasaur, Golbez, GorillaWarfare, Gracefool, Graham87, Grimmfarmer, Guiltyspark, Guitargod2323, Hairy Dude, HarlandQPitt, Harrymcfogs, Hcberkowitz, Helix84, Hovden, Hydrargyrum, Icairns, Imcdnzl, Indrian, Iranway, Ivan Pozdeev, Ivan.Lt, J.delanoy, JHunterJ, JTN, Jan Kunder, Jasper Deng, Jengelh, Jez9999, Jhbdel, JidGom, Johnuniq, Jokerspuppet, JonDePlume, JonHarder, Jondel, Jonshea, Josh Parris, Joshf, Joy, Jpbowen, Jsnx, Just Another Dan, Jyoti.mickey, KD5TVI, Karada, Karstbj, Kbdank71, Kbrose, Kenyon, Keycard, Kgfleischmann, Kristof vt, Ksn, Kvng, Kwi, Kzollman, Laserbrian, Lavenderbunny, Leuk he, LiDaobing, Lightdarkness, LittleOldMe, LobStoR, Lspo99, M gol, MARQUIS111, MER-C, Magnus Manske, Mallow40, Maltest, Mandarax, Mannafredo, Manop, Marchash, MarcoTolo, Mav, Mblumber, Mditto, Mercury543210, Mindmatrix, Mintguy, Misza13, Mmmeg, Murjek, Mygerardromance, Naniwako, Nazli, Nealmcb, Nilmerg, Nimiew, Nixdorf, Nkansahrexford, Nubiatech, Nurg, Nyttend, Oalbacha, Oystein, PPBlais, Para, Pash, Pde, Pdelong, Peyre, Phatom87, Phenry, Philadams, Philbert2.71828, Piano non troppo, PierreAbbat, Pinkadelica, Plugwash, Pmsyyz, Pol098, Profchakraborty iitkanpur, Psychocim, Quarl, Quest for Truth, Rabarberski, Ramsey585, Rchandra, RedWolf, Rick Sidwell, Rob.bosch, Robert Brockway, Rohitthakral, Ross Fraser, Rrburke, Rushtoshankar, Ryan Roos, SF007, SLATE, SQL, SalineBrain, SaulPerdomo, Sbmehta, Seikku Kaita, Shahid789, Shirimasen, Shiro jdn, Sideswipe091976, Siipikarja, Simetrical, Simon South, SimonEast, SimonSellick, Slackerhobo, Smalljim, SoWhy, Sollosonic, SpyMagician, Steelmans1980, Stephan Leeds, Stephenb, Steven Zhang, Sujirou, Sun Creator, Svetovid, Syndicate, Taestell, Tagishsimon, TakuyaMurata, Teles, The Anome, The Inedible Bulk, Tiddly Tom, Tide rolls, Tobias Bergemann, Tommy2010, TonyHagale, Tresiden, Tristanb, Truthanado, Tsunanet, Tverbeek, Twasono, Twilsonb, Ulric1313, UncleBubba, Urhixidur, Valenciano, Vanished user 5zariu3jisj0j4irj, VanishingUser, Wavelength, Wernher, Wiki alf, Wikipelli, Wimblykit, Winterheart, WithGLEE, WojPob, Wolf0403, Wolfkeeper, Wolfrock, Woohookitty, WorldTechIT, Wrs1864, Xmm0, Xpanzion, Yk4ever, YordanGeorgiev, Zap Rowsdower, Zbxgscqf, Zhlmmc, Zondor, Zundark, 633 anonymous edits

Image Sources, Licenses and Contributors

Image:ARP Spoofing.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:ARP_Spoofing.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:0x55534C

File:DHCP Server.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DHCP_Server.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Patpat

Image:Domain name space.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Domain_name_space.svg *License*: Public Domain *Contributors*: Bjankuloski06en, GeorgHH, Ggia, Herbythyme, LionKimbro, Ma-Lik, Mdd, Waldir, 12 anonymous edits

Image:An example of theoretical DNS recursion.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:An_example_of_theoretical_DNS_recursion.svg *License*: Public Domain *Contributors*: LionKimbro, Syp, The Anome, Waldir, 4 anonymous edits

Image:DNS in the real world.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DNS_in_the_real_world.svg *License*: Public Domain *Contributors*: LionKimbro, Syp, Waldir, 2 anonymous edits

File:Ipv6 address leading zeros.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv6_address_leading_zeros.svg *License*: Public Domain *Contributors*: Ipv6_address.svg: Indeterminate derivative work: BobbyPeru (talk)

Image:Ipv4 address.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv4_address.svg *License*: Public Domain *Contributors*: Indeterminate

File:World IPv6 launch logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:World_IPv6_launch_logo.svg *License*: Creative Commons Attribution 3.0 *Contributors*: Matma Rex

File:Ipv6 header.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv6_header.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Mro

Image:Full Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Full_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Restricted Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Restricted_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Port Restricted Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Port_Restricted_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Symmetric NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Symmetric_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

License