# Copperbelt University
# Computer Science Department

# Internet Technologies

By Dr Derrick Ntalasha

**DIT 330: INTERNET TECHNOLOGIES I**

## IP Encapsulation, Fragmentation and Reassembly

How does a host or router send a datagram across a physical network? How do routers handle the problem of sending large datagrams?

When a host or router handles a datagram, IP software first selects the next hop to which the datagram should be sent, N and then transmits the datagram across a physical network to N. However, network hardware does not understand datagram format or Internet addressing. Instead each hardware technology defines a frame format and a physical addressing scheme; the hardware only accepts and delivers packets that adhere to the specified frame format and use the specified hardware addressing scheme. More important, because an internet can contain heterogeneous network technology, the frame format needed to cross a network may differ from the frame format needed to cross the previous network.

### Encapsulation

How can a datagram be transmitted across a physical network that does not understand the datagram format? The answer lies in a technique known as encapsulation. When an IP datagram is encapsulated in a frame, the entire datagram is placed in the data area of a frame. The network hardware treats a frame that contains a datagram exactly like any other frame. Figure 38 shows the concept of encapsulation.

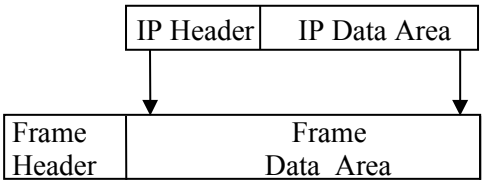


**Figure 1: An IP datagram encapsulated in a hardware frame. In practice, the frame format used with some technologies includes a frame header and frame trailer.**

How does a receiver know whether the data area in an incoming frame contains an IP datagram or other data? The sender and receiver must agree on the value used in the frame type field. When it places a datagram in a frame, the sender assigns the frame type field the special value that is reserved for IP. When a frame arrives with the special value in its type field, the receiver knows that the data area contains an IP datagram.

A frame that carries an IP datagram must have a destination address as usual. Therefore, in addition to placing a datagram in the data area of a frame, encapsulation requires the sender to supply the physical address of the next computer to which the datagram should be sent. To compute the appropriate address, software on the sending computer must perform address binding. The binding translates the IP address of the next hop into an equivalent hardware address, which is used as the destination address in the frame.

**Transmission Across an Internet**

Encapsulation applies to one transmission at a time. After the sender selects a next hop, the sender encapsulates the datagram in a frame and transmits the result across the physical network to the next hop. When the frame reaches the next hop, the receiving software removes the IP datagram and discards the frame (unencapsulation). If the datagram must be forwarded across another network, a new frame is created. If a datagram has to cross several networks on its way to its destination, then it is possible that each network that it crosses uses a different hardware technology than the others, meaning that the frame formats can differ. However, it is worthy noting that during the encapsulation and unencapsulation, the hosts and the routers that handle the datagram store the datagram in memory with no additional header. When the datagram passes across a physical network, the datagram is encapsulated in a frame suitable for that network. The size of the frame header that appears before the datagram depends on the network technology. For example, when crossing an Ethernet network, the frame encapsulating the datagram will have an Ethernet header. Similarly, if the network is an FDDI ring, the frame encapsulating the datagram will have an FDDI header.

Because of the encapsulation and unencapsulation on its journey to its destination, frame headers do not accumulate during the trip. Therefore, when the datagram reaches its final destination, the frame that carried the datagram is discarded and the datagram appears exactly the same size as when it was originally sent.

**MTU, Datagram Size, and Encapsulation**

Each hardware technology specifies the maximum amount of data that a frame can carry. The limit is known as a *maximum transmission unit (MTU)*. There is no exception to the MTU limit - the network hardware is not designed to accept or transfer frames that carry more data than the MTU allows. Thus a datagram must be smaller or equal to the network MTU or it can not be encapsulated for transmission.

In an internet that contains heterogeneous networks, MTU restrictions can cause a problem. In particular, because a router can connect networks with different MTU values, the router can receive a datagram over one network that can not be sent over another. An IP router uses a technique known as *fragmentation* to solve the problem of heterogeneous MTUs. When a datagram is larger than the MTU of the network over which it must be sent, the router divides the datagram into smaller pieces called *fragments*, and sends each fragment independently.

A fragment has the same format as other datagrams - a bit in the FLAGS field of the header indicates whether a datagram is a fragment or a complete datagram. Other fields in the header are assigned information that can be used to reassemble the fragments to reproduce the original datagram. In particular, the FRAGMENT OFFSET field in the header of a fragment specifies where in the original datagram the fragment belongs.

To fragment a datagram for transmission across a network, a router uses the network MTU and the datagram header size to calculate the maximum amount of data that can be sent in each fragment and the number of fragments that will be needed. The router then creates the fragments. It begins by starting each fragment with a copy of the original header, and then modifies individual header fields. For example, the router sets the appropriate bit in the FLAGS field to indicate that the datagram is a fragment. Finally, the router copies the appropriate data from the original datagram into the fragment, and transmits the result.

**Reassembly**

The process of creating a copy of the original from fragments is called reassembly. Because each fragment begins with a copy of the original datagram header, all fragments have the same destination address as the original datagram from which they were derived. Furthermore, the fragment that carries the final piece of data has an additional bit set in the header. Thus, a receiver performing reassembly can tell whether all fragments have arrived successfully.

The Internet Protocol specifies that the ultimate destination host should reassemble fragments. Requiring the ultimate destination to reassemble fragments has two main advantages:

- It reduces the amount of state information in routers. When forwarding a datagram, a router does not need to know whether the datagram is a fragment.
- It allows routers to change dynamically. If an intermediate router reassembles fragments, all fragments would need to reach that router. By postponing reassembly until the ultimate destination, IP is free to pass some fragments from a datagram along a different route than other fragments.

**Identifying a Datagram**

IP does not guarantee delivery. Therefore, individual fragments can be lost or arrive out of order. More important, if a source sends multiple datagrams to a given destination, the fragments from the datagrams can arrive in arbitrary order. How does IP software reassemble fragments that arrive out of order? A sender places a unique identification number in the INDENTIFICATION field of each outgoing datagram. When a router fragments the datagram, the router copies the identification number into each fragment. A receiver uses the identification number and IP source address in the incoming fragment to determine the datagram to which the fragment belongs. In addition, the FRAGMENT OFFSET field tells a receiver how to order the fragments within a given datagram.

**Fragment Loss**

If an underlying network drops packets, an encapsulated datagram or fragment can be lost. When all fragments from a datagram arrive, the datagram can be reassembled. However, a problem arises when one or more fragments from a datagram arrive, and some fragments are delayed or lost. Although the datagram can not be reassembled, the receiver must save the fragments in case missing fragments are only delayed.

A receiver can not hold fragments an arbitrarily long time because fragments occupy space in the receiver's memory. To avoid exhausting memory, IP specifies a maximum time to hold fragments. When the first fragment arrives from a given datagram, the receiver starts a timer. If all fragments of a datagram arrive before the timer expires, the receiver cancels the timer and reassembles the datagram. However, if the timer expires before all fragments arrive, the receiver discards those datagrams that have arrived.

The result of IP's reassembly timer is all-or-nothing: either all fragments arrive and IP reassembles the datagram, or IP discards the entire datagram. In particular, there is no mechanism for a receiver to tell the sender which fragments have arrived. The design makes sense because the sender does not know about fragmentation. Furthermore, if the sender did transmit the datagram, routes may be different, which means a retransmission would not necessarily traverse the same routers. Hence, there is no guarantee that a retransmitted datagram would be fragmented in the same way as the original.

**Fragmenting a Fragment**

After performing fragmentation, a router forwards each fragment on to its destination. What happens if a fragment eventually reaches another network that has a smaller MTU? The fragmentation scheme has been planned carefully to make it possible to further fragment a fragment. Another router along the path divides the fragment into smaller fragments. In a poorly designed internet where networks are arranged in a sequence of decreasing MTUs, each router along the path must further fragment each fragment.

IP does not distinguish between original fragments and subfragments. In particular, a receiver can not know whether an incoming fragment was the result of one router fragmenting a datagram or multiple routers fragmenting fragments. The advantage of making all fragments the same is that a receiver can perform reassembly of the original datagram without first reassembling subfragments. Doing so saves CPU time, and reduces the amount of information needed in the headers of each fragment.

# Copperbelt University
# Computer Science Department

By Dr Derrick Ntalasha

**CS 460: INTERNET TECHNOLOGIES**

## The Future IP (IPv6)

So far we have discussed the current version of the Internet Protocol and have seen that an IP datagram consists of a header that is followed by data. The header contains information such as a destination address that IP software uses to deliver the datagram to its destination. Each header field has a fixed size to make processing efficient. We now know that an IP datagram is encapsulated in a network frame as it travels across a physical network. The question is: What is the future of the Internet Protocol? To answer this question, we begin by assessing the strengths and limitations of the current version of IP (IPv4), and then consider an entirely new version of IP that the Internet Engineering Task Force (IETF) has proposed to replace the current version.

**The success of IP (IPv4)**

The current version of IP has been extremely successful. It has made it possible for the Internet to handle heterogeneous networks, dramatic changes in hardware technology, and extreme increases in scale. To handle heterogeneity, IP defines a uniform packet format (the IP datagram) and a packet transfer mechanism. IP datagrams are the fundamental unit of communication in the Internet - when an application transfers data across the Internet from one computer to another, the data travels in an IP datagram. IP also defines a set of addresses that allow applications and higher layer protocols to communicate across heterogeneous networks without knowing the differences in hardware addresses used by the underlying network systems. The demonstration of scalability is evident because the current Internet includes million of users around the world.

The current version of IP (IPv4) has also accommodated changes in hardware technology. Although the protocol was defined before local area network technologies became popular, the original design has continued to work well through several generations of hardware technologies. IP is now used over networks that operate several orders of magnitude faster than networks that were in use when IP was designed. Furthermore, some modern networks offer frame sizes that are much larger than the frame sizes that were available when IP was defined. More important, IP works efficiently over such networks because it can take advantage of the increased frame size.

**The motivation for change**

Question: If IP works so well, why change? The primary motivation for change arises from the limited address space. When IP was defined, only a few computer networks existed. The designers decided to use 32 bits for an IP address because doing so allowed the Internet to include over a million networks. However, the global Internet is growing exponentially, with the size doubling in less than a year. At the current growth rate, each of the possible network prefixes will soon be assigned, and no further growth will be possible. Thus, the primary motivation for defining a new version of IP arose from the address space limitation - larger addresses are necessary to accommodate continued growth of the Internet.

Secondary motivations for changes in IP have arisen from new Internet applications. For example, applications that deliver audio and video need to deliver data at regular intervals. To keep such information flowing through the Internet without disruption, IP must avoid changing routes frequently. Although the current IP datagram header includes a field that can be used to request a type of service, the protocol did not define a type of service that can be used for real-time delivery of audio and video.

The area of networked entertainment requires high performance because the service has to be delivered in real-time as we have seen. Actually, the possibility is that every television set will become an Internet host in future. As the world of digital high definition approaches, the differences between a computer and a television set will diminish.

New applications are being developed that require more complex addressing and routing capabilities. For example, interest has increased in *collaboration technologies* that provide communication among a group of colleagues analogous to a telephone conference call. To make collaboration effective, an internet needs a mechanism that allows groups to be created or changed, and provides a way to send a copy of a packet to each participant in a given group. In addition to groups that must each receive a copy of a packet, some applications use groups to handle load sharing. That is, several identical copies of a service exist, and a packet sent to the group is routed to the copy of the service that is closest to the sender. Thus a new version of IP needs to include mechanisms that make such addressing and routing possible.

**Characterisation of Features in IPv6**

IPv6 retains many of the design features that have made the current IP protocol (IPv4) so successful. Like IPv4, IPv6 is connectionless each datagram contains a destination address, and each datagram is routed independently. Like IPv4, the header in a datagram contains a maximum number of hops the datagram can take before being discarded. More important, IPv6 retains most of the general facilities provided by the IPv4 options.

Despite retaining the basic concepts from the current version, IP version 6 changes all the details. For example, IPv6 uses large addresses and an entirely new datagram header format. Finally, IPv6 uses a series of fixed-length headers to handle optional information instead of a single header with a variable-length options field. The new features in IPv6 can roughly be grouped into six main categories:

1. *Address size*. Instead of 32 bits, each IPv6 address contains 128 bits. The resulting address space is large enough to accommodate continued growth of the world-wide Internet for many decades.

2. *Header format*. The IPv6 datagram header is completely different from the IPv4 header. Almost every field in the header has been changed; some have been replaced.

3. *Extension headers*. Unlike IPv4, which uses a single header format for all datagrams, IPv6 encodes information into separate headers. A datagram consists of the base IPv6 header followed by zero or more extension headers, followed by data.

4. *Support for audio and video*. IPv6 includes a mechanism that allows a sender and receiver to establish a high-quality path through the underlying network and to associate datagrams with that path. Although the mechanism is intended for use with audio and video applications that require high performance guarantees, the mechanism can also be used to associate datagrams with low-cost paths.

5. *Extensible protocol*. Unlike IPv4, IPv6 does not specify all possible protocol features. Instead, the designers have provided a scheme that allows a sender to provide additional information to a

datagram. The extension scheme makes IPv6 more flexible than IPv4, and means that new features can be added to the designs as needed.

6. *Internet Security*. The current Internet Protocol (IPv4) has a number of security problems and lacks effective privacy and authentication mechanisms below the application layer. IPv6 remedies these shortcomings by providing in-built security features into the protocol itself.

**IPv6 datagram format**

As figure 39 shows, an IPv6 datagram begins with a *base header*, which is followed by zero or more extension headers, followed by data.

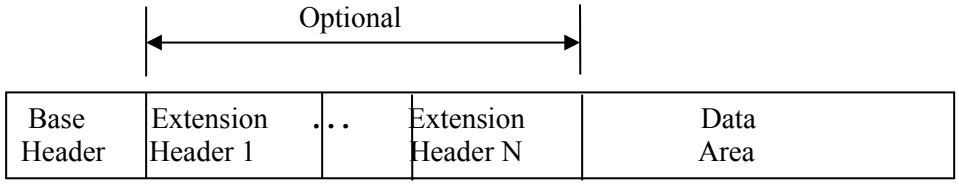


**Figure 2: The general format of an IPv6 datagram. Extension headers are optional - the minimum datagram has a base header followed by data.**

Although it illustrates the general datagram structure, fields in the figure are not drawn to scale. In particular, some extension headers can be larger than the base header, while others can be smaller. Furthermore, in many datagrams, the size of the data area is much larger than the size of the headers.

**IPv6 Base header format**

Although it is twice as large as an IPv4 header, the IPv6 base header contains less information. Figure 40 illustrates the format. As the figure shows, most of the space in the header is devoted to two fields that identify the sender and the recipient. As in IPv4, the SOURCE ADDRESS field identifies the sender, and the DESTINATION ADDRESS field identifies the intended recipient. Each address occupies sixteen octets, four times more than an IPv4 address.

In addition to the source and destination address, the base header contains six fields. The VERS field identifies the protocol as version 6. The PRIORITY field specifies the routing priority class. This means that the 4-bit Priority field in the IPv6 header enables a source to identify the desired delivery priority of its packets, relative to other packets from the same source. The Priority values are divided into two ranges: Values 0 through 7 are used to specify the priority of traffic for which the source is providing congestion control, i.e., traffic that "backs off" in response to congestion, such as TCP traffic. Values 8 through 15 are used to specify the priority of traffic that does not back off in response to congestion, e.g., "real-time" packets being sent at a constant rate. The PAYLOAD LENGTH field corresponds to IPv4's datagram length field. Unlike IPv4, the PAYLOAD LENGTH specifies only the size of the data being carried (i.e., the payload); the size of the header is excluded. The HOP LIMIT corresponds to the IPv4 TIME-TO-LIVE field. IPv6 interprets the HOP LIMIT strictly - the datagram will be discarded if the HOP LIMIT counts down to zero before the datagram arrives at its destination.

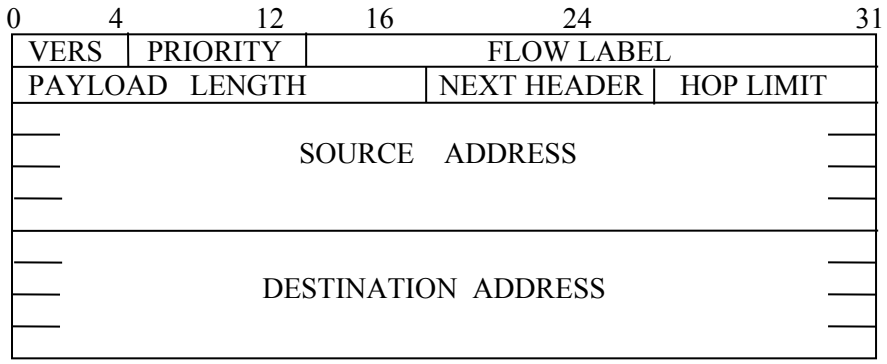


**Figure 3: The format of an IPv6 base header.**

The field FLOW LABEL is intended for use with new applications that require performance guarantees. The label can be used to associate a datagram with a particular underlying network path. The label is divided into two parts - one used to specify a *traffic class*, and the other used to define a specific path. The traffic class specifies general characteristics that the datagram needs. For example, to send interactive traffic (e.g., keystrokes and mouse movements), one might specify a general traffic class for low delay. To send real-time audio across an internet, however, a sender might request the underlying network hardware to establish a path that has a delay which is less than 100 milliseconds. When the path is established, the network system returns an identifier that the sender places in each datagram to be sent along the path. Routers use the value in the FLOW LABEL to route the datagram along the prearranged path.

The FLOW LABEL and the PRIORITY fields in the IPv6 header may be used together by a host to identify those packets for which it requests special handling by IPv6 routers, such as non-default quality of service or "real-time" service. This capability is important in order to support applications, which require some degree of consistent throughput, delay, and/or jitter. Such applications are commonly described as "multi- media" or "real-time" applications.

Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet. A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers. The nature of that special handling might be conveyed to the routers by a control protocol, such as a resource reservation protocol, or by information within the flow's packets themselves, e.g., in a hop-by-hop option. There may be multiple active flows from a source to a destination, as well as traffic that is not associated with any flow. A flow is uniquely identified by the combination of a source address and a non-zero flow label. Packets that do not belong to a flow carry a flow label of zero.

The NEXT HEADER field is used to specify the type of information that follows the current header. For example, if the datagram includes an extension header, the NEXT HEADER field specifies the type of extension header. If no extension header exists, the NEXT HEADER field specifies the type of data being carried in the datagram, as shown by the concept illustrated in figure 41.

(a)


(b)

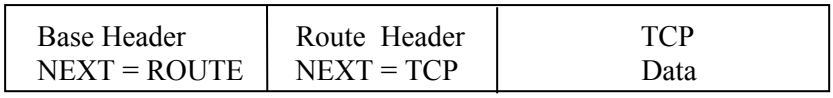


**Figure 4: Two IPv6 datagrams in which (a) contains a base header plus data, and (b) contains a base header, route header and data. The NEXT HEADER field in each header specifies the type of the item that follows.**

### How IPv6 handles multiple headers

Because the standard specifies a unique value for each possible header type, there is never ambiguity about the interpretation of the NEXT HEADER fields. A receiver uses the NEXT HEADER field in each header to determine what follows. If the value in the field corresponds to a type used for data, the receiver passes the datagram to a software module that handles the data. If the value in the NEXT HEADER field corresponds to another header, IP software parses the header and interprets its contents. Once it finishes with a header, IP uses the NEXT HEADER field to determine whether data or another header follows.

How does IP software know where a particular header ends and the next item begins? Some header types have a fixed size. For example, a base header has a fixed size of exactly forty octets. To move to the item following a base header, IPv6 software simply adds 40 to the address of the base header.

Some extension headers do not have a fixed size. In such cases, the header must contain sufficient information to allow IPv6 to determine where the header ends. For example, figure 42 illustrates the general form of an IPv6 options header that carries information similar to the options in an IPv4 datagram.

The options extension header illustrates one way IPv6 handles headers that do not have fixed size. When composing a datagram, the sender stores the length of the options header in field HEADER LEN. When a receiver encounters an options extension header, it uses the HEADER LEN field to determine the location of the next item, and the NEXT HEADER field to determine the type.

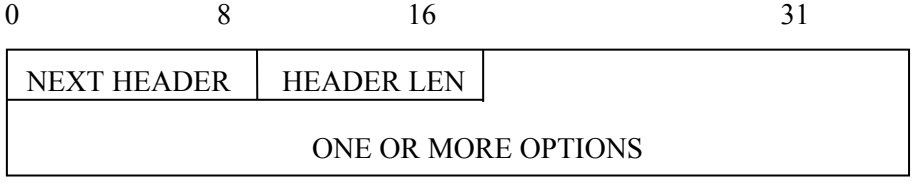


**Figure 5: The IPv6 options extension header. Because the size of the options header can vary from one datagram to another, the HEADER LEN field specifies the exact length.**

**Fragmentation, Reassembly and path MTU**

Although IPv6 fragmentation resembles IPv4 fragmentation, the details differ. Like IPv4, a prefix of the original datagram is copied into each fragment, and the payload length is modified to be the length of the fragment. Unlike IPv4, however, IPv6 does not include fields for fragmentation information in the base header. Instead, IPv6 places them in a separate fragment extension header. The presence of a fragment extension header identifies the datagram as a fragment. Figure 43 illustrates IPv6 fragmentation.

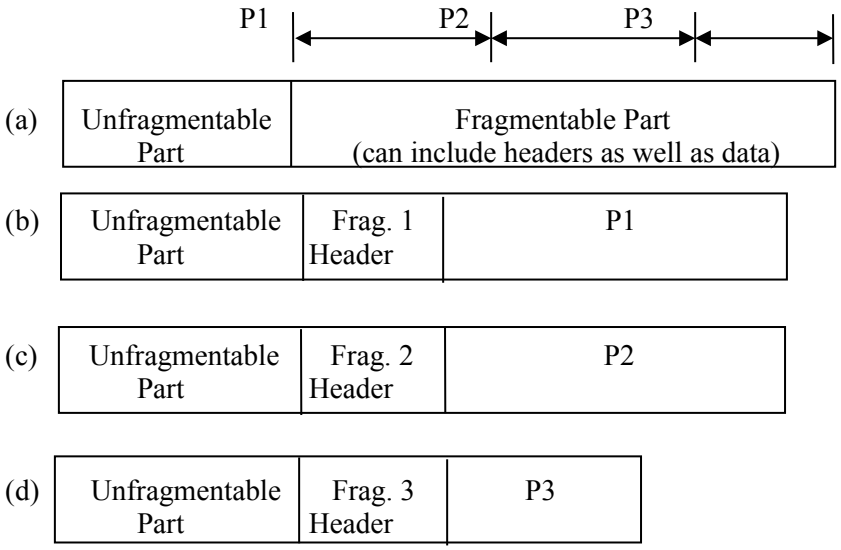


**Figure 6: Illustration of fragmentation in IPv6. The fragmentable part of the original datagram (a), is placed in the payload area of fragments (b, c, and d). Each fragment begins with a copy of the unfragmentable part and a fragment extension header.**

As figure 43 illustrates, each fragment is smaller than the original datagram. As with IPv4, the fragment size is chosen to be the maximum transmission unit (MTU) size of the underlying network over which the fragment must be sent. Thus, the final fragment may be smaller than the others because it represents the amount remaining after MTU-size pieces have been extracted from the original datagram.

Fragmentation in IPv6 differs dramatically from fragmentation in IPv4. In IPv4, a router performs fragmentation when the router receives a datagram that is too large for the network over which the datagram must be sent. In IPv6, a sending host is responsible for fragmentation. That is, hosts are expected to choose a datagram size that will not require fragmentation. Routers along the path that receive a large datagram will not fragment the datagram.

How can a host choose a datagram size that will not result in fragmentation? The host must learn the MTU of each network along the path to the destination, and must choose a datagram size to fit the smallest. The maximum MTU along a path from a source to a destination is known as the *path MTU*, and the process of learning the path MTU is known as *path MTU discovery*. In general, path MTU discovery is an iterative procedure. A host sends a sequence of various-size datagrams to the destination to see if they arrive without error. Once a datagram is small enough to pass through without fragmentation, the host chooses a datagram size equal to the path MTU.

**The purpose of multiple headers**

Why does IPv6 use separate extension headers? There are two reasons: economy and extensibility. Partitioning the datagram functionality into separate headers is economical because it saves space. To understand why, one must realise that although the IPv6 protocol includes many facilities, designers expect a given datagram to use only a subset. Having separate headers in IPv6 makes it possible to define a large set of features without requiring each datagram header to have at least one field for each feature. For example, although many IPv4 datagrams are not fragmented, the IPv4 header has fields used to hold fragmentation information. In contrast, IPv6 does not waste space on fields for fragmentation unless the datagram is fragmented. Because most datagrams only need a few headers, avoiding unnecessary header fields can save considerable space. Smaller datagrams also take less time to transmit. Thus, reducing datagram size also reduces the bandwidth consumed.

To understand extensibility, consider adding a new feature to a protocol. A protocol like IPv4 that uses a fixed header format requires a complete change - the header must be redesigned to accommodate fields needed to support the new feature. In IPv6, however, existing protocol headers can remain unchanged. A new NEXT HEADER type is defined as well as a new header format.

The chief advantage of placing new functionality in a new header lies in the ability to experiment with a new feature before changing all computers in the Internet. For example, suppose the owners of two computers wish to test a new datagram encryption technique. The two must agree on the details of an experimental encryption header. The sender adds the new header to a datagram, and the receiver interprets the header in the incoming datagrams. As long as the new header appears after the headers used for routing, routers in the internet between the sender and receiver can pass the datagram without understanding the experimental header. If an experimental header is incorrectly placed before routing headers, a router will discard the datagram. Once an experimental feature proves useful, it can be incorporated into the standard.

**IPv6 addressing**

Like IPv4, IPv6 assigns a unique address for each connection between a computer and a physical network. Thus, if a computer (e.g., a router) connects to three physical networks, the computer is assigned three addresses. Also like IPv4, IPv6 separates each such address into a prefix that identifies the network and a suffix that identifies a particular computer on the network.

Despite adopting the same approach for assigning computer addresses, IPv6 addressing differs from IPv4 addressing in significant ways.

- First, all address details are completely different. In particular, addresses do not have defined classes. Instead, the boundary between the prefix and suffix can fall anywhere within the address and can not be determined from the address alone. Thus, a *prefix length* must be associated with each address (e.g., in a routing table) to enable software to know where the prefix ends.

- Second, IPv6 defines a set of special addresses that differ dramatically from IPv4 special addresses. In particular, IPv6 does not include a special address for broadcasting on a given network. Instead, each IPv6 address is one of three basic types:

1. *Unicast.* The address corresponds to a single computer. A datagram sent to the address is routed along a shortest path to the computer.

2. *Multicast.* The address corresponds to a set of computers, possibly at many locations; membership in the set can change at any time. When a datagram is sent to the address, IPv6 delivers one copy of the datagram to each member of the set.

3. *Anycast.* The address corresponds to a set of computers that share a common address prefix (e.g., all reside in a single location). A datagram sent to the address is routed along a shortest path and then delivered to exactly one of the computers (e.g., the computer closest to the sender).

Anycast addressing was originally known as *cluster addressing*. The motivation for such addressing arises from a desire to allow replication of services. For example, a corporation that offers a service over the network assigns an anycast address to several computers that all provide the service. When a user sends a datagram to the anycast address, IPv6 routes the datagram to one of the computers in the set (i.e., in the cluster). If a user from another location sends a datagram to the anycast address, IPv6 can choose to route the datagram to a different member of the set, allowing both computers to process requests at the same time.

The use of anycast addresses in the IPv6 source route can allow nodes to control the path, which their traffic flows. An IPv6 anycast address is an address that is assigned to more than one interfaces (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the "nearest" interface having that address, according to the routing protocols' measure of distance.

Anycast addresses, when used as part of a route sequence, permits a node to select which of several internet service providers it wants to carry its traffic, for example. This capability is sometimes called "source selected policies". This would be implemented by configuring anycast addresses to identify the set of routers belonging to internet service providers (e.g., one anycast address per internet service provider). These anycast addresses can be used as intermediate addresses in an IPv6 routing header, to cause a packet to be delivered via a particular provider or sequence of providers. Other possible uses of anycast addresses are to identify the set of routers attached to a particular subnet, or the set of routers providing entry into a particular routing domain.

Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address.

Other addresses as used in IPv4 are still found in IPv6. These include:

### *The Unspecified Address (This computer address in IPv4)*

The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any computer. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 datagrams sent by an initializing host before it has learned its own address. The unspecified address must not be used as the destination address of IPv6 datagrams or in IPv6 Routing Headers.

### *The Loopback Address*

The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a computer to send an IPv6 datagram to itself. It may never be assigned to any interface. The loopback address must not be

used as the source address in IPv6 datagrams that are sent outside of a single computer. An IPv6 datagram with a destination address of loopback must never be sent outside of a single computer.

**IPv6 colon hexadecimal notation**

Although an address that occupies 128 bits can accommodate Internet growth, writing such numbers can be unwieldy. For example, consider a 128-bit number written in dotted decimal notation:

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

To help reduce the number of characters used to write an address, the designers of IPv6 propose using a more compact syntactic form known as *colon hexadecimal notation* (*colon hex*) in which each group of 16 bits is written in hexadecimal with a colon separating groups. For example, when the above number is written in colon hex, it becomes:

69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF

As the example illustrates, colon hex notation requires fewer characters to express an address. An additional optimisation known as zero compression further reduces the size. Zero compression replaces sequences of zeroes with two colons. For example, the address:

FF0C:0:0:0:0:0:0:B124 can be written as FF0C::B124

The large IPv6 address space and the proposed address allocation scheme make zero compression especially important because the designers expect many IPv6 addresses to contain strings of zeroes. In particular, to help ease the transition to the new protocol, the designers mapped existing IPv4 addresses into the IPv6 address space. Any IPv6 address that begins with 96 zero bits contains an IPv4 address in the low-order 32-bits.

**Routing in IPv6**

Routing in IPv6 is almost identical to IPv4 routing under CIDR except that the addresses are 128- bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms (OSPF, RIP, IDRP, ISIS, etc.) can be used to route IPv6. IPv6 also includes simple routing extensions which support powerful new routing functionality. These capabilities include:

- Provider Selection (based on policy, performance, cost, etc.)
- Host Mobility (route to current location)
- Auto-Readdressing (route to new address)

The new routing functionality is obtained by creating sequences of IPv6 addresses using the IPv6 Routing option. The routing option is used by an IPv6 source to list one or more intermediate nodes (or topological group) to be "visited" on the way to a packet's destination. This function is very similar in function to IPv4's Loose Source and Record Route option.

In order to make address sequences a general function, IPv6 hosts are required in most cases to reverse routes in a packet it receives (if the packet was successfully authenticated using the IPv6 Authentication Header) containing address sequences in order to return the packet to its originator. This approach is taken to make IPv6 host implementations from the start support the handling and reversal of source

routes. This is the key for allowing them to work with hosts which implement the new features such as provider selection or extended addresses.

Three examples show how the address sequences can be used. In these examples, address sequences are shown by a list of individual addresses separated by commas. For example:

SRC, I1, I2, I3, DST

Where the first address is the source address, the last address is the destination address, and the middle addresses are intermediate addresses.

For these examples assume that two hosts, H1 and H2 wish to communicate. Assume that H1 and H2's sites are both connected to providers P1 and P2. A third wireless provider, PR, is connected to both providers P1 and P2 as shown in figure 44.

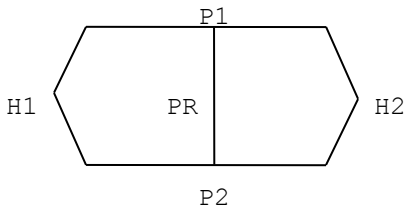


**Figure 7: Example of Routing in IPv6**

The simplest case (no use of address sequences) is when H1 wants to send a packet to H2 containing the addresses: H1, H2

When H2 replied it would reverse the addresses and construct a packet containing the addresses:
H2, H1

In this example either provider could be used, and H1 and H2 would not be able to select which provider traffic would be sent to and received from. If H1 decides that it wants to enforce a policy that all communication to/from H2 can only use provider P1, it would construct a packet containing the address sequence: H1, P1, H2

This ensures that when H2 replies to H1, it will reverse the route and the reply would also travel over P1. The addresses in H2's reply would look like: H2, P1, H1

If H1 became mobile and moved to provider PR, it could maintain (not breaking any transport connections) communication with H2, by sending packets that contain the address sequence: H1, PR, P1, H2

This would ensure that when H2 replied it would enforce H1's policy of exclusive use of provider P1 and it would send the packet to H1 and then to the new location on provider PR. The reversed address sequence would be: H2, P1, PR, H1

The address sequence facility of IPv6 can be used for provider selection, mobility, and readdressing. It is a simple but powerful capability.

## IPv6's Integrated Security Features

The current Internet has a number of security problems and lacks effective privacy and authentication mechanisms below the application layer. IPv6 remedies these shortcomings by having two integrated options that provide security services. These two options may be used singly or together to provide differing levels of security to different users. This is very important because different user communities have different security needs.

The first mechanism, called the "IPv6 Authentication Header", is an extension header, which provides authentication and integrity (without confidentiality) to IPv6 datagrams. The Authentication Header (AH) is a mechanism for providing integrity and authentication for IP datagrams. The IP Authentication Header (AH) provides security by adding authentication information to an IP datagram. The authentication information is calculated using all the fields in the IP datagram (including not only the IP header but also other headers and user data), which do not change in transit. Fields or options, which need change in transit (e.g. hop count or time to live) are considered to be zero for the calculation of the authentication data. The encryption keys of the users involved in the data transmission process are also used in the calculation. The calculation produces a unique digital signature, which can also be used by the Authentication Header to provide non-repudiation. The Authentication Header (AH) also provides data integrity because any changes made that are made to the payload during transmission are detected by the packet's hashed signature in the header. However, AH does not provide confidentiality because it does not encrypt the packet's payload.

The Authentication Header can also be used to eliminate a significant class of network attacks, including host masquerading attacks. The use of the IPv6 Authentication Header is particularly important when source routing is used with IPv6 because of the known risks in IP source routing. The placement of the Authentication Header at the internet layer can help provide host origin authentication to those upper layer protocols and services that currently lack meaningful protections. The second security extension header provided with IPv6 is the "IPv6 Encapsulating Security Payload (ESP) Header". This mechanism provides integrity and confidentiality to IPv6 datagrams by using encryption. The IP Authentication Header (AH) may be used in conjunction with Encapsulating Security Payload (ESP) to provide authentication. When desiring integrity and authentication without confidentiality the IP Authentication Header (AH) can be used instead of Encapsulating Security Payload (ESP).

## IPv6 Transition Mechanisms

The IPv6 transition mechanisms are a set of protocol mechanisms implemented in hosts and routers, along with some operational guidelines for addressing and deployment, designed to make the transition of the Internet to IPv6 work with as little disruption as possible.

The challenge for IPv6 is for its transition to be complete before IPv4 routing and addressing break. The transition will be much easier if IPv4 addresses are still globally unique. The two transition requirements, which are the most important are flexibility of deployment and the ability for IPv4 hosts to communicate with IPv6 hosts. There will be IPv6-only hosts, just as there will be IPv4-only hosts. The capability must exist for IPv6-only hosts to communicate with IPv4-only hosts globally while IPv4 addresses are globally

unique. It is important to realise that the deployment strategy for an IPv6 must be as flexible as possible because the Internet is too large for any kind of controlled roll out to be successful.

The same requirement is also true for IPv6. The Internet has a large installed base. Features need to be designed into an IPv6 to make the transition as easy as possible. As with processors and operating systems, IPv6 must be backwards compatible with IPv4. New features alone are not adequate to motivate users to deploy new protocols. Therefore, IPv6 must have a great transition strategy as well as new features.

The key transition objective is to allow IPv6 and IPv4 hosts to interoperate. A second objective is to allow IPv6 hosts and routers to be deployed in the Internet in a highly diffuse and incremental fashion, with few interdependencies. A third objective is that the transition should be as easy as possible for end-users, system administrators, and network operators to understand and carry out.

The IPv6 transition mechanisms provide a number of features, including:

- Incremental upgrade and deployment. Individual IPv4 hosts and routers may be upgraded to IPv6 one at a time without requiring any other hosts or routers to be upgraded at the same time. New IPv6 hosts and routers can be installed one by one.
- Minimal upgrade dependencies. The only prerequisite to upgrading hosts to IPv6 is that the DNS server must first be upgraded to handle IPv6 address records. There are no pre-requisites to upgrading routers.
- Easy Addressing. When existing installed IPv4 hosts or routers are upgraded to IPv6, they may continue to use their existing address. They do not need to be assigned new addresses. Administrators do not need to draft new addressing plans.
- Low start-up costs. Little or no preparation work is needed in order to upgrade existing IPv4 systems to IPv6, or to deploy new IPv6 systems. The mechanisms employed by the IPv6 transition mechanisms include:
    1. An IPv6 addressing structure that embeds IPv4 addresses within IPv6 addresses, and encodes other information used by the transition mechanisms.
    2. A model of deployment where all hosts and routers upgraded to IPv6 in the early transition phase are "dual" capable (i.e. implement complete IPv4 and IPv6 protocol stacks).
    3. The technique of encapsulating IPv6 packets within IPv4 datagram to carry them over segments of the end-to-end path where the routers have not yet been upgraded to IPv6.

The IPv6 transition mechanisms ensures that IPv6 hosts can interoperate with IPv4 hosts anywhere in the Internet up until the time when IPv4 addresses run out, and allows IPv6 and IPv4 hosts within a limited scope to interoperate indefinitely after that. This feature protects the huge investment users have made in IPv4 and ensures that IPv6 does not render IPv4 obsolete.

The incremental upgrade features of the IPv6 transition mechanisms allow the host and router vendors to integrate IPv6 into their product lines at their own pace, and allows the end users and network operators to deploy IPv6 on their own schedules.

To recap, we can say that IPv6 is a new version of IP, which is designed to be an evolutionary step from IPv4. It is a natural increment to IPv4. It can be installed as a normal software upgrade in internet devices and is interoperable with the current IPv4. Its deployment strategy was designed not to have any "flag" days. IPv6 is designed to run well on high performance networks (e.g., ATM) and at the same time is still

efficient for low bandwidth networks (e.g., wireless). In addition, it provides a platform for new internet functionality.

In addition to the obvious requirement of an internet protocol which can support large scale routing and addressing, IPv6 is a protocol which imposes a low overhead and supports auto configuration and mobility as a basic element. The nature of nomadic computing requires an internet protocol to have built in authentication and confidentiality. It also goes without saying that these devices will need to communicate with the current generation of computers. The requirement for low overhead comes from the wireless media. Unlike LANs which are very high speed, the wireless media are several orders of magnitude slower due to such constraints as available frequencies and error rates.