The Internet Protocol Journal

June 2001 Volume 4, Number 2

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From The Editor

A user of a laptop computer "on the road" typically connects to the Internet in one of two ways. The oldest, and most common method, is to dial into an ISP's network and obtain an IP address using the *Point-to-Point Protocol* (PPP). The other method involves attaching the laptop to a local network (usually via Ethernet) and obtaining an IP address through the *Dynamic Host Configuration Protocol* (DHCP). The "local network" could be anything from the high-speed connection provided in some hotels, to an enterprise network at some corporation or other institution. In all cases, the IP address is fixed for the duration of the network session, and the routing of packets from the laptop back to its "home" network remains a relatively straight-forward task (ignoring NATs, firewalls and other complexities for the moment). Suppose however, the mobile computer is using a wireless connection and traveling between several networks over a short period of time. In this scenario one would still like to maintain network connectivity in a seamless manner. The IETF has been working on Mobile IP to address this problem. Mobile IP is the subject of our first article by Bill Stallings.

The art of cryptography is certainly not new, but its use in computer-communications is a more recent phenomena. The *Data Encryption Standard* (DES) has been widely used since it was standardized in 1977. The strength of a particular encryption scheme depends on the key length and the sophistication of the mathematics involved in transforming the so-called cleartext to the encrypted form. As computers have become more powerful it is now possible to systematically "guess" the 56-bit DES keys in a matter of hours, thus a new encryption standard is needed. This new standard, known as the *Advanced Encryption Standard* (AES), is described by Edgar Danielyan.

Many aspects of computer networking can be described as "controversial," that is, there are strongly held opinions about a particular technology or its use. In this issue we begin a new series of articles labelled "Opinion," hoping to bring out some of the different views held by members of the networking community. We hope you will take issue with some of these columns and send us your own opinion piece. We begin the series with an article by Geoff Huston entitled "The Middleware Muddle." Let us know what you think by sending your comments to **`ipj@cisco.com`**

—*Ole J. Jacobsen, Editor and Publisher*
`ole@cisco.com`

Mobile IP

by William Stallings

In response to the increasing popularity of palm-top and other mobile computers, Mobile IP was developed to enable computers to maintain Internet connectivity while moving from one Internet attachment point to another. Although Mobile IP can work with wired connections, in which a computer is unplugged from one physical attachment point and plugged into another, it is particularly suited to wireless connections.

The term "mobile" in this context implies that a user is connected to one or more applications across the Internet, that the user's point of attachment changes dynamically, and that all connections are automatically maintained despite the change. This scenario is in contrast to a user, such as a business traveler, with a portable computer of some sort who arrives at a destination and uses the computer notebook to dial into an *Internet Service Provider* (ISP).

In this latter case, the user's Internet connection is terminated each time the user moves, and a new connection is initiated when the user dials back in. Each time an Internet connection is established, software in the point of attachment (typically an ISP) is used to obtain a new, temporarily assigned IP address. For each application-level connection (for example, *File Transfer Protocol* [FTP], Web connection), this temporary IP address is used by the user's correspondent. A better term for this kind of use is "nomadic."

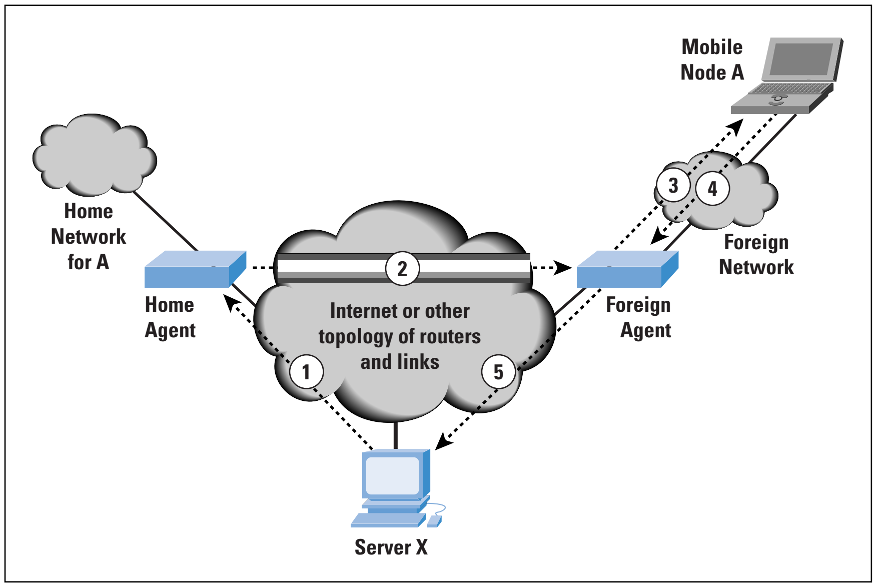
We begin with a general overview of Mobile IP and then look at some of the details.

Operation of Mobile IP

Routers make use of the IP address in an IP datagram to perform routing. In particular, the *network portion* of an IP address is used by routers to move a datagram from the source computer to the network to which the target computer is attached. Then the final router on the path, which is attached to the same network as the target computer, uses the *host portion* of the IP address to deliver the IP datagram to the destination. Further, this IP address is known to the next higher layer in the protocol architecture. In particular, most applications over the Internet are supported by *Transmission Control Protocol* (TCP) connections. When a TCP connection is set up, the TCP entity on each side of the connection knows the IP address of the correspondent host. When a TCP segment is handed down to the IP layer for delivery, TCP provides the IP address. IP creates an IP datagram with that IP address in the IP header and sends the datagram out for routing and delivery. However, with a mobile host, the IP address may change while one or more TCP connections are active.

Figure 1 shows in general terms how Mobile IP deals with the problem of dynamic IP addresses. A mobile node is assigned to a particular network, known as its *home network*. Its IP address on that network, known as its *home address,* is static. When the mobile node moves its attachment point to another network, that is considered a *foreign network* for this host. When the mobile node is reattached, it makes its presence known by registering with a network node, typically a router, on the foreign network known as a *foreign agent*. The mobile node then communicates with a similar agent on the user's home network, known as a *home agent,* giving the home agent the *care-of address* of the mobile node; the care-of address identifies the foreign agent's location. Typically, one or more routers on a network will implement the roles of both home and foreign agents.

Figure 1: Mobile IP Scenario



When IP datagrams are exchanged over a connection between the mobile node (A) and another host (server X in Figure 1), the following operations occur:

1. Server X transmits an IP datagram destined for mobile node A, with A's home address in the IP header. The IP datagram is routed to A's home network.

2. At the home network, the incoming IP datagram is intercepted by the home agent. The home agent encapsulates the entire datagram inside a new IP datagram, which has the A's care-of address in the header, and retransmits the datagram. The use of an outer IP datagram with a different destination IP address is known as *tunneling*.

3. The foreign agent strips off the outer IP header, encapsulates the original IP datagram in a network-level *Protocol Data Unit* (PDU) (for example, a LAN *Logical Link Control* [LLC] frame), and delivers the original datagram to A across the foreign network.

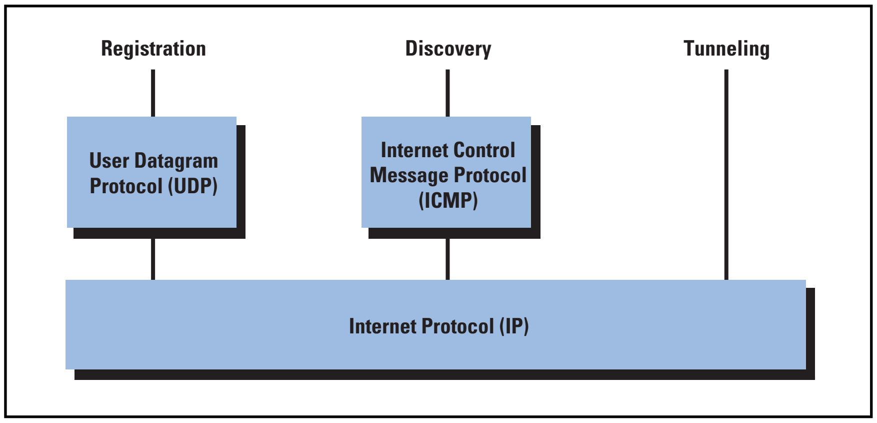
4. When A sends IP traffic to X, it uses X's IP address. In our example, this is a fixed address; that is, X is not a mobile node. Each IP datagram is sent by A to a router on the foreign network for routing to X. Typically, this router is also the foreign agent.
5. The IP datagram from A to X travels directly across the Internet to X, using X's IP address.

To support the operations illustrated in Figure 1, Mobile IP includes three basic capabilities:

- *Discovery:* A mobile node uses a discovery procedure to identify prospective home agents and foreign agents.
- *Registration:* A mobile node uses an authenticated registration procedure to inform its home agent of its care-of address.
- *Tunneling:* Tunneling is used to forward IP datagrams from a home address to a care-of address.

Figure 2 indicates the underlying protocol support for the Mobile IP capability. The registration protocol communicates between an application on the mobile node and an application in the home agent, and hence uses a transport-level protocol. Because registration is a simple request/response transaction, the overhead of the connection-oriented TCP is not required, and, therefore, the *User Datagram Protocol* (UDP) is used as the transport protocol. Discovery makes use of the existing *Internet Control Message Protocol* (ICMP) by adding the appropriate extensions to the ICMP header. ICMP is a connectionless protocol well suited for the discovery operation. Finally, tunneling is performed at the IP level.

Figure 2: Protocol Support for Mobile IP


Registration
Discovery
Tunneling
User Datagram Protocol (UDP)
Internet Control Message Protocol (ICMP)
Internet Protocol (IP)

Discovery

The discovery process in Mobile IP is very similar to the router advertisement process defined in ICMP. Accordingly, agent discovery makes use of ICMP router advertisement messages, with one or more extensions specific to Mobile IP.

The mobile node is responsible for an ongoing discovery process. It must determine if it is attached to its home network, in which case IP datagrams may be received without forwarding, or if it is attached to a foreign network.

Because handoff from one network to another occurs at the physical layer, a transition from the home network to a foreign network can occur at any time without notification to the network layer (that is, the IP layer). Thus, discovery for a mobile node is a continuous process.

For the purpose of discovery, a router or other network node that can act as an agent periodically issues a router advertisement ICMP message with an advertisement extension. The router advertisement portion of the message includes the IP address of the router. The advertisement extension includes additional information about the role of the router as an agent, as discussed subsequently. A mobile node listens for these *agent advertisement messages*. Because a foreign agent could be on the home network of the mobile node (set up to serve visiting mobile nodes), the arrival of an agent advertisement does not necessarily tell the mobile node that it is on a foreign network. The mobile node must compare the network portion of the router IP address with the network portion of its own home address. If these network portions do not match, then the mobile node is on a foreign network.

The *agent advertisement extension* follows the ICMP router advertisement fields and consists of the following fields:

- *Type:* 16, indicates that this is an agent advertisement.
- *Length:* (6 + 4*N*), where *N* is the number of care-of addresses advertised.
- *Sequence number:* The count of agent advertisement messages sent since the agent was initialized.
- *Lifetime:* The longest lifetime, in seconds, that this agent is willing to accept a registration request from a mobile node.
- *R:* Registration with this foreign agent is required (or another foreign agent on this network). Even those mobile nodes that have already acquired a care-of address from this foreign agent must reregister.
- *B:* Busy. The foreign agent will not accept registrations from additional mobile nodes.
- *H:* This agent offers services as a home agent on this network.
- *F:* This agent offers services as a foreign agent on this network.
- *M:* This agent can receive tunneled IP datagrams that use minimal encapsulation, explained subsequently.
- *G:* This agent can receive tunneled IP datagrams that use *Generic Routing Encapsulation* (GRE), explained subsequently.
- *Y:* This agent supports the use of Van Jacobson header compression, an algorithm defined in RFC 1144 for compressing fields in the TCP and IP headers.
- *Care-of address:* The care-of address or addresses supported by this agent on this network. There must be at least one such address if the F bit is set. There may be multiple addresses.

There may also be an optional *prefix-length extension* following the advertisement extension. This extension indicates the number of bits in the router address that define the network number. The mobile node uses this information to compare the network portion of its own IP address with the network portion of the router. The fields include the following:

- *Type:* 19, indicates that this is a prefix-length advertisement.
- *Length: N,* where *N* is the value of the Num Addrs field in the ICMP router advertisement portion of this ICMP message. In other words, this is the number of router addresses listed in this ICMP message.
- *Prefix length:* The number of leading bits that define the network number of the corresponding router address listed in the ICMP router advertisement portion of this message. The number of prefix length fields matches the number of router address fields (*N*).

Foreign agents are expected to periodically issue agent advertisement messages. If a mobile node needs agent information immediately, it can issue an ICMP router solicitation message. Any agent receiving this message will then issue an agent advertisement.

As was mentioned, a mobile node may move from one network to another because of some handoff mechanism, without the IP level being aware of it. The agent discovery process is intended to enable the agent to detect such a move. The agent may use one of two algorithms for this purpose:

- *Use of Lifetime field:* When a mobile node receives an agent advertisement from a foreign agent that it is currently using or that it is now going to register with, it records the Lifetime field as a timer. If the timer expires before the agent receives another agent advertisement from the agent, then the node assumes that it has lost contact with that agent. If, in the meantime, the mobile node has received an agent advertisement from another agent and that advertisement has not yet expired, the mobile node can register with this new agent. Otherwise, the mobile node should use agent solicitation to find an agent.
- *Use of network prefix:* The mobile node checks whether any newly received agent advertisement is on the same network as the current care-of address of the node. If it is not, the mobile node assumes that it has moved and may register with the agent whose advertisement the mobile node has just received.

The discussion so far has involved the use of a care-of address associated with a foreign agent; that is, the care-of address is an IP address for the foreign agent. This foreign agent will receive datagrams at this care-of address, intended for the mobile node, and then forward them across the foreign network to the mobile node. However, in some cases a mobile node may move to a network that has no foreign agents or on which all foreign agents are busy.

As an alternative, the mobile node may act as its own foreign agent by using a *colocated care-of address*. A colocated care-of address is an IP address obtained by the mobile node that is associated with the current interface to a network of that mobile node.

The means by which a mobile node acquires a colocated address is beyond the scope of Mobile IP. One means is to dynamically acquire a temporary IP address through an Internet service such as *Dynamic Host Configuration Protocol* (DHCP). Another alternative is that the colocated address may be owned by the mobile node as a long-term address for use only while visiting a given foreign network.

Registration

When a mobile node recognizes that it is on a foreign network and has acquired a care-of address, it needs to alert a home agent on its home network and request that the home agent forward its IP traffic. The registration process involves four steps:

1. The mobile node requests the forwarding service by sending a registration request to the foreign agent that the mobile node wants to use.
2. The foreign agent relays this request to the home agent of that mobile node.
3. The home agent either accepts or denies the request and sends a registration reply to the foreign agent.
4. The foreign agent relays this reply to the mobile node.

If the mobile node is using a colocated care-of address, then it registers directly with its home agent, rather than going through a foreign agent.

The registration operation uses two types of messages, carried in UDP segments. The *registration request message* consists of the following fields:

- *Type:* 1, indicates that this is a registration request.
- *S:* Simultaneous bindings. The mobile node is requesting that the home agent retain its prior mobility bindings. When simultaneous bindings are in effect, the home agent will forward multiple copies of the IP datagram, one to each care-of address currently registered for this mobile node. Multiple simultaneous bindings can be useful in wireless handoff situations to improve reliability.
- *B:* Broadcast datagrams. Indicates that the mobile node would like to receive copies of broadcast datagrams that it would have received if it were attached to its home network.
- *D:* Decapsulation by mobile node. The mobile node is using a colocated care-of address and will decapsulate its own tunneled IP datagrams.
- *M:* Indicates that the home agent should use minimal encapsulation, explained subsequently.

- *V:* Indicates that the home agent should use Van Jacobson header compression, an algorithm defined in RFC 1144 for compressing fields in the TCP and IP headers.
- *G:* Indicates that the home agent should use GRE encapsulation, explained subsequently.
- *Lifetime:* The number of seconds before the registration is considered expired. A value of zero is a request for deregistration.
- *Home address:* The home IP address of the mobile node. The home agent can expect to receive IP datagrams with this as a destination address, and must forward those to the care-of address.
- *Home agent:* The IP address of the mobile node home agent. This informs the foreign agent of the address to which this request should be relayed.
- *Care-of address:* The IP address at this end of the tunnel. The home agent should forward IP datagrams that it receives with the mobile node home address to this destination address.
- *Identification:* A 64-bit number generated by the mobile node, used for matching registration requests to registration replies and for security purposes, as explained subsequently.
- *Extensions:* The only extension so far defined is the authentication extension, explained subsequently.

The *registration reply message* consists of the following fields:

- *Type:* 3, indicates that this is a registration reply.
- *Code:* Indicates result of the registration request.
- *Lifetime:* If the code field indicates that the registration was accepted, the number of seconds before the registration is considered expired. A value of zero indicates that the mobile node has been deregistered.
- *Home address:* The home IP address of the mobile node.
- *Home agent:* The IP address of the mobile node home agent.
- *Identification:* A 64-bit number used for matching registration requests to registration replies.

The only extension so far defined is the authentication extension, explained subsequently.

A key concern with the registration procedure is security. Mobile IP is designed to resist two types of attacks:

1. A node may pretend to be a foreign agent and send a registration request to a home agent so as to divert traffic intended for a mobile node to itself.
2. A malicious agent may replay old registration messages, effectively cutting the mobile node from the network.

The technique that is used to protect against such attacks involves the use of message authentication and the proper use of the identification field of the registration request and reply messages.

For purposes of message authentication, each registration request and reply contains an *authentication extension* with the following fields:

- *Type:* Used to designate the type of this authentication extension.
- *Length:* 4 plus the number of bytes in the authenticator.
- *Security parameter index (SPI):* An index that identifies a security context between a pair of nodes. This security context is configured so that the two nodes share a secret key and parameters relevant to this association (for example, authentication algorithm).
- *Authenticator:* A code used to authenticate the message. The sender inserts this code into the message using a shared secret key. The receiver uses the code to ensure that the message has not been altered or delayed. The authenticator protects the entire registration request or reply message, any extensions prior to this extension, and the type and length fields of this extension.

The default authentication algorithm uses keyed MD5 to produce a 128-bit message digest. For Mobile IP, a "prefix+suffix" mode of operation is used. The MD5 digest is computed over the shared secret key, followed by the protected fields from the registration message, followed by the shared secret key again. Three types of authentication extensions are defined:

- *Mobile-home:* This extension must be present and provides for authentication of the registration messages between the mobile node and the home agent.
- *Mobile-foreign:* The extension may be present when a security association exists between the mobile node and the foreign agent. The agent will strip this extension off before relaying a request message to the home agent and add this extension to a reply message coming from a home agent.
- *Foreign-home:* The extension may be present when a security association exists between the foreign agent and the home agent.

Note that the authenticator protects the identification field in the request and reply messages. As a result, the identification value can be used to thwart replay types of attacks. As was mentioned, the identification value enables the mobile node to match a reply to a request. Further, if the mobile node and the home agent maintain synchronization so that the home agent can distinguish a reasonable identification value from a suspicious one, then the home agent can reject suspicious messages. One way to do this is to use a timestamp value. As long as the mobile node and home agent have reasonably synchronized values of time, the timestamp will serve the purpose. Alternatively, the mobile node could generate values using a pseudorandom number generator. If the home agent knows the algorithm, then it knows what identification value to expect next.

Tunneling

When a mobile node is registered with a home agent, the home agent must be able to intercept IP datagrams sent to the mobile node home address so that these datagrams can be forwarded via tunneling. The standard does not mandate a specific technique for this purpose but references *Address Resolution Protocol* (ARP) as a possible mechanism. The home agent needs to inform other nodes on the same network (the home network) that IP datagrams with a destination address of the mobile node in question should be delivered (at the link level) to this agent. In effect, the home agent steals the identity of the mobile node in order to capture packets destined for that node that are transmitted across the home network.

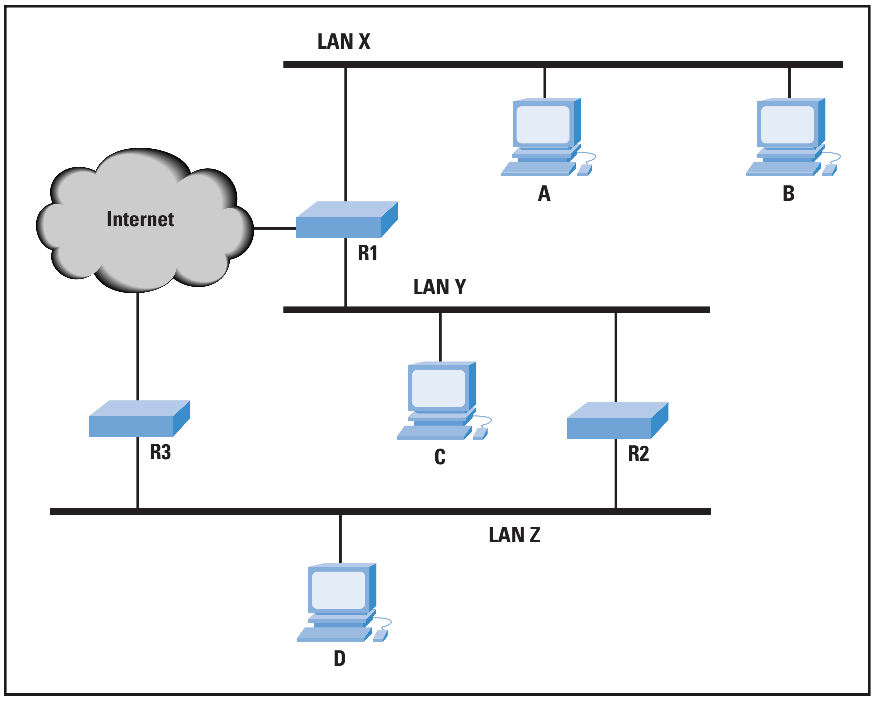


Figure 3: A Simple Internetworking Example

For example, suppose that R3 in Figure 3 is acting as the home agent for a mobile node that is attached to a foreign network elsewhere on the Internet. That is, there is a host H whose home network is LAN Z that is now attached to some foreign network. If host D has traffic for H, it will generate an IP datagram with H's home address in the IP destination address field. The IP module in D recognizes that this destination address is on LAN Z and so passes the datagram down to the link layer with instructions to deliver it to a particular *Media Access Control* (MAC)-level address on Z. Prior to this time, R3 has informed the IP layer at D that datagrams destined for that particular address should be sent to R3. Thus, the MAC address of R3 is inserted by D in the destination MAC address field of the outgoing MAC frame. Similarly, if an IP datagram with the mobile node home address arrives at router R2, it recognizes that the destination address is on LAN Z and will attempt to deliver the datagram to a MAC-level address on Z. Again, R2 has previously been informed that the MAC-level address it needs corresponds to R3.

For traffic that is routed across the Internet and arrives at R3 from the Internet, R3 must simply recognize that for this destination address, the datagram is to be captured and forwarded.

To forward an IP datagram to a care-of address, the home agent puts the entire IP datagram into an outer IP datagram. This is a form of encapsulation, just as placing an IP header in front of a TCP segment encapsulates the TCP segment in an IP datagram. Three options for encapsulation are allowed for Mobile IP and we will review the first two of the following options:

- *IP-within-IP encapsulation:* This is the simplest approach, defined in RFC 2003.
- *Minimal encapsulation:* This approach involves fewer fields, defined in RFC 2004.
- *Generic routing encapsulation (GRE):* This is a generic encapsulation procedure, defined in RFC 1701, that was developed prior to the development of Mobile IP.

In the IP-within-IP encapsulation approach, the entire IP datagram becomes the payload in a new IP datagram (Figure 4a). The inner, original IP header is unchanged except to decrement *Time To Live* (TTL) by 1. The outer header is a full IP header. Two fields (indicated as unshaded in the figure) are copied from the inner header. The version number is 4, the protocol identifier for IPv4, and the type of service requested for the outer IP datagram is the same as that requested for the inner IP datagram.

Figure 4a: Mobile IP Encapsulation

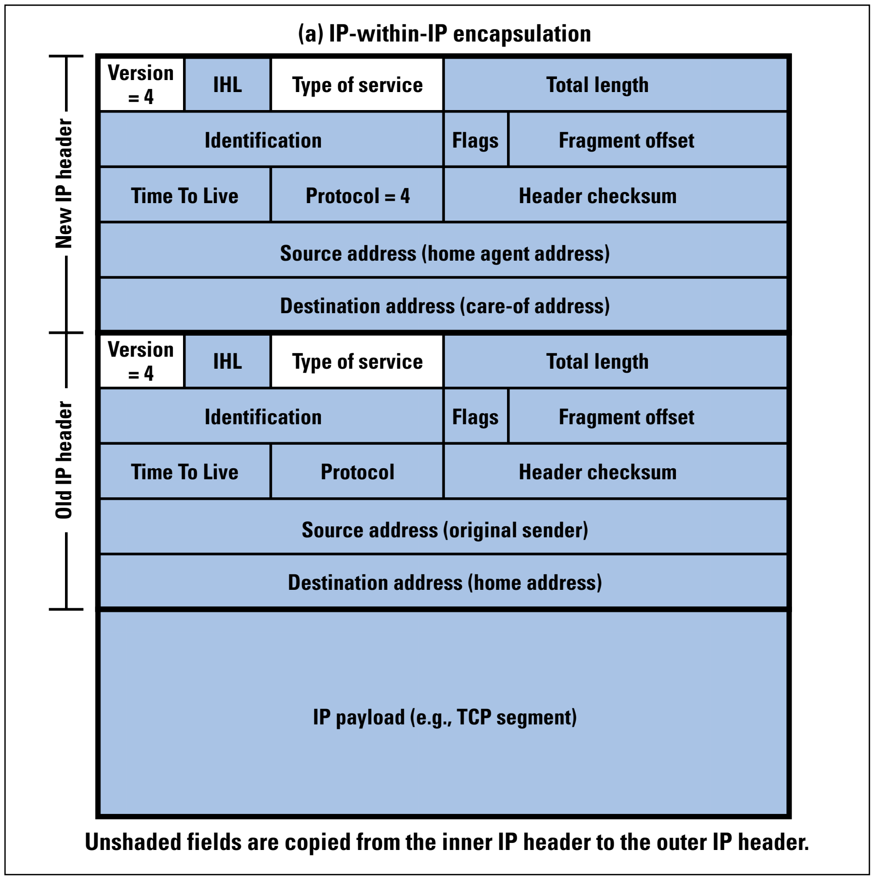
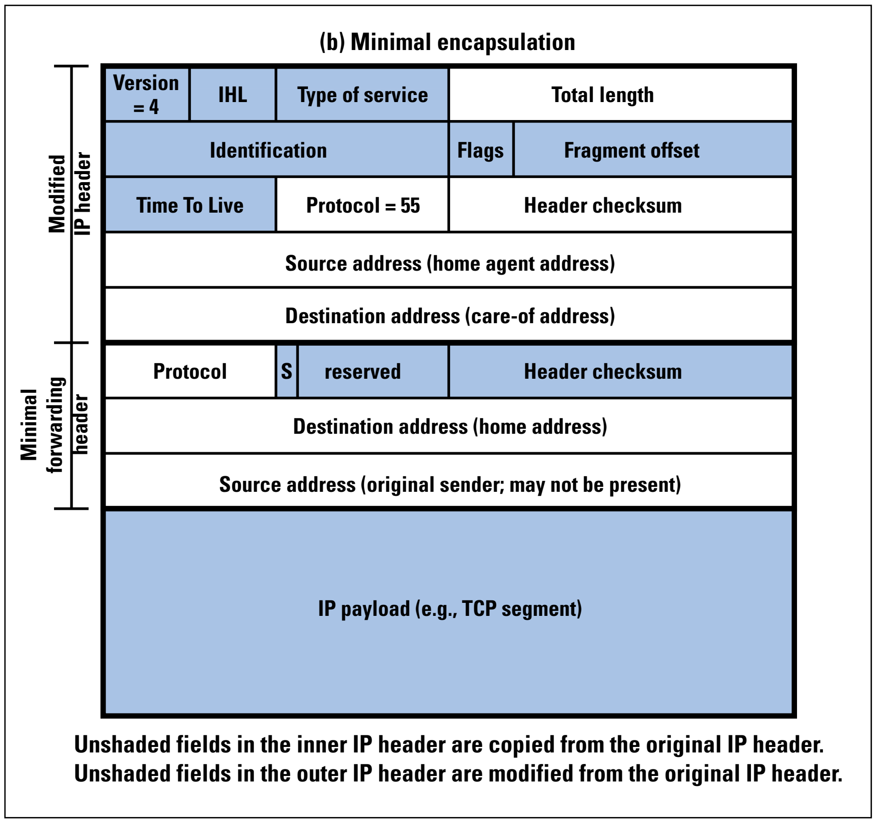

(a) IP-within-IP encapsulation
New IP header
Version = 4
IHL
Type of service
Total length
Identification
Flags
Fragment offset
Time To Live
Protocol = 4
Header checksum
Source address (home agent address)
Destination address (care-of address)
Old IP header
Version = 4
IHL
Type of service
Total length
Identification
Flags
Fragment offset
Time To Live
Protocol
Header checksum
Source address (original sender)
Destination address (home address)
IP payload (e.g., TCP segment)
Unshaded fields are copied from the inner IP header to the outer IP header.

Figure 4b: Mobile IP Encapsulation


(b) Minimal encapsulation
Version = 4
IHL
Type of service
Total length
Identification
Flags
Fragment offset
Time To Live
Protocol = 55
Header checksum
Source address (home agent address)
Destination address (care-of address)
Protocol
S
reserved
Header checksum
Destination address (home address)
Source address (original sender; may not be present)
IP payload (e.g., TCP segment)
Modified IP header
Minimal forwarding header
Unshaded fields in the inner IP header are copied from the original IP header.
Unshaded fields in the outer IP header are modified from the original IP header.

In the inner IP header, the source address refers to the host that is sending the original datagram, and the destination address is the home address of the intended recipient. In the outer IP header, the source and destination addresses refer to the entry and exit points of the tunnel. Thus, the source address typically is the IP address of the home agent, and the destination address is the care-of address for the intended destination.

Example: Consider an IP datagram that originates at server X in Figure 1 and that is intended for mobile node A. The original IP datagram has a source address equal to the IP address of X and a destination address equal to the IP home address of A. The network portion of A's home address refers to A's home network, so the datagram is routed through the Internet to A's home network, where it is intercepted by the home agent. The home agent encapsulates the incoming datagram with an outer IP header, which includes a source address equal to the IP address of the home agent and a destination address equal to the IP address of the foreign agent on the foreign network to which A is currently attached. When this new datagram reaches the foreign agent, it strips off the outer IP header and delivers the original datagram to A.

Minimal encapsulation results in less overhead and can be used if the mobile node, home agent, and foreign agent all agree to do so. With minimal encapsulation, the new header is inserted between the original IP header and the original IP payload (Figure 4b). It includes the following fields:

- *Protocol:* Copied from the Destination Address field in the original IP header. This field identifies the protocol type of the original IP payload and thus identifies the type of header that begins the original IP payload.
- *S:* If 0, the original source address is not present, and the length of this header is 8 octets. If 1, the original source address is present, and the length of this header is 12 octets.
- *Header checksum:* Computed over all the fields of this header.
- *Original destination address:* Copied from the Destination Address field in the original IP header.
- *Original source address:* Copied from the Source Address field in the original IP header. This field is present only if the S bit is 1. The field is not present if the encapsulator is the source of the datagram (that is, the datagram originates at the home agent).

The following fields in the original IP header are modified to form the new outer IP header:

- *Total length:* Incremented by the size of the minimal forwarding header (8 or 12).
- *Protocol:* 55; this is the protocol number assigned to minimal IP encapsulation.
- *Header checksum:* Computed over all the fields of this header; because some of the fields have been modified, this value must be recomputed.
- *Source address:* The IP address of the encapsulator, typically the home agent.
- *Destination address:* The IP address of the exit point of the tunnel. This is the care-of address and may be either the IP address of the foreign agent or the IP address of the mobile node (in the case of a colocated care-of address).

The processing for minimal encapsulation is as follows. The encapsulator (home agent) prepares the encapsulated datagram with the format of Figure 4b. This datagram is now suitable for tunneling and is delivered across the Internet to the care-of address. At the care-of address, the fields in the minimal forwarding header are restored to the original IP header and the forwarding header is removed from the datagram. The total length field in the IP header is decremented by the size of the minimal forwarding header (8 or 12) and the header checksum field is recomputed.

References

Reference [1] is a good survey article on mobile IP; a somewhat less technical, more business-oriented description from the same author is [2]. For greater detail, see [3]. The August 2000 issue of *IEEE Personal Communications* contains numerous articles on enhancements to the current Mobile IP standard. The Web site of the IETF Working Group on Mobile IP, which contains current RFCs and Internet Drafts is at:
`http://ietf.org/html.charters/mobileip-charter.html`

[1] Perkins, C., "Mobile IP," *IEEE Communications Magazine,* May 1997.

[2] Perkins, C., "Mobile Networking through Mobile IP," *IEEE Internet Computing,* January-February 1998.

[3] Perkins, C., *Mobile IP: Design Principles and Practices,* ISBN 0-201-63469-4, Prentice Hall PTR, 1998.

[4] Solomon, J., *Mobile IP: The Internet Unplugged,* ISBN 0138562466, Prentice Hall PTR, 1998.

WILLIAM STALLINGS is a consultant, lecturer, and author of over a dozen books on data communications and computer networking. He also maintains a computer science resource site for CS students and professionals at **`WilliamStallings.com/StudentSupport.html`**. He has a PhD in computer science from M.I.T. His latest book is *Wireless Communications and Networks* (Prentice Hall, 2001). His home in cyberspace is **`WilliamStallings.com`** and he can be reached at **`ws@shore.net`**

Goodbye DES, Welcome AES

by Edgar Danielyan

Much has changed since introduction of the *Data Encryption Standard* (DES)[2] in 1977. Our hardware is faster, we have more memory, and the use of computer networks in all areas of human activity is increasing. The widely used DES has, on several occasions, been proven to be inadequate for many applications—especially those involving the transmission of sensitive information over public networks such as the Internet, where the entire transmission may be intercepted and cryptanalyzed. Specialized hardware has been built that can determine the 56-bit DES key in a few hours. These considerations, and others, have signaled that a new standard algorithm and longer keys are necessary.

Fortunately, in January 1997, the U.S. *National Institute of Standards and Technology* (NIST) announced that it's time for a new encryption standard: the *Advanced Encryption Standard* (AES). They formalized their requirements and issued a call for candidate algorithm nominations in September 1997. The deadline for submissions was June 1998, when a total of 15 algorithms were submitted for consideration. This article shows why DES is outdated and should not be used for any purposes that require serious encryption. It also provides a brief description of the soon-to-come replacement of DES, the Advanced Encryption Standard.

Data Encryption Standard

Published as the U.S. Federal Information Processing Standard 46 in 1977, DES is still widely used, despite being proven inadequate for use in many applications. It is a symmetric block cipher (shared secret key), with its block size fixed at 64 bits. There are four defined modes of operation, with the *Electronic Code Book* (ECB) mode being the most widely used[1]. Additionally, DES has been incorporated into numerous other standards, such as American Bankers Association's *Protection of Personal Identification Numbers in Interchange Standard, Management and Use of Personal Identification Numbers Standard, Key Management Standard,* and three ANSI standards, *Data Encryption Algorithm* (DEA), *Standard for Personal Identification Number (PIN) Management and Security,* and *Standard for Financial Institution Message Authentication*[3]. In particular, DES is also specified as an approved algorithm in the *IP Security Architecture* (IPSec) standard[9], which is used in the equipment from many different suppliers.

Key Length

Key length is one of the two most important security factors of any encryption algorithm—the other one being the design of the algorithm itself. DES uses a 64-bit block for the key; however, 8 of these bits are used for odd parity and are, therefore, not counted in the key length. The effective key length is then calculated as 56 bits, giving 2^{56} possible keys. A true 64-bit key has 256 times as many keys, whereas a 128-bit key is 2^{72} times "better" than a 56-bit key. As if this was not enough, DES also has so-called *weak* and semi-*weak* keys. During the encryption process, the key is used to generate two values that are used for separate purposes during the process. These 16 weak and semi-weak keys will produce values that don't appear to be random. They will give outputs of all-ones, all-zeros, or distinguishable patterns of ones and zeros. It is generally recognized that these 16 key values should not be used. The key length was known to be a factor in trusting DES soon after DES was published. For this reason, people started exploring the use of multiple encryption passes and multiple keys. *Triple DES* (3DES) is a way of using DES encryption three times.

The most common method is to first encrypt the data block with one key. The output of this operation is run through the decryption process with a second key, and the output of that operation is run through the encryption process again with the first key. This process makes the effective key length 112 bits long. Again, the problem with weak and semi-weak keys remains. The disadvantage of Triple DES is that it is about one-third as fast as DES when processing data. This effort just slightly extended the life of DES while a suitable alternative could be found.

Breaking the DES

In addition to the brute-force key search (for example, trying every possible key in order to recover the plaintext—for DES that would be 2^{56} keys), there is also a technique known as *cryptanalysis*, which may be used to find the key or the plaintext. Essentially, there are two publicized ways to cryptanalyze DES: *differential* and *linear*. Discovered by Biham and Shamir in 1990, differential cryptanalysis was previously unknown to the public. In short, differential cryptanalysis looks at the difference between pairs of ciphertext and uses the information about these differences to find the key. Linear cryptanalysis, discovered by M. Matsui, on the other hand, uses a method called *linear approximations* to analyze block ciphers (not only DES). Because some internal structures used in DES are not designed to be strong against linear cryptanalysis, it is quite effective when used against DES. To show that the DES is inadequate and should not be used in important systems anymore, RSA Data Security[7] sponsored a challenge to see how long it would take to decrypt successively more difficult algorithms (see **`http://www.rsasecurity.com/rsalabs/challenges`** for more information). Two organizations played key roles in breaking the DES: the distributed.net and the *Electronic Frontier Foundation* (EFF).

distributed.net

distributed.net[6] is a worldwide distributed computing network. Started in 1997, the company now has thousands of participants who are contributing their idle computing power to provide an equivalent of about 160,000 Pentium II computers working in parallel. The company's mission statement says, in particular:

"We will deploy our software to form an immense, globally distributed computer that solves large-scale problems and provides an accessible pool of computational power to projects that need it. This deployment will also demonstrate the real-world utility of both distributed computing in general and our software in particular."

It may be said that they are doing well: projects undertaken and successfully completed by distributed.net include the CS Cipher, DES III, DES II 2, and RC5-56 challenges. At the time of writing, distributed.net is working on two projects: breaking RC5 with a 64-bit key and finding *Optimal Golomb Rulers* (OGRs). The idea behind distributed.net is that it is possible to distribute chunks of data over the Internet to be processed in parallel by participating computers during their idle time. The results of these calculations are then sent to a central computer that coordinates the distributed computation. The same principle is used by the SETI (Search for Extraterrestrial Intelligence) @ Home project.

Electronic Frontier Foundation

The EFF's DES cracking computer was designed by Cryptography Research, Advanced Wireless Technologies, and the EFF[5]. The design was based upon theoretical work by Michael Wiener[10]. It checked 90 billion keys per second, was assembled in six Sun 2 cabinets, and had 27 boards and 1800 custom chips. Built for less than $250,000, it found the key in approximately 56 hours of brute-force search.

DES I

The DES I contest was the first attempt to prove that DES is no longer fit for any serious use. It was completed on June 17, 1997, by R. Verser in a collaborative effort, after checking about 14 percent (10,178,478, 175,420,416 keys) of the key space. It took 84 days.

DES II

There were, in fact, two DES II challenges. distributed.net participated in the first one, which began on January 13, 1998, and completed it on February 23, 1998. About 63 quadrillion keys were checked. At the end, the participants of distributed.net were checking 28 gigakeys per second. The decrypted text was "The unknown message is: Many hands make light work." The EFF won the second challenge on July 15, 1998, in less than three days, with distributed.net coming in second. This time the plaintext read "It's time for those 128-, 192-, and 256-bit keys."

DES III

The DES III contest, announced by RSA Data Security on December 12, 1998, to start on January 18, 1999, was also a success. In an official press release, RSA said:

"First adopted by the federal government in 1977, the 56-bit DES algorithm is still widely used by financial services and other industries to protect sensitive on-line applications, despite growing doubts about its vulnerability to hackers. It has been widely known that 56-bit keys, such as those offered by the government's DES standard, offer marginal protection against a committed adversary."

It took 22 hours and 15 minutes for Electronic Frontier Foundation's Deep Crack computer and distributed.net's worldwide distributed computing network to find out the 56-bit DES key, decipher the message, and win the $10,000 contest. The decrypted message read "See you in Rome (Second AES Conference, March 22–23, 1999)" and was found after checking about 30 percent of the key space. This latest exercise finally proved that DES belongs to the past.

AES Timeline

In April 1997, NIST organized a workshop to consider criteria and submission guidelines of candidate algorithms; later in September, an official call for nominations was published in the U.S. Federal Register. By June 1998, 15 algorithms were submitted to the NIST for consideration:

- CAST-256 (Entrust Technologies)
- CRYPTON (Future Systems)
- DEAL (Richard Outerbridge, Lars Knudsen)
- DFC (National Centre for Scientific Research, France)
- E2 (NTT)
- FROG (TecApro Internacional)
- HPC (Rich Schroeppel)
- LOKI97 (Lawrie Brown, Josef Pieprzyk, Jennifer Seberry)
- MAGENTA (Deutsche Telekom)
- Mars (IBM)
- RC6 (RSA)
- Rijndael (Joan Daemen, Vincent Rijmen)
- Safer+ (Cylink)
- Serpent (Ross Anderson, Eli Biham, Lars Knudsen)
- Twofish (Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson)

NIST asked for public comments on these 15 algorithms and set the date for the second AES candidate conference to March 1999, to be held in Rome, Italy. The candidate algorithms were tested from both cryptological and performance viewpoints. One of the original NIST requirements for the algorithm was that it had to be efficient both in software and hardware implementations. (DES was originally practical only in hardware implementations.) Java and C reference implementations were used to do performance analysis of the algorithms. A few months later, a NIST press release announced the selection of 5 out of 15 algorithms that survived rigorous testing and cryptanalysis. This fact is not to say that the algorithms that were not selected were broken or were without merit. Those algorithms either were not as efficient, or were not as practical to implement.

The selected algorithms were Mars, RC6, Rijndael, Serpent, and Twofish. These algorithms were accepted as cryptologically strong and flexible, as well as able to be efficiently implemented in software and hardware. In August 2000, the National Security Agency published the VHDL model for performance testing of algorithms when implemented in hardware. Finally, in October 2000, a NIST press release announced the selection of Rijndael as the proposed Advanced Encryption Standard.

Rijndael

Rijndael[4] (pronounced "Reign Dahl," "Rain Doll," or "Rhine Dahl") was designed by Joan Daemen, PhD (Proton World International, Belgium) and Vincent Rijmen (Catholic University of Leuven, Belgium). Both authors are internationally known cryptographers. Rijndael is an efficient, symmetric block cipher. It supports key and block sizes of 128, 192, and 256 bits. The main design goals for the algorithm were simplicity, performance, and strength (that is, resistance against cryptanalysis). When used in *Cipher Block Chaining Message Authentication Code* (CBC MAC) mode, Rijndael can be used as a MAC algorithm; it also may be used as a hash function and as a pseudo random number generator (both are special mathematical functions widely used in cryptography; an example of a hash function is *Message Digest 5* (MD5)—a popular message digest algorithm by Ron Rivest). In their specification of the algorithm, the authors specifically state the strength of Rijndael against differential, truncated differential, linear, interpolation, and Square attacks. Although Rijndael is not based on Square[8], some ideas from the Square algorithm design are used in Rijndael.

Square is a 128-bit symmetric iterated block cipher designed by Daemen, Rijnmen, and Knudsen. Its primary design goal was strength against both linear and differential cryptanalyses; the high degree of parallelism of the Square algorithm allows efficient implementation on parallel computers.

Of course, the length of the key is also very important, especially because the most efficient known attack against Rijndael is an exhaustive key search. It would take 2^{255} runs of Rijndael to find a key 256 bits long. To the credit of the authors, Rijndael does not use "parts" or tables from other algorithms, making it easy to implement alone.

Table 1: Comparing DES and AES

	DES	AES
Key Length	56 bits	128, 192, or 256 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher
Block Size	64 bits	128, 192, or 256 bits
Developed	1977	2000
Cryptanalysis resistance	Vulnerable to differential and linear cryptanalysis; weak substitution tables	Strong against differential, truncated differential, linear, interpolation and Square attacks
Security	Proven inadequate	Considered secure
Possible Keys	2^{56}	2^{128}, 2^{192}, or 2^{256}
Possible ASCII printable character keys*	95^{7}	95^{16}, 95^{24}, or 95^{32}
Time required to check all possible keys at 50 billion keys per second**	For a 56-bit key: 400 days	For a 128-bit key: 5×10^{21} years

* When a text password input by a user is used for encryption (there are 95 printable characters in ASCII).

** In theory, the key may be found after checking 1/2 of the key space. The time shown is 100% of the key space.

Summary

It is expected that AES will be officially published as a *Federal Information Processing Standard* (FIPS) in April–June 2001, and implementations of AES in various security systems probably will surface shortly thereafter. In the meantime, authoritative information on AES developments may be found on NIST's Web site at **`http://csrc.nist.gov/encryption/aes/`**. The full mathematical specification of the algorithm and reference implementations in C and Java are also available from the same Web site.

References

[1] *Applied Cryptography,* 2nd edition, by Bruce Schneier, 1996, John Wiley & Sons.

[2] National Institute of Standards and Technology (NIST), **http://www.nist.gov**

[3] American National Standards Institute (ANSI), **http://www.ansi.org**

[4] The Rijndael Specification, **http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf**

[5] Electronic Frontier Foundation, **http://www.eff.org**

[6] distributed.net, **http://www.distributed.net**

[7] RSA Security, **http://www.rsa.com**

[8] Square Specification, **http://www.esat.kuleuven.ac.be/~rijmen/square**

[9] Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol," RFC 2401, November 1998.

[10] Michael Wiener, "Efficient DES Key Search," Proceedings of the CRYPTO'93 Conference, August 1993.

[11] Madson, C., Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV," RFC 2405, November 1998.

[A prior version of this article was published in the February 2001 issue of the *;login:* magazine].

EDGAR DANIELYAN is a Cisco Certified Network, Design and Security Professional, as well as member of ACM, USENIX, SAGE, and the IEEE Computer Society. He has worked for a national telco, a bank, the United Nations, and the Ministry of Defense, among others. Currently self-employed, he consults and writes on internetworking, UNIX, and security. E-mail: **edd@danielyan.com**

Opinion: The Middleware Muddle

by Geoff Huston

[This occasional column is an individual soapbox on views of various aspects of the Internet. The views stated here are intended to be mildly provocative, and, if backed to the wall, the author will rapidly disclaim any responsibility for them whatsoever!]

It is not often that an entire class of technology can generate an emotive response. But, somehow, middleware has managed to excite many strong reactions. For some *Internet Service Providers* (ISPs), middleware—in the form of *Web caches*—is not only useful, it's critical to the success of their enterprise. For many corporate networks, middleware—in the form of *firewalls*—is the critical component of their network security measures. For such networks, middleware is an integral part of the network. Other networks use middleware, in the form of *Network Address Translators* (NATs), as a means of stretching a limited number of Internet public addresses to provide connectivity services to a much larger local network. For others, middleware is seen as something akin to network heresy. For them, not only does middleware often break the basic semantics of the Internet Protocol, it is also in direct contravention to the end-to-end architecture of the Internet. Middleware, they claim, breaks the operation of entire classes of useful applications, and this makes the Internet a poorer network as a result.

Emotions have run high in the middleware debate, and middleware has been portrayed as being everything from absolutely essential to the operation of the Internet as we know it, to being immoral and deceptive. Strong stuff indeed from an engineering community, even one as traditionally opinionated as Internet engineers.

So what is middleware all about and why the fuss?

It may be helpful to start with a definition of middleware. One definition of middleware is that of anything in the network that functions at a level in a network reference model above that of end-to-end transport (TCP/IP), and below that of the application environment (the *Application Programming Interface* [API])[1]. Of course, this definition encompasses a very broad class of services that covers everything from *Authentication, Authorization, and Accounting* (AAA) servers and *Domain Name System* (DNS) servers through to various forms of information discovery services and resource management.

Another possible definition of middleware adopts the perspective of the integrity of the end-to-end model of Internet architecture[2]. From this perspective, middleware is a class of network devices that do something other than forward or discard an IP packet onward along the next hop to the destination address of the packet—in other words, anything other than a packet-switching element that sits in the transmission path of the packet.

With such an end-to-end definition of middleware, these middleware units may intercept the packet and alter the header or payload of the packet, redirect the packet to be delivered to somewhere other than its intended destination, or process the packet as if it were addressed to the middleware device itself. From this perspective, AAA, the DNS, and related services from our first definition are simply applications that traverse the network.

There's nothing like confusion over definitions to fuel a debate, and this area is no exception. However, a debate over definitions is too often a dry one. So, in the interest of adding a little more incendiary material to the topic, let's simply use this second definition of middleware to look further at the issues.

Why would a network go to all this bother to trap and process certain packets? Surely it's easier and cheaper to simply forward the packet onward to its intended destination? The answer can be "yes" or "no," depending on how you feel about the role of middleware in TCP/IP.

An Example: Cache Middleware

Let's look at this in a bit more detail, using a specific flavor of middleware to illustrate the middleware dilemma. A common form of middleware is the *Transparent Web Cache*. Such a Web cache is constructed using two parts, an *interceptor* and a *cache system*. The interceptor is placed into the network, either as a software module added to a router or as a device, which is spliced into a point-to-point link. The interceptor takes all incoming TCP traffic addressed to port 80 (a *Hypertext Transfer Protocol* [HTTP] session) and redirects it across to the cache system. All other traffic is treated normally. The cache system accepts all such redirected packets as if they were directly addressed to the cache itself. It responds to the HTTP requestor as if it were the actual intended destination, using a source address that matches the destination address of the original request, assuming the identity of the actual intended content server. If the requested Web object is located in the local cache, it will deliver the object to the requestor immediately. If it is not in the cache, it will set up its own session with the original destination, send it the original request, and feed the response back to the requestor, while also keeping a copy for itself in its cache.

Caching of content works well in the Web world simply because so much Web traffic today is movement of the same Web page to different recipients. It is commonly reported that up to one half of all Web traffic in the Internet is a duplicate transmission of content. If an ISP locally caches all Web content as it is delivered, and checks the cache before passing through a content request, then the ISP's upstream Web traffic volume may be halved. Even a moderately good cache will be able to service about one quarter of the Web content from the cache. That amount of local caching can be translated into a significant cost saving for the ISP.

The cached Web content is traffic that is not purchased as transit traffic from an upstream ISP, representing a potential saving on the cost of upstream transit services. This saving, in turn, can allow the ISP to operate at a lower price point in the retail market. The cache is also located closer to the ISP's customers, and with appropriate tuning, the cache can also deliver cached content to the customer at a consistently much faster rate than a request to the original content server. For very popular Web sites the originating server may be operating more slowly under extreme load, while the local cache continues to operate at a more consistent service level. The combination of the potential for improved performance and lower overall cost is certainly one that looks enticing: the result is the same set of Web transactions delivered to customers, but cheaper and faster.

End-to-End Issues with Cache Middleware

But not everything is perfect in this transparent caching world. What if the Web server used a security model that served content only to certain requestors, and the identity of the requestor was based on their IP address? This is not a very good security model, admittedly, but it's simple, and because of its simplicity this practice enjoys very common usage. With the introduction of a transparent cache, the Web client sees something quite strange. The Web client can ping the Web server, the client can communicate with any other port on the server, and if the client were to query the status of the server, the Web server would be seen to be functioning quite normally. But, mysteriously, the client cannot retrieve any Web content from the server, and the server does not see any such request from the client. The middleware cache is sitting inside a network somewhere on the path between the client and the service, but it may well be the case that neither the end client or the end server are aware of the deployment of the middleware unit. It is not surprising that this is a remarkably challenging operational problem for either the client or the server to correctly diagnose.

A similar case is where a Web server wishes to deliver different content to different requestors, based on some inference gained from the source IP address of the requestor, or the time of day, or some other variable derived from the circumstances of the request. A transparent cache will not detect such variations in the response of the server and will instead deliver the same version of the cached content to all clients whose requests pass through the transparent cache. Variations of this situation of perceived abnormal service behavior abound, all clustered around the same concept that it is unwise in such an environment for a server to assume that it is always communicating with the end client. Indeed the situation is common enough that the Web application has explicit provision for instructing cache servers about whether the content can be cached and replayed in response to similar subsequent requests.

More subtle vulnerabilities also are present in such a middleware environment. A client can confidently assert that packets are being sent to a server, and the server appears to be responding, but the data appears to have been corrupted. Has the server been compromised? It may look like this is the case, but when middleware is around, looks can be deceiving. If the integrity of the cache is compromised, and different pages are substituted in the cache, then to the clients of the cache it appears that the integrity of original server has been compromised. The twist with transparent cache middleware is that the clients of the cache may be unaware that the cache exists, let alone that their requests are being redirected to the cache server. Any abnormalities in the responses they receive are naturally attributed to problems with the security of the server and the integrity of the associated service.

The common theme of these issues is that there are sets of inconsistent assumptions at play here. On the one hand, the assumption of an end-to-end architecture leads an application designer to assume that an IP session opened with a remote peer will indeed be with that remote peer, and not with some intercepting network-level proxy agent attempting to mimic the behavior of that remote peer. On the other hand, is the assumption that transactions adhere to a consistent and predictable protocol, and transactions may be intercepted and manipulated by middleware as long as the resultant interaction behaves according to the defined protocol.

Middleware Architecture

Are transparent caches good or bad? Is the entire concept of middleware good or bad?

There is no doubt that middleware can be very useful. Cache systems can create improved service quality and reduced cost. NATs can reduce the demand for public IP address space. Firewalls can be effective as security policy agents. Middleware can perform load balancing across multiple service points for a particular class of applications, such as a Web server farm. Middleware can dynamically adjust the Internal Protocol parameters of a TCP session to adapt to particular types of networks, or various forms of network service policies. Middleware can provide services within the network that relieve the end user of a set of tasks and responsibilities, and middleware can improve some aspects of the service quality. Middleware can make an Internet service faster, cheaper, more flexible, and more secure, although probably not all at the same time. But middleware comes at a steep long-term price.

The advantage of the Internet lies in its unique approach to network architecture. In a telephone network, the end device—a telephone handset—is a rather basic device consisting of a pair of transducers and a tone generator. All the functionality of the telephone service is embedded within the network itself.

The architecture of the Internet is the complete opposite. The network consists of a collection of packet switches with basic functionality. The service is embedded within the protocol stack and the set of applications that are resident on the connected device. Within this architecture, adding new services to the network is as simple as distributing new applications among those end systems that want to use the application. The network makes no assumptions about the services it supports, and network services can be added, refined, and removed without requiring any change to the network itself. This results in a cheap, flexible, and basic network, and it passes the entire responsibility for service control to the network users. The real strength of the Internet lies in its architectural simplicity and lack of complex interdependencies within the network.

Middleware cuts across this model by inserting directly into the network functionality that alters the behavior of the network. IP or TCP Packet Header fields may be altered on the fly, or, as with a transparent cache, middleware may intercept user traffic, use an application level interpreter to interpret the upper-level service request associated with the traffic, and generate a response, acting as an unauthorized proxy for the intended recipient. With middleware present in an IP network, sending a packet to an addressed destination and receiving a response with a source address of that destination is no guarantee that you have actually communicated with the addressed remote device. You may instead be communicating with a middleware box, or have had the middleware box alter your traffic in various ways that are not directly visible to the sender.

In such an environment, it's not just the end-user applications that define an Internet-deployed service, because middleware is also part of the Internet service architecture. Services may be deployed that are reliant on the existence of middleware to be effective. Streaming video services, for example, become far more viable as a scalable Internet service when the streaming video server content is replicated across a set of middleware streaming systems deployed close to end users of the service. To change the behavior of a service that has supporting middleware deployed requires the network middleware to be changed. A new service may not be deployed until the network middleware is altered to permit its deployment. Any application requiring actual end-to-end communications may have to have additional functionality to detect if there is network middleware deployed along the path, and then explicitly negotiate with this encountered middleware to ensure that its actual communication will not be intercepted and proxied or otherwise altered.

Conclusion

The cumulative outcome is that such a middleware-modified Internet service model is not consistent with an end-to-end architecture. It represents the introduction of a more muddled service architecture where the network may choose to selectively intervene in the interaction between one device and another. Such a network architecture may not have stable scaling properties. Such an architecture may not readily support entire classes of new applications and new services. Such an architecture may not be sufficiently flexible and powerful to underpin a ubiquitous global data communications system. All this middleware overhead makes applications more complex, makes the network more complex, and makes networking more expensive, more limited, and less flexible.

From this perspective, middleware is an unglamorous hack. To adapt a 350-year-old quote from Thomas Hobbes, middleware is nasty, brutish, and short-sighted. It is, hopefully, a temporary imposition on an otherwise elegant, simple, and adequate Internet architecture.[3, 4]

References

[1] Aiken, B. et.al, "Network Policy and Services: A Report of a Workshop on Middleware," RFC 2768, February 2000.

[2] Carpenter, B. ed., "Architectural Principles of the Internet," RFC 1958, June 1996.

[3] Hobbes, Thomas (1588–1679), *Leviathan,* London, 1651. Available from many sources, including, ISBN 0140431950, Penguin Press, 1982.

[4] **http://www.orst.edu/instruct/phl302/texts/hobbes/leviathan-contents.html**

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for the past decade, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. Huston is currently the Chief Scientist in the Internet area for Telstra. He is also a member of the Internet Architecture Board, and is the Secretary of the Internet Society Board of Trustees. He is author of *The ISP Survival Guide,* ISBN 0-471-31499-4, *Internet Performance Survival Guide: QoS Strategies for Multiservice Networks,* ISBN 0471-378089, and coauthor of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks,* ISBN 0-471-24358-2, a collaboration with Paul Ferguson. All three books are published by John Wiley & Sons. E-mail: **gih@telstra.net**

Book Review

Internetworking with TCP/IP

Internetworking with TCP/IP (Vol. 1): Principles, Protocols, and Architectures, Douglas E. Comer, ISBN 0-13-018380-6, Prentice Hall, 2000.

Internetworking With TCP/IP (Vol. 1): Principles, Protocols, and Architectures (fourth edition) is the latest update to Comer's landmark work containing *Internetworking With TCP/IP (Vol. 2): Design, Implementation, and Internals* and *Internetworking With TCP/IP (Vol. 3): Client-Server Programming and Applications/BSD Socket Version.* As a recent engineering graduate, I wish I had read this book sooner; it is very concise and would have saved me a lot of time early in my studies.

Comer imparts Volume 1 in four sections. The first section provides a basic introduction to general networking including descriptions of typical network components. This section is most helpful for the entry-level student or casual reader. Advanced readers may want to skip right to the next section of the text, which continues with coverage of the TCP/IP networking environment from the host's point of view. Here, the organization and operation of local host protocols, addressing, and routing are thoroughly discussed. After reading this portion of the book, you will definitely understand how your desktop computer communicates on the network. Next, the global Internet architecture is laid out in a very comprehensible format. The reader is introduced to router-to-router protocols and algorithms that don't seem so complicated after this treatment. Lastly, application-level services and the client-server model of networking are covered in the final portion of the book.

Classic Reference

When reviewing one of the eminent texts in the field, it is of limited use to comment on the work chapter by chapter. However, I am compelled to comment on the quality of Chapter 11, Protocol Layering. This chapter is particularly interesting because Comer directly compares the ISO 7-layer reference model to the TCP/IP 5-layer model. As is par for this book, the comparison is clear and concise. Furthermore, the advantages and disadvantages of protocol layering are discussed in general and a realistic perspective is provided with reference to actual software implementation practices which may result in layer blurring. This is a very cogent presentation of the interaction between theory and reality in engineering. Although covering a specific topic, it could easily serve as an object lesson in a discussion of "real world" engineering techniques. In addition to Chapter 11, the chapters covering Internet routing (14 through 16) really shine as mainstays of this book. The Internet is viewed from the top down and "big network" protocols such as the *Border Gateway Protocol* (BGP) are given good coverage. This is an area where very few people are completely comfortable and Comer once more brings the important material forward in an easily understandable fashion. In the following paragraphs, I will highlight some of the new material included in the fourth edition.

New TCP/IP Concepts

The book's handling of *Classless Inter-Domain Routing* (CIDR) is very informative. In addition to explaining the inner-workings of the address space, Comer points out the requirement for new routing algorithms. This is an associated cost of adopting this new concept that is often overlooked when CIDR is presented.

Two new and important IP topics are also well-presented. Comer begins his treatment of IP Version 6 (IPv6) with a quick history of the protocol and a review of the logic behind this change. The new address space notation and allocation by type are explained very well. New advantages provided under IPv6 protocol structures are then discussed. Additionally, Mobile IP concepts and practicalities are introduced. Comer does a good job of bringing out both good news and bad news of this crucial new networking technology.

Coverage of *Random Early Drop* (RED) was rather brief and really needs more detail before readers can thoroughly grasp the concept. However, this would require greater mathematical sophistication on the part of the reader. Accordingly, depth of coverage is forgone in the interest of readability.

The section on *Network Address Translation* (NAT) does not adequately explain the dynamic nature of IP address assignment across hosts and data flows. An additional detailed example would help here.

Multimedia

In the application-level services section of the book, Comer offers a hasty explanation of how voice and video are sent over IP internets and how IP Telephony operates. The H.323 protocol is briefly mentioned as the low-bandwidth videoconferencing standard. However, it is not presented in its full importance as an umbrella recommendation from the *International Telecommunications Union* (ITU). A chapter explaining the roles of subordinate H.320 protocols in general would be a welcome addition to this section. *Quality of Service* (QoS) concepts such as *Resource Reservation Protocol* (RSVP), *Differentiated Services* (Diff-Serv), and *Real Time Protocol* (RTP) are likewise given short rift. However, IP Multicast is given significant treatment in one of the book's longest chapters; its concepts, mechanics, and implementation choices are thoroughly addressed.

Security

The book provides clear introductions to *Virtual Private Networks* (VPNs) and the IPSec set of protocols. The actual mechanics of IPSec are detailed thoroughly. Various required algorithms are introduced and pertinent RFC references are pointed out. Finally, firewall basics and implementation issues are covered. Overall, these sections clearly define the pertinent security concepts and make them simple.

Prerequisite Knowledge

This book thoroughly covers the fundamental principles of network design including implementation trade-offs and their associated foibles. However, understanding this text requires little more than a modest understanding of basic computer and networking concepts. An introductory programming course that covers computer organization, the binary number system, and basic data structures should suffice. From this point, the student can use the text for initial network familiarization as well as a future reference to ground the more abstract topics in network design.

A Must-Have Reference

An extensive, concept-based overview of the TCP/IP internetworking protocols makes Comer's Volume 1 the classic introduction to TCP/IP. He makes this an enjoyable read by breaking the topic into short, digestible chapters. Additionally, Comer pauses throughout the text to intersperse review material. Recurrent, italicized summaries provide a significant advantage to the student. These asides concisely summarize key points and provide a coherent set of landmarks for quick review and study.

By itself, Volume 1 is broad enough to be complete as an introduction to IP networking protocols. Comer further extends the work by pointing the reader to very specific resources for in-depth information including web pages and specific RFC numbers for applicable topics at the end of each chapter. One of life's simple treasures is found in the *Guide to RFCs* (Appendix 1). Here, the first 2728 RFCs are organized by major categories and subtopics. At last, a navigable index of RFCs has been incorporated with a superb text from which the beginner can delve the body of networking knowledge.

—*Albert C. Kinney*
`kinney@ieee.org`

Would You Like to Review a Book for IPJ?

We receive numerous books on computer networking from all the major publishers. If you've got a specific book you are interested in reviewing, please contact us and we will make sure a copy is mailed to you. The book is yours to keep if you send us a review. We accept reviews of new titles, as well as some of the "networking classics." Contact us at **`ipj@cisco.com`** for more information.

Fragments

Jonathan B. Postel Service Award for 2001 Presented to Daniel Karrenberg

Internet Society (ISOC) Chairman Brian Carpenter presented the *2001 Jonathan Postel Service Award* to Mr. Daniel Karrenberg, one of the pioneers of the Internet's development in Europe, during the opening ceremony of the 2001 INET Conference. His early work was at the University of Dortmund creating a basic networked e-mail and USENET service. The success of this initiative was the seed on which the first precommercial network, EUnet, was built. As the Internet came to Europe in the late 1980s, Mr. Karrenberg was active in organizing the first RIPE meeting and in creating the RIPE NCC to serve as secretariat for the Internet community in Europe. The RIPE NCC became the first *Regional Internet Registry* as we know them, taking on address allocation as one of its core services. Daniel headed the effort from the start, working hard to maximize the benefit for the community.

Mr. Karrenberg humbly accepted the award, thanking the Internet community for this recognition and pledging to continue his work guided by the spirit of Jon Postel.

The Jonathan B. Postel Service Award was established by the Internet Society to honor a person who has made outstanding contributions in service to the data communications community. It is named for Dr. Jonathan B. Postel to recognize and commemorate the extraordinary stewardship exercised by Jon over the course of a thirty year career in networking. The Award consists of an engraved crystal globe and US $20,000.00. The first award was presented posthumously to Jon Postel himself, accepted by his mother, Lois Postel at INET '99. Scott Bradner received the second award during INET 2000. For additional information on Jon Postel's life and contributions, please visit:
`http://www.isoc.org/postel/`

RFC 1149 Implemented

The Internet Engineering Task Force (IETF) has a long tradition of publishing humorous *Request For Comments* (RFCs) each year on April 1st. One of the more famous such RFCs is "A Standard for the Transmission of IP Datagrams on Avian Carriers," RFC 1149, by David Waitzman, published on April 1, 1990. This "carrier pigeon" RFC was recently implemented by a group in Bergen, Norway. For details see:
`http://www.blug.linux.no/rfc1149/`

Jon Crowcroft Joins IPJ Editorial Advisory Board

We are pleased to announce that Dr. Jon Crowcroft of University College London has joined the Editorial Advisory Board for the *Internet Protocol Journal* (IPJ). Dr. Crowcroft has been working in the field of internetworking and protocol design since the early days of the ARPANET. For more information, see:
`http://www.cs.ucl.ac.uk/staff/J.Crowcroft/`

We would also like to thank Edward Kozel, the creator of IPJ, for his support and advice over the last three years. Mr. Kozel has left Cisco to pursue other interests.

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com

CISCO SYSTEMS

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive, M/S SJ-10/5
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
Cisco Systems, Inc.