Name: Syed Asad Abrar

Roll No: L164292

Section: CS-B

**CN Assignment 3**

Q1. IPv6 fixed header is 40 bytes long and contains the following information.

|  |  |
| --- | --- |
| S.N. | Field & Description |
| 1 | Version (4-bits): It represents the version of Internet Protocol, i.e. 0110. |
| 2 | Traffic Class (8-bits): These 8 bits are divided into two parts. The most significant 6 bits are used for Type of Service to let the Router Known what services should be provided to this packet. The least significant 2 bits are used for Explicit Congestion Notification (ECN). |
| 3 | Flow Label (20-bits): This label is used to maintain the sequential flow of the packets belonging to a communication. The source labels the sequence to help the router identify that a particular packet belongs to a specific flow of information. This field helps avoid re-ordering of data packets. It is designed for streaming/real-time media. |
| 4 | Payload Length (16-bits): This field is used to tell the routers how much information a particular packet contains in its payload. Payload is composed of Extension Headers and Upper Layer data. With 16 bits, up to 65535 bytes can be indicated; but if the Extension Headers contain Hop-by-Hop Extension Header, then the payload may exceed 65535 bytes and this field is set to 0. |
| 5 | Next Header (8-bits): This field is used to indicate either the type of Extension Header, or if the Extension Header is not present then it indicates the Upper Layer PDU. The values for the type of Upper Layer PDU are same as IPv4’s. |
| 6 | Hop Limit (8-bits): This field is used to stop packet to loop in the network infinitely. This is same as TTL in IPv4. The value of Hop Limit field is decremented by 1 as it passes a link (router/hop). When the field reaches 0 the packet is discarded. |
| 7 | Source Address (128-bits): This field indicates the address of originator of the packet. |
| 8 | Destination Address (128-bits): This field provides the address of intended recipient of the packet. |

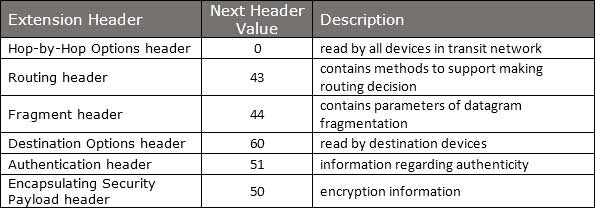
## Extension Headers

In IPv6, the Fixed Header contains only that much information which is necessary, avoiding those information which is either not required or is rarely used. All such information is put between the Fixed Header and the Upper layer header in the form of Extension Headers. Each Extension Header is identified by a distinct value.

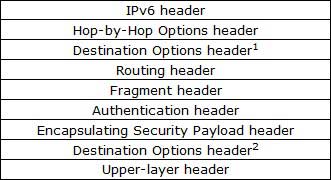
When Extension Headers are used, IPv6 Fixed Header’s Next Header field points to the first Extension Header. If there is one more Extension Header, then the first Extension Header’s ‘Next-Header’ field points to the second one, and so on. The last Extension Header’s ‘Next-Header’ field points to the Upper Layer Header. Thus, all the headers points to the next one in a linked list manner.

If the Next Header field contains the value 59, it indicates that there are no headers after this header, not even Upper Layer Header.

The following Extension Headers must be supported as per RFC 2460:



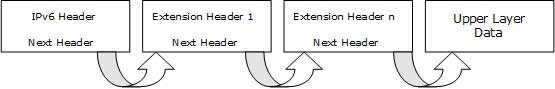
The sequence of Extension Headers should be:



These headers:

* 1. should be processed by First and subsequent destinations.
* 2. should be processed by Final Destination.

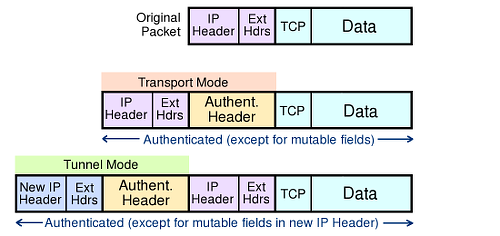
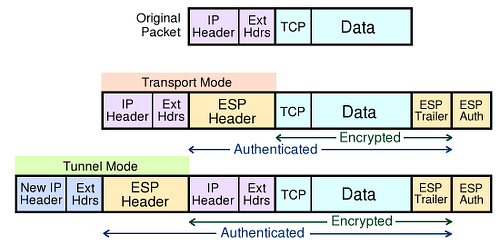
Extension Headers are arranged one after another in a linked list manner, as depicted in the following diagram:



**Q2.** The mechanisms and architectures used to provide QoS in a network are implemented in the same way in both versions of IP. The difference between IPv4 and IPv6 QoS focuses on the process of traffic classification in which packets or flows are differentiated by various parameters such as source or destination address, DSCP, or IP precedence values and types of protocol levels. IPv6 provides easier classification of packets with identifiers of traffic. Additionally, the flow label field has the advantage of being located before the address fields, helping to reduce delays in the verification of the package. The flow label field of the IP header has not been standardized. At the moment, there are rules of use and proposed use, which only cover some aspects of its implementation. At a commercial level, the flow label field has not been implemented so the QoS in IPv6 works in the same way as in IPv4. It is necessary that both applications and higher layer protocols can specify the value of the flow label field. Additionally, not all QoS features applied to IPv4 and IPv6 have been developed, which does not allow better management of QoS services in IPv6.

**Q3.** Unlike IPv4, IPsec security is mandated in the IPv6 protocol specification, allowing IPv6 packet authentication and/or payload encryption via the Extension Headers. However, IPsec is not automatically implemented, it must be configured and used with a security key exchange.

IPsec defines cryptography-based security for both IPv4 and IPv6 in RFC 4301. IPsec support is an optional add-on in IPv4, but is a mandatory part of IPv6. It provides two security headers which can be used separately or together: Authentication Header (AH) and Encapsulating Security Payload (ESP), used in conjunction with security key exchange.

* **Authentication Header**  
  AH provides connectionless integrity, data-origin authentication and protection against replay attacks. It authenticates with an Integrity Check Value (ICV) calculated over the payload, the header, and unchanging fields of the IPv6 header and options. AH does not provide privacy and confidentiality of packet contents. See RFC 2402.
* **Encapsulating Security Payload**  
  ESP also provides connectionless integrity, data-origin authentication, protection against replay attacks, limited traffic flow confidentiality, but also provides privacy and confidentiality through encryption of the payload. See RFC 2406.
* **IPsec Modes**  
  IPSec operates in two different modes: Transport mode (host-to-host) and Tunnel mode (gateway-to-gateway or gateway-to-host).  
  + **Transport mode:** the IPv6 header of the original packet is used, followed by the AH or ESP header, then the payload.
  + **Tunnel mode:** a new IPv6 header encapsulates the AH or ESP header and the original IP header and payload.Extension headers (Hop-by-Hop, Routing, Fragmentation) immediately follow their IP headers, except for Destination Options, which can appear before or after AH or ESP. ('TCP' below indicates any upper layer protocol.)
* **AH in Transport & Tunnel Modes**  
    
    
  AH authenticates the packet and the outermost IPv6 addresses (except for mutable fields), but does not encrypt payloads. AH cannot be used to traverse NATs, as it calculates the integrity check value (ICV) over source and destination addresses: NATs translate addresses, so would invalidate ICVs.
* **ESP in Transport & Tunnel Modes**  
    
    
  ESP authentication does not include the outermost IPv6 headers, but in Tunnel mode it protects the original headers. ESP is used to build virtual private network tunnels between sites. It permits NAT traversal, as it does not use the outermost address values in the ICV calculation. If AH and ESP are used together, ESP is applied first, then AH authenticates the entire new packet.
* **The Security Association**  
  Security Association is a record of the authentication algorithm, encryption algorithm, keys, mode (transport or tunnel), sequence number, overflow flag, expiry of the SA, and anti-replay window. The SA is held in a database at each endpoint, indexed by outer destination address, IPsec protocol (AH or ESP), and Security Parameter Index value. Selection of SA can be manually (pre-shared keys) but preferably is automated with Internet Key Exchange (IKE, IKEv2). IKE uses Diffie-Hellman techniques to create a shared secret encryption key used to negotiate SA data. For key exchange, IKE depends on a Public Key Infrastructure (PKI), which is not yet widespread. The framework and syntax for key exchange is ISAKMP (Internet Security Association and Key Management Protocol). See RFC 2408.

**Q4.** Some of the possible strategies are discussed below.

* **Stateless IP/ICMP Translation (SIIT)**

SIIT translates between the packet header formats in IPv6 and IPv4. The SIIT method defines a class of IPv6 addresses called IPv4-translated addresses. They have the prefix ::ffff:0:0:0/96 and may be written as ::ffff:0:a.b.c.d, in which the IPv4 formatted address a.b.c.d refers to an IPv6-enabled node. The prefix was chosen to yield a zero-valued checksum to avoid changes to the transport protocol header checksum. The algorithm can be used in a solution that allows IPv6 hosts that do not have a permanently assigned IPv4 address to communicate with IPv4-only hosts. Address assignment and routing details are not addressed by the specification. SIIT can be viewed as a special case of stateless network address translation.

* **Tunnel broker**

A tunnel broker provides IPv6 connectivity by encapsulating IPv6 traffic in IPv4 Internet transit links, typically using 6in4. This establishes IPv6 tunnels within the IPv4 Internet. The tunnels may be managed with the Tunnel Setup Protocol (TSP) or AYIYA.

* **6rd**

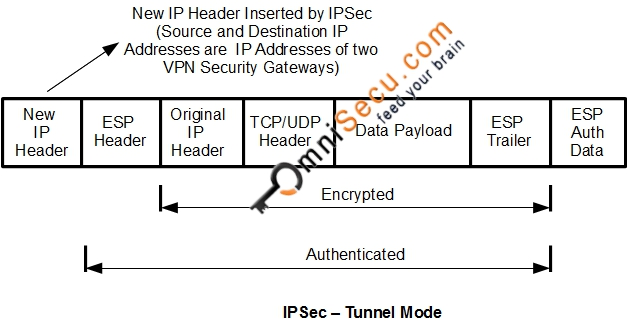
6rd is a mechanism to facilitate rapid deployment of the IPv6 service across IPv4 infrastructures of Internet service providers (ISPs). It uses stateless address mappings between IPv4 and IPv6 addresses, and transmits IPv6 packets across automatic tunnels that follow the same optimized routes between customer nodes as IPv4 packets.

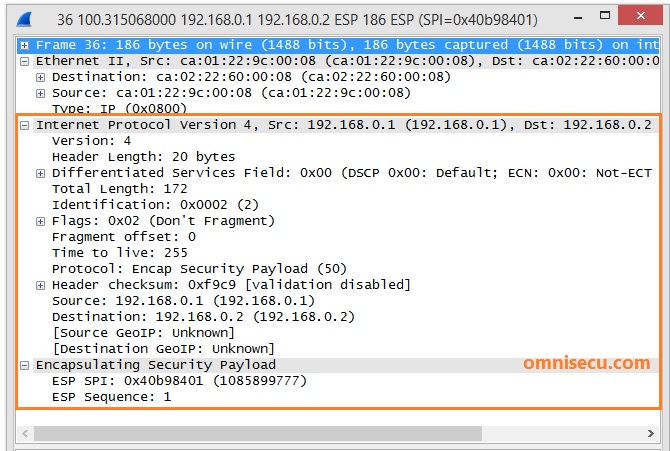
* **DNS64**

DNS64 describes a DNS server that when asked for a domain's AAAA records, but only finds A records, synthesizes the AAAA records from the A records. The first part of the synthesized IPv6 address points to an IPv6/IPv4 translator and the second part embeds the IPv4 address from the A record. The translator in question is usually a NAT64 server. The standard-track specification of DNS64 is in RFC 6147. There are two noticeable issues with this transition mechanism:

* It only works for cases where DNS is used to find the remote host address, if IPv4 literals are used the DNS64 server will never be involved.
* Because the DNS64 server needs to return records not specified by the domain owner, DNSSEC validation against the root will fail in cases where the DNS server doing the translation is not the domain owner's server.

**Q5.** IPSec does not work if we have a NAT Device between two IPSec peers, performing Port Address Translation. It is not possible for the IPSec ESP packets to traverse (Travel across or pass over) across a NAT Device performing PAT.Before proceeding, you need to know what is Network Address Translation (NAT) and what is Port Address Translation (PAT).In Port Address Translation (PAT), the NAT Device changes the source Port Number (TCP or UDP) with another port number.To perform Port Address Translation (PAT), a NAT device must be able to open TCP/UDP header and find Source TCP/UDP Port Number. The TCP and UDP Port Numbers are not visible for a NAT device performing PAT between IPSec Peers, because TCP/UDP headers are encrypted and encapsulated with ESP header. When IPSec is used to secure IPv4 traffic, original TCP/UDP Port Numbers are kept encrypted and encapsulated using ESP. Following image shows how IPSec encapsulates IPv4 datagram. For more details visit IPSec VPN Modes - Tunnel Mode and Transport Mode.

Following image shows a Wireshark capture of ESP encapsulated IPSec packet.



Note that TCP/UDP headers are not visible. TCP/UDP headers are kept encrypted as ESP data payload.

NAT Traversal (NAT-T) technology is used in IPSec to overcome the above mentioned problem.

NAT Traversal (NAT-T) technology can detect whether both IPSec peers support NAT-T. NAT Traversal (NAT-T) technology can also detect NAT devices between IPSec Peers. ISAKMP Main Mode messages one and two are used to detect whether both IPSec peers support NAT-T. If both IPSec peers support NAT-T, NAT Devices are detected in ISAKMP Main Mode messages three and four.

Once a NAT PAT device is detected between IPSec Peers, NAT-T encapsulates ESP packets inside an unencrypted UDP header with both Source and Destination ports as 4500. Now the NAT PAT devices have a UDP header and port number to play with and PAT happens as usual.

**Q7.** TCP fairness requires that a new protocol receive no larger share of the network than a comparable TCP flow. This is important as TCP is the dominant transport protocol on the Internet, and if new protocols acquire unfair capacity they tend to cause problems such as congestion collapse. To compare both networks in terms of fairness, one important point we should take into account is the underlying network assumed by TCP Vegas. When the original TCP Vegas was proposed, the authors did not consider the RED (Random Early Detection) mechanism, which is now being introduced in the operating network. TCP Vegas may or may not be effective even when the router is equipped with the RED mechanism. We therefore consider two packet scheduling mechanisms, the RED router as well as the conventional drop-tail router.

* Drop-tail router

TCP Vegas connections suffer from significantly low throughput, compared with TCP Reno connections. It is due to the difference of buffer occupancy at the router. TCP Reno connections can increase their window sizes until the buffer becomes full and packet loss occurs. On the other hand, TCP Vegas connections do not inflate the window size larger than . The average window size of TCP Reno connections becomes large as the router buffer size B[packets] is increased. The increase of the window size of each TCP Reno connection can directly lead to the throughput improvement. On the other hand, the window size of TCP Vegas connections remains unchanged regardless of the router buffer size. Therefore, buffer occupancy of TCP Vegas connections is decreased as the router buffer size is set to be large. That is, the larger the router buffer size becomes, the worse the fairness between TCP Reno and TCP Vegas connections becomes.

* RED router

The fairness between two versions of TCP is greatly improved when compared with the case of drop-tail router, while the total throughputs of all connections are almost identical for the large buffer size. It can be explained as follows. With the RED algorithm, TCP Reno connections do not inflate their window sizes until the router buffer becomes fully utilized, since packet loss occurs before the buffer becomes full due to the essential nature of the RED algorithm. It results in the decrease of buffer occupancy of TCP Reno connections, leading to throughput degradation of TCP Reno connections. It also contributes the throughput improvement of TCP Vegas connections. The observation can be confirmed by our analysis. In contrast with the drop-tail router case, the window size of TCP Reno is independent of the router buffer size, since the total number of packets transmitted between two events of packet losses is only dependent on the packet-dropping probability of the RED algorithm p. Therefore, throughput values of two versions are not changed even when the router buffer size becomes large.

# References

|  |  |  |
| --- | --- | --- |
| [1] |  | "IPv6 - Headers," [Online]. Available: https://www.tutorialspoint.com/ipv6/ipv6\_headers.htm. |
| [2] |  | "Internet Protocol version 6 (IPv6) Header," [Online]. Available: https://www.geeksforgeeks.org/internet-protocol-version-6-ipv6-header/. |
| [3] |  | "IPv6 packet," [Online]. Available: https://en.wikipedia.org/wiki/IPv6\_packet. |
| [4] |  | O. Salcedo, A. Rios and G. Rubio, "Quality of Service over IPV6 and IPV4," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011* , 2011. |
| [5] |  | "IPv6 Packet Security," [Online]. Available: http://www.ipv6now.com.au/primers/IPv6PacketSecurity.php. |
| [6] |  | "IPv6 transition mechanism," [Online]. Available: https://en.wikipedia.org/wiki/IPv6\_transition\_mechanism. |
| [7] |  | "What is NAT-Traversal (Network Address Translation - Traversal)," [Online]. Available: http://www.omnisecu.com/tcpip/what-is-nat-traversal-in-ipsec.php. |
| [8] |  | "Fairness measure," [Online]. Available: https://en.wikipedia.org/wiki/Fairness\_measure. |
| [9] |  | K. Kurata, G. Hasegawa and M. Murata, "Fairness Comparisons between TCP Reno and TCP Vegas for Future Deployment of TCP Vegas," 2000. |