**Chapter 27: IPv6 Protocol**

Table of Contents

[27.2 Packet Format 3](#_Toc86834308)

[Base Header 4](#_Toc86834309)

[Extension Headers 5](#_Toc86834310)

[Security 7](#_Toc86834311)

[27.3 Transition from IPv4 to IPv6 8](#_Toc86834312)

[Dual Stack 8](#_Toc86834313)

[Tunnelling 8](#_Toc86834314)

[Header Translation 9](#_Toc86834315)

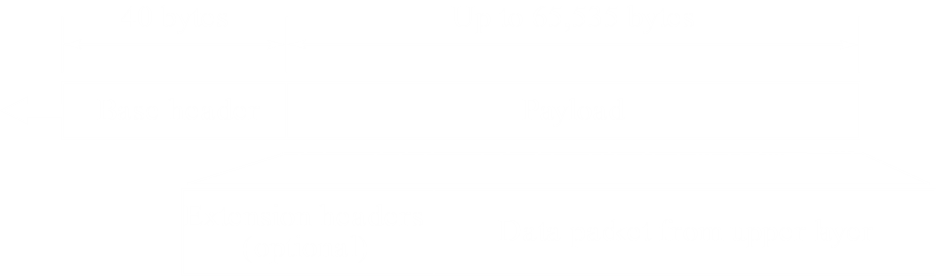
The **IPv6 Protocol** was proposed was several reasons:

* Larger **address space**
* Better **header format**
* New **options**
* Allowance for **extension**, which guarantees the protocol can be extended in the future.
* Support for **resource allocation**, required for real-time communication.
* Support for **security**
* Elimination of **unnecessary processing** of mandatory headers, such as for options that were rarely used like fragmentation, by implementing them as optional headers.

Despite these benefits, IPv6 has still not been fully adopted globally. The reasons for this include the introduction of new technologies to IPv4 such as:

* **Classless addressing**
* **DHCP** for dynamic addressing
* **NAT**

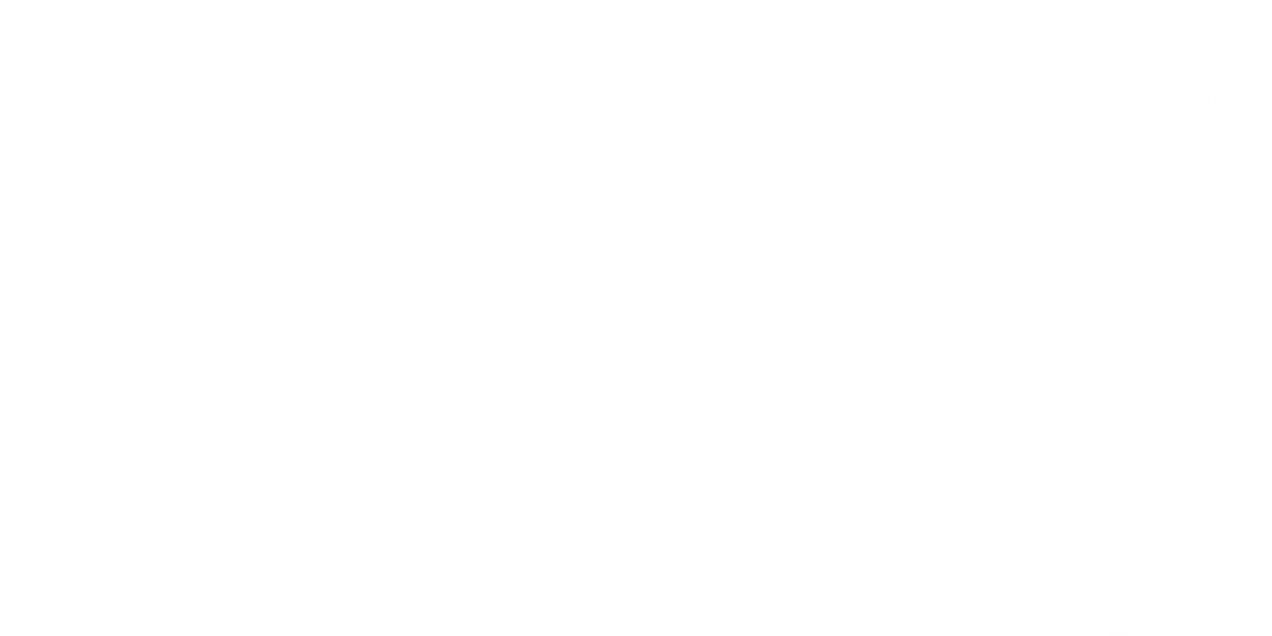
## 27.2 Packet Format



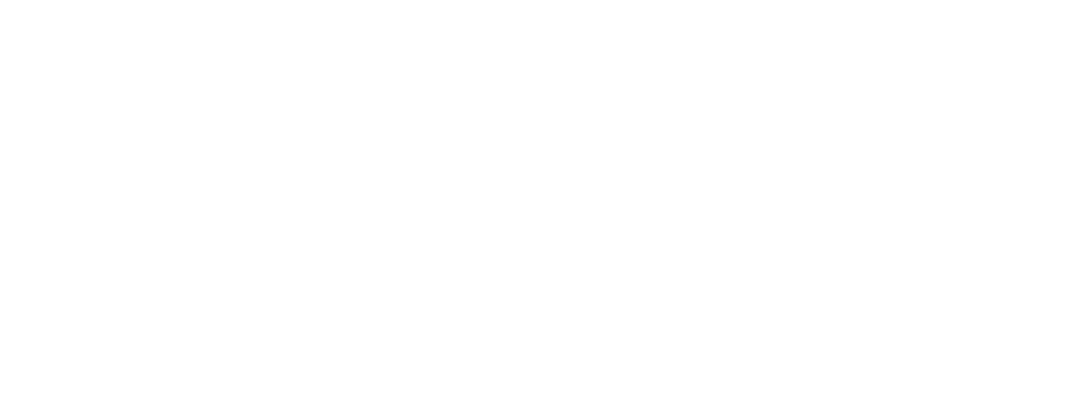
An **IPv6 packet** looks like the diagram above.

The first difference in this new packet format is the fixed **base header**. 40 bytes worth of data is guaranteed to be present at the start of every IPv6 packet. This additional led to the elimination of some fields, such as the **header length** field, since it is fixed.

Next, we have the **payload** section, which is divided into two parts, the **extension headers** and the **data packet** from the upper layer. The extension headers are where different **options** are added. This is implemented as a linked list, with the base header pointing to the first extension header and that one pointing to the second and so on. The last extension header (or the base header, if there are no extension headers), will point to the data from the upper layer. In total, up to **six extension headers** can be added.



### Base Header



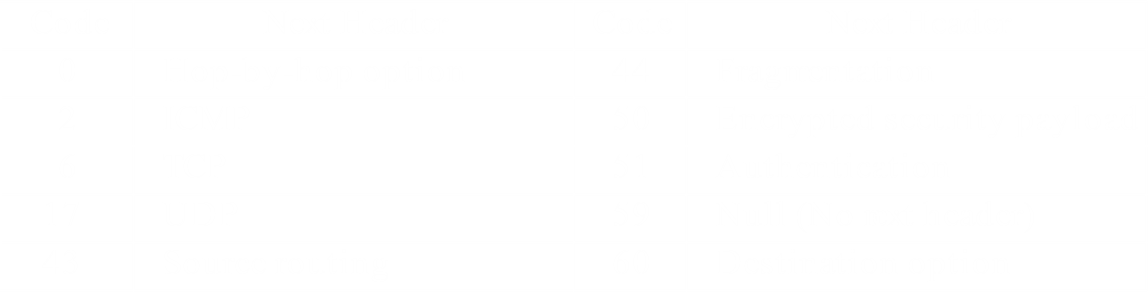
This is what the **mandatory base header** looks like.

* **VER** - This is the protocol version, which will be 6 in this case.
* **Traffic Class** – This is similar to the Type of Service field from IPv4.
* **Flow Label** – This allows for connection-oriented routing approaches. Routers can ignore the routing table and use the flow label to determine how to route the packet, similar to Virtual Circuit Switching. This, along with the Traffic Class field, makes IPv6 fast.
* **Payload Length** – Since we know the header length is fixed, we do not need separate fields for Header Length and Total Length anymore. We just need to know the Payload Length.
* **Next Header** – This is a pointer to the next extension header or the data.
* **Hop Limit** – This is the same as the TTL field.
* **Source Address**
* **Destination Address**

Notice that the **checksum** field is missing. IPv6 leaves this to the Transport Layer, which is the one that needs the checksum.

The **Fragmentation fields** are also missing, since it was very rarely used. This is why intermediary routers are not allowed to fragment packets, only the source machine. To do this, the source machine needs to know the entire path beforehand, since it will have to fragment packets based on the capabilities of the path.

### Extension Headers



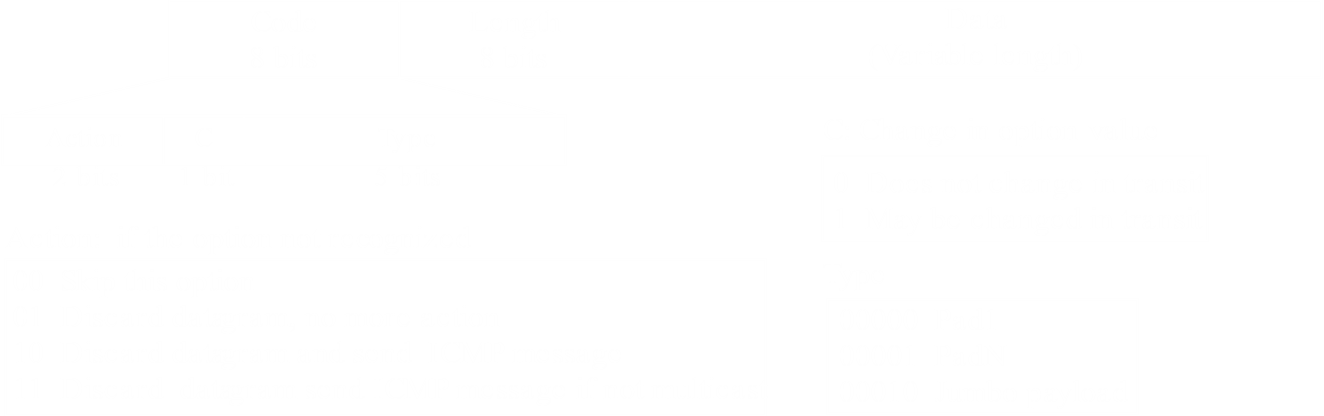
The table above shows the different possible **extension headers**.

IPv6 supports a total of 6 different **options**, which are implemented as extension headers.

* **Hop-by-Hop** – This utterly misleadingly named option allows for the alignment features that were provided by the No-Operation and End-of-Operation flags from IPv4. Here, they are called **Pad1**, which is used for a single byte of padding, and **PadN**, which is used for bytes of padding. This is needed when a particular option needs to start from a specific position in the header, which requires padding to get it to that position.

Another type of the Hop-by-Hop option is **Jumbo Payload**, which is used for huge amounts of data, larger than the 65 kilobytes provided.





* **Destination** – This allows the source machine to send some information to the destination machine that none of the intermediary routers are able to read.
* **Source Routing** – This is the same as source routing in IPv4, i.e. the hops are predefined. Both Strict Source Routing and Loose Source Routing are supported.
* **Fragmentation** – This is the same as the fragmentation process from IPv4. It can only occur at the source, not in intermediary routers.
* **Authentication** – This provides authentication services.



The **security parameter index** identifies the specific algorithm used.

* **Encrypted Security Payload** (ESP) – This provides encryption services.



### Security

Security has three parts:

* **Confidentiality**
* **Integrity**
* **Availability**

ESP provides confidentiality. Since the data is encrypted, it remains confidential.

Authentication provides both integrity and availability. It provides integrity since it can be verified that data genuinely came from the sender it claims to be coming from. It provides availability since the data will be available to only those with proper authentication, thus combating things like [DOS attacks](https://en.wikipedia.org/wiki/Denial-of-service_attack).

Confidentiality is easier to achieve than integrity. Encrypting data is easy enough, but ensuring that data that is publicly available is genuinely from the source is much more difficult. For example, if we find an official looking document that says all students from a grade received full marks in a test, we may have concerns about whether the document is correct. This concern is a concern about the integrity of the document.

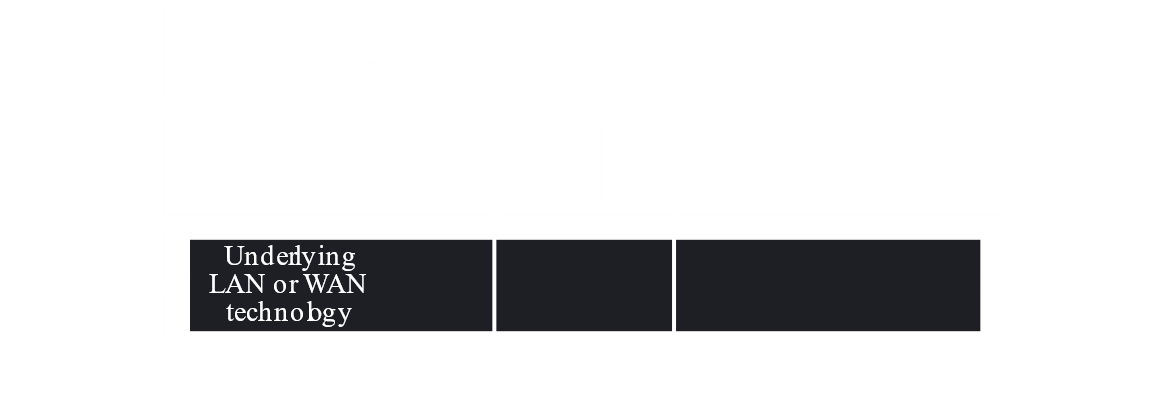
## 27.3 Transition from IPv4 to IPv6

It is not practically possible to shift from IPv4 to IPv6 for every device in the world at once. In order to deal with this, three mechanisms have ben proposed to allow for graceful transition:

1. Dual Stack
2. Tunnelling
3. Header Translation

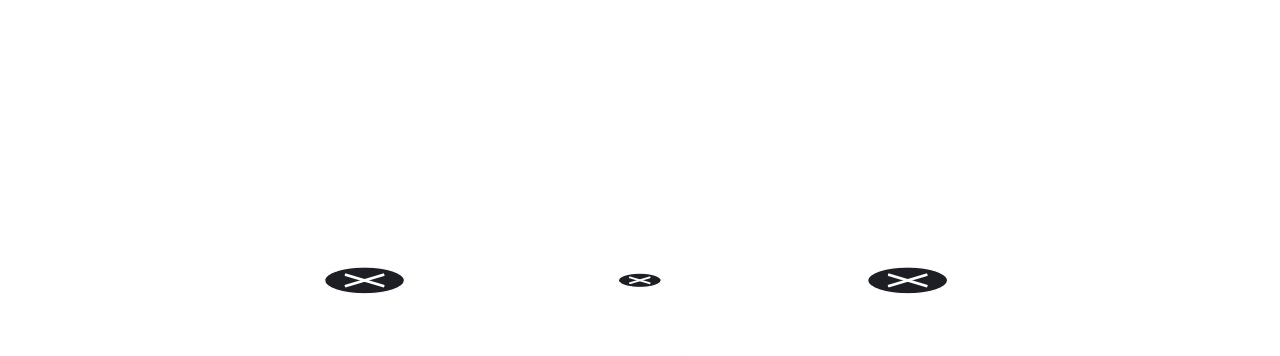
### Dual Stack

**Dual Stack** essentially says that nodes that support IPv6 should also support IPv4. The node can use IPv4 or IPv6 depending on the address.



### Tunnelling

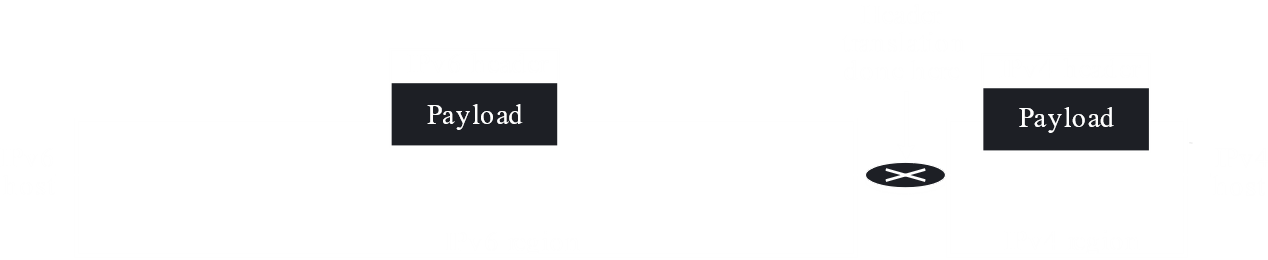
If the situation is such that both the sender and receiver support IPv6 but intermediary routers do not, then we have a situation called **Tunnelling**. This is resolved by using **compatible address**, which are IPv6 addresses generated from IPv4 addresses, as discussed in the previous chapter.



Essentially, an IPv6 packet is placed inside an IPv4 packet, so the receiver can extract the original IPv6 packet.

### Header Translation

**Header Translation** is used when just the receiver does not support IPv6. The header is translated from IPv6 to IPv4 by a router before it reaches the receiver.



For Header Translation, we used **mapped addresses**, as studied in the previous chapter.