ECE4110
Internetwork Programming

Introduction and Overview
Listen to wire

Detect a preamble

Yes

No

Read Destination Address

Broadcast Address

Yes

No

My Address

Yes

No

Read Data Frame Contents

End of Frame

Yes

No

Perform Integrity Check

Check Passed?

Yes

Deliver data to Designated Process

No

Ignore Transmission

Are signals detected data carrying or noise?

Is the listening computer the intended party or is it for someone else?

Where does the message end?

Did the computer get a good message or is it corrupted?
DIX Ethernet Data Frame Format

<table>
<thead>
<tr>
<th>Preamble (8 Bytes)</th>
<th>Destination MAC Address (6 Bytes)</th>
<th>Source MAC Address (6 Bytes)</th>
<th>Type (2 Bytes)</th>
<th>Data 46-1500 bytes</th>
<th>CRC (4 Bytes)</th>
<th>Gap 12 Bytes</th>
</tr>
</thead>
</table>

- Preamble - alternating 0’s and 1’s with last bit a 1. Must not be noise if see this.
- Destination medium access control (MAC) address - compare this to our own hardwired network interface card (NIC) address. Standards exist and manufactures comply.
  - Normally a NIC knows its own address and the broadcast address.
- Source MAC address - sender’s NIC address.
- Type - so we know what kind of data transporting, one example is Internet Protocol.
- Cyclic Redundancy Check - so we can tell if what we got was received correctly, if not throw it away.
- Aside: After the Ethernet Frame, a 12 Byte Interframe Gap must always follow.

**Note:** Receive process does not look at data, that is done elsewhere.

**Note:** Data is in most network technologies called a “packet.”
Computer X on Network 1 wants to send a message to computer Y on Network 5.
NETWORK ADDRESS IS SOFTWARE CONFIGURABLE

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Destination Mac Address</th>
<th>Source Mac Address</th>
<th>Type</th>
<th>INFO</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Destination: 5 : Y
- Source: 1 : X
- Other Stuff
- Data

ROUTING TABLES

Computer X

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Distance</th>
<th>Next Router</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>A</td>
<td>X1</td>
</tr>
</tbody>
</table>
Router A’s Routing Table:

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Distance</th>
<th>Next Router</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>C</td>
<td>A2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>A</td>
<td>A1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>F</td>
<td>A3</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Routers can Dynamically exchange information or they can be statically configured.
What Happens To Ethernet Frame and Packet Address As Data Frame Goes From Computer X To Computer Y

<table>
<thead>
<tr>
<th>Data Presently On Network #</th>
<th>Mac Destination Address</th>
<th>Mac Source Address</th>
<th>Packet Dest Address</th>
<th>Packet Source Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>X</td>
<td>Y : 5</td>
<td>X : 1</td>
</tr>
<tr>
<td>9</td>
<td>C</td>
<td>A</td>
<td>Y : 5</td>
<td>X : 1</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>C</td>
<td>Y : 5</td>
<td>X : 1</td>
</tr>
<tr>
<td>5</td>
<td>Y</td>
<td>D</td>
<td>Y : 5</td>
<td>X : 1</td>
</tr>
</tbody>
</table>

Routing - Intelligent delivery of data from point A to point B on an Internetwork.

Router - Perform routing service.

Typically uses shortest path

Reference: “Networking Unix”, by Douba, Sams Publishing Chapter 1,2.
Layered Internet Protocol Stack vs OSI Reference Model

Internet Protocol Stack

OSI Reference Model

<table>
<thead>
<tr>
<th>Application</th>
<th>Transport</th>
<th>Internet</th>
<th>Physical</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Transport</th>
<th>Network</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation</td>
<td>Transport</td>
<td>Data Link</td>
<td>Physical</td>
</tr>
<tr>
<td>Session</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operation Of TCP/IP Concepts

Host A

- APPX
- APPY
- TCP
- IP
- Physical

Global Network Address

ethernet 1

Router J

IP Interface1 Interface2

Host B

- APPX
- APPY
- TCP
- IP
- Physical

ethernet 2
Operation of TCP/IP Concepts Continued:

- IP: is implemented in all end systems and routers.

- TCP: is implemented only in the end systems.

- Each host on a subnetwork must have a unique global internet address which is used by IP for routing and delivery.

- Each application within a host must have an address unique within host used by TCP. Addresses are known as ports.
Tracing Through This Example Network:

Suppose: Port 1 Host A to Port 2 Host B.
Host A sends message down to TCP.

TCP hands down to IP.

IP down to network access layer (ethernet) with instructions to send to Router J.

At Router J ethernet packet header examined. Using IP header info know to send on to subnetwork 2.

At Host B all headers removed.

Reference: “High speed networks TCP/IP and ATM design principles”
By Stallings chapter 2, 10, 11
**The TCP/IP Protocol Architecture**

**Communication Task Layers:**

- **Physical layer** - Physical interface between computer and the network (nature of signals, data rate).

- **Link Layer** - Access to and sending data across a network for two end systems attached to the network. Example: ethernet

- **Internet layer** - Allows data to traverse multiple interconnected networks. Internet protocol (IP) is used at this layer to provide the routing function across multiple networks.
  - IP implemented in end systems (computers) and also in routers. ( Everywhere)
  - Router is a processor that connects two networks with primary function to relay data from one network to the other on its route from the source to the destination.

- **Transport layer** - Allows data to be exchanged reliably. Transmission control protocol (TCP) is the most commonly used protocol to do this. (End Stations)

- **Application layer** - module to perform support for application like FTP, TELNET, etc.
## Another View

<table>
<thead>
<tr>
<th>SMTP</th>
<th>FTP</th>
<th>TFTP</th>
<th>DNS</th>
<th>SNMP</th>
</tr>
</thead>
</table>

### Application

| TCP | UDP |

### Transport

<table>
<thead>
<tr>
<th>IGMP</th>
<th>ICMP</th>
</tr>
</thead>
</table>

### Internet

| IP   | ARP | RARP |

### Physical

| Ethernet |

---

SMTP – Simple Mail Transfer Protocol  
FTP- File Transfer Protocol  
TFTP- Trivial File Transfer Protocol  
DNS- Domain Name Service  
SNMP- Simple Network Management Protocol  
TCP- Transmission Control Protocol  
UDP- User Datagram Protocol  
IGMP- Internet Group Management Protocol  
ICMP- Internet Control Message Protocol  
IP- Internet Protocol  
ARP- Address Resolution Protocol  
RARP Reverse Address Resolution Protocol
Protocol Data Units In TCP/IP Architecture

<table>
<thead>
<tr>
<th>APPLICATION DATA</th>
<th>APPLICATION BYTE STREAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/UDP HEADER</td>
<td>TCP SEGMENT OR UDP USER DATAGRAM</td>
</tr>
<tr>
<td>IP HEADER</td>
<td>IP DATAGRAM</td>
</tr>
<tr>
<td>ETHERNET HEADER</td>
<td>PHYSICAL LEVEL PACKET (For Example Ethernet Frame)</td>
</tr>
</tbody>
</table>
TCP Header:

- Bit 0: Source Port
- Bit 4: Destination Port
- Bit 10: Sequence Number
- Bit 15: Acknowledgement Number
- Bit 31 (32 Bits): Header Length
- Flags: URG, ACK, PSH, RST, SYN, FIN
- Window
- Checksum
- Urgent Pointer
- Options + Padding
TCP Header Details:

- **Source Port** (16 bits) – source TCP user.

- **Destination Port** (16 bits) – destination TCP user.

- **Sequence Number** (32 bits) – sequence number of the first data octet in this segment except when SYN flag is set. If SYN is set this field is the initial sequence number (ISN).

- **Acknowledgement Number** (32 bits) – a piggybacked acknowledgement, contains the sequence number of the next data octet that the TCP entity expects to receive.

- **Header Length** (4 bits) – number of 32 bit words in the header.

- **Unused** (6 bits) – unused.
TCP Header Details Continued:

- **Flags** (6 bits) –
  - URG – urgent pointer field significant.
  - ACK – acknowledgement field significant.
  - *PSH* – push function.
  - RST – reset the connection.
  - SYN – synchronize the sequence numbers.
  - FIN – no more data from sender.

- **Window** (16 bits) – flow control credit allocation contains the number of data octets beginning with the one indicated in the acknowledgement field that the receiver is willing to accept.

- **Checksum** (16 bits) – an error detection calculated over header and data.

- **Urgent Pointer** (16 bits) – points to the last BYTE in a sequence of urgent data.

- **Options** (variable) – optional features.

* **Data Stream Push Note**: Normally TCP decides when sufficient data have accumulated to form a segment for transmission. The user can require TCP to transmit all outstanding data up to and including that labeled with a push flag.
TCP OPTIONS:

SINGLE BYTE:

END OF OPTION  00000000 (Only one may be used and is only necessary if the very last byte of the last 32 bit word is not already the end of an option)

NO OPERATION  00000001 (Used for Padding/aligning an option on a 16 bit or a 32 bit boundary)

MULTIPLE BYTE:

MAXIMUM SEGMENT SIZE:

<table>
<thead>
<tr>
<th>ID CODE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000010</td>
<td>0000100</td>
</tr>
</tbody>
</table>

WINDOW SCALE FACTOR:

<table>
<thead>
<tr>
<th>ID CODE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000011</td>
<td>0000011</td>
</tr>
</tbody>
</table>

TIMESTAMP (USED IN ROUND TRIP TIME CALCULATIONS):

<table>
<thead>
<tr>
<th>ID CODE</th>
<th>ID CODE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000100</td>
<td>0001010</td>
<td>4 BYTE TIME STAMP VALUE</td>
</tr>
</tbody>
</table>

Example:
<mss 512, nop, wscale 0, nop, nop, timestamp 146647>
TCP CHECKSUM CALCULATION AT THE SENDER

1. Add the pseudoheader to the TCP segment
2. Fill the checksum field with zeros
3. Divide the total bits into 16 bit words
4. If total length is not a multiple of 16 bits add one byte of all zeros. (Only used in the checksum calculation, not sent in datagram)
5. Add 16 bit words using ones complement arithmetic
6. Complement the result and put this in checksum field of header
7. Drop pseudoheader and any padding
TCP CHECKSUM CALCULATION AT THE RECEIVER

1. Add the pseudoheader to the received datagram.
2. Add padding if needed
3. Divide total bits into 16 bit sections
4. Add all sections using ones complement arithmetic
5. Complement the result
6. If the result is zero drop the pseudo header, remove padding if any, and you have the TCP segment.

Aside: Ones complement arithmetic used end around carry. If the final column of numbers has a carry, add that carry back into the sum. (See Appendix C pp 783-784 Fououzan)

Note: The purpose of the pseudo-header is to double-check that the data has arrived to the correct host.
User Datagram Protocol (UDP) Is Another Possibility

UDP Header:

<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>31 (32 Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE PORT</td>
<td>DESTINATION PORT</td>
<td></td>
</tr>
<tr>
<td>SEGMENT LENGTH (# of BYTES)</td>
<td>CHECKSUM</td>
<td></td>
</tr>
</tbody>
</table>
UDP CHECKSUM CALCULATION AT THE SENDER

1. Add the pseudoheader to the UDP datagram
2. Fill the checksum field with zeros
3. Divide the total bits into 16 bit words
4. If total length is not a multiple of 16 bits add one byte of all zeros. (Only used in the checksum calculation, not sent in datagram)
5. Add 16 bit words using ones complement arithmetic
6. Complement the result and put this in checksum field of header
7. Drop pseudoheader and any padding

<table>
<thead>
<tr>
<th>32 BIT SOURCE IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 BIT DESTINATION IP ADDRESS</td>
</tr>
<tr>
<td>8 BITS OF ZEROS</td>
</tr>
<tr>
<td>SOURCE PORT</td>
</tr>
<tr>
<td>UDP TOTAL LENGTH (*SAME AS ABOVE)</td>
</tr>
<tr>
<td>DATA (THIS IS TEMPORARILY A MULTIPLE OF 16 BITS BECAUSE OF Padding IF NEEDED)</td>
</tr>
</tbody>
</table>
### UDP SENDER CHECKSUM CALCULATION EXAMPLE

<table>
<thead>
<tr>
<th>1.0.2.1</th>
<th>2.1.0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>THIS IS WHAT WE ARE CALCULATING</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

00000001 00000000  1.0  
00000010 00000001  2.1  
00000010 00000001  2.1  
00000000 00000000  0.0  
00000000 00010001  0 and 17  
00000000 00001011  11  
00000000 00000011  3  
00000000 00000100  4  
00000000 00001011  11  
00000000 00000000  This is what we are calculating  
00000000 00000001  0 and 1 data  
00000010 00000000  2 and padding  

The ones complement sum for this equals:

00000111 00110001

Thus the checksum value entered into the datagram is:

11111000 11001110
### IP Header: (IPV4)

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
<td>Version of IP protocol</td>
</tr>
<tr>
<td>IHL</td>
<td>4</td>
<td>Internet Header Length</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
<td>Type of Service</td>
</tr>
<tr>
<td>Total Length</td>
<td>16</td>
<td>Total Length of Packet</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
<td>Identification of Packet</td>
</tr>
<tr>
<td>Flags</td>
<td>16</td>
<td>Flags of Packet</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>16</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>Time to Live</td>
<td>16</td>
<td>Time to Live</td>
</tr>
<tr>
<td>Protocol</td>
<td>16</td>
<td>Protocol Type</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16</td>
<td>Header Checksum</td>
</tr>
<tr>
<td>Source Address</td>
<td>32</td>
<td>Source Address</td>
</tr>
<tr>
<td>Destination Address</td>
<td>32</td>
<td>Destination Address</td>
</tr>
<tr>
<td>Options + Padding</td>
<td>32</td>
<td>Options + Padding</td>
</tr>
</tbody>
</table>

**20 Bytes**
IP Header Details:

- **Version** (4 Bits) - version number now is 4
- **Internet Header Length (IHL)** (4 Bits) - length of IP header in 32 bit words. Minimum value is 5 for a minimum header of 20 bytes. When option field max size, IHL = 15

- **Type of service** (8 bits)
  3 Bits Precedence 4 Bits TOS One Bit
  Not used at present 0000 normal service
  0001 minimize cost
  0010 maximize reliability
  1000 minimize delay

- **Total Length** (16 Bits) - Total IP datagram length in bytes including IP header. Maximum length value is $2^{16}-1 = 65,535$
It is possible to have an IP datagram that is too big for some networks. For example, the Maximum Transfer Unit (MTU) of data that may be encapsulated into an ethernet frame is 1,500 bytes. If we have an IP datagram larger than this, we will have to fragment the datagram. This means breaking the datagram into smaller pieces. The header is copied for use in every fragment. The flags, fragmentation offset, and the total length fields are set to new values in each fragment.

• **Identification** (16 Bits) - a sequence number. Counter in the source machine keeps track of how many IP datagrams it has sent. This value is copied into the identification field. Fragments all have the same identification number.

• **Flags** (3 Bits)
  • More Bit - when datagram is fragmented this bit indicates if this is the last fragment. A value of one indicates there are more fragments, a value of zero means this is the end.
  • Do not fragment bit - do not fragment if this bit is a one
  • Unused bit

• **Fragment Offset** (13 Bits) - where in the original datagram this fragment belongs measured in 64 bit units. This means divide the offset location in bytes by 8 to get the value in the header. Conversely multiply the value you see in a header before you use it. Since offset field is only 13 bits, use this divide/multiply by 8 method to allow larger than $2^{13}-1$ offsets.
Example:

Given a 3,500 byte IP datagram and an ethernet network with an Maximum Transfer Unit = 1,500 bytes. Assume there are no options included in the 20 byte IP header.

We have 3,500 total bytes – 20 header bytes = 3,480 data bytes to transport.

The MTU limit = 1,500 so the largest a data payload may be is 1,500 – 20 header = 1480

The amount of data we carry must be a multiple of 8 because of the offset field definition.

184 * 8 = 1,472
185 * 8 = 1,480
186 * 8 = 1,488

We should choose 1,480 data bytes in the first segment so that the total size is 1480 data plus 20 header = 1,500 which is equal to or less than the MTU of 1,500. The offset field will have the value 0, the more flag will be equal to 1.

The second datagram will also carry 1,480 bytes with the offset field equal to 1480/8= 185. The more flag will be equal to 1

The third datagram will carry the remaining 3,480 – (1,480 + 1,480) = 520 bytes. It will be carried in an IP datagram with 520 + 20 Overhead bytes = 540 bytes. The offset field will be (1,480+1,480) / 8 )=370, and the more flag will equal 0.
In the event a fragment is itself fragmented, then the offset value used is relative to the original datagram prior to any fragmentation.

Example:
Suppose that second datagram in the previous example (which is 1,500 bytes in size) is further fragmented into two fragments because we encounter a smaller MTU = 766.

The resulting new fragments for the second fragmented datagram from our original example are:
For an MTU of 766 we may have a max of 766 – 20 overhead = 746 data bytes.
746/8 = 93.25 so we should use 93*8 = 744 data bytes in one new segment,
leaving 1480 total data – 744 in one segment = 736 data left for next segment. So
First additional segment has 744 + 20 = 764 bytes total, offset value = same old start of 185 from before.
Next additional segment has 736 + 20 = 756 bytes total, new offset = 185 + 744/8 = 185 + 93 = 278.
Note that the offset value is relative to the relative position of the data to the original (source) data.
IP Header Details:

- **Time To Live (8 Bits)** - how long datagram exists usually done by hop count. Every router decreases TTL by at least one.

- **Protocol (8 Bits)** - indicates next higher level protocol IE TCP, UDP, ICMP, IGMP, etc
  1 ICMP Internet Control Message Protocol
  2 IGMP Internet Group Message Protocol
  6 TCP Transmission Control Protocol
  17 UDP User Datagram Protocol
  41 IPV6
  89 OSPF Open Shortest Path First Routing Protocol

- **IP Header Checksum (16 Bits)** - applied to header only.

  At the sender the checksum field is first set to zero, then the header is divided into 16 bit pieces. These pieces are added together using one’s complement arithmetic with a result that is also 16 bits long. The sum is complemented (inverted). This is the check sum value put in the datagram header.

  At the receiver, the 16 bit pieces are once again added using ones complement arithmetic. If there are no errors in the header, the complemented result will be all zeros.
IP Checksum Calculation Example:

Given the following IP Header, calculate the IP checksum value.

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>0</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You are calculating this

255.255.0.0

1.1.0.0

4, 5, and 0
64
2
0 and 0
16 and 17
255 and 255
0 and 0
1 and 1
0 and 0
SUM
CHECKSUM

01000101 00000000
00000000 01000000
00000000 00000010
00000000 00000000
00100000 00010001
11111111 11111111
00000000 00000000
00000000 00000000
01010110 01010100
10101001 10101011

(This example had a 1 end around carry added back to the solution)
• **Source Address** - (32 Bits )

• **Destination Address** - (32 Bits )

• **Options**  (Variable)

• **Padding**  (Variable) - used so header always multiple of 4 Bytes.

• **Data**  (Variable) - max length including header 65, 535

Option bytes may or may not exist. If they do exist they look like this:
IP Options:

Single Byte:
   No Operation  00000001
   End of Options  00000000

Multiple Byte Varying in Length:
   Record Route
   Strict Source Routing
   Loose Source Routing
   Timestamp
The Copy Bit:
0 Copy only in the first fragment
1 Copy into all fragments

The Class Bits:
00 Datagram Control
10 Debugging and Management

The Type Bits:
00000 End of Operation
00001 No Operation
00011 Loose Source Routing
00100 Timestamp
00111 Record Route
01001 Strict Source Route
Example Loose Source Route Option:

<table>
<thead>
<tr>
<th>Code Field</th>
<th>Total Length</th>
<th>Integer Pointer Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO OP 00000001</td>
<td>10000011</td>
<td></td>
</tr>
</tbody>
</table>

| First IP address                   |              |                       |
| Second IP address                  |              |                       |
|                                   | .            |                       |
|                                   | .            |                       |
|                                   | .            |                       |

Last IP address

Note that these 4 byte words would be appended to the end of the IP header just before the data. That is where options are put.

A No Operation byte is a filler byte to make it so that options are always a multiple of 4 bytes.

Pointer is an integer number to the first empty byte in this option field. Makes more sense in an option that fills in fields as you go along.
(IPV6)

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Priority</td>
<td>Flow Label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload Length</td>
<td>Next Header</td>
<td>Hop Limit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP V6 Header Details:

- **Version** (4 Bits) – 0110 in binary means version 6

- **Priority** (4 Bits) – Similar to precedence field of IPV4

- **Flow Label** (28 Bits) – Distinguish packets that require the same treatment

- **Payload Length** - (16 Bits) Length of data carried after the header

- **Next Header** - (8 Bits) – Set to protocol type (UDP, TCP, etc) or extension header type sort of like options in IPV4

- **Hop Limit** (8 Bits)- Now named hops instead of Time to Live

- **Source Address** (128 Bits)

- **Destination Address** (128 Bits)