Ethernet at the Data Link Layer

Binary to Decimal and Hexadecimal Conversion

Understanding the differences between *binary*, *decimal*, and *hexadecimal* numbers and how to convert one format into the other is very important before discussing the **TCP/IP protocol stack** and **IP addressing** in further. So let's get started with binary numbering. Each digit used is limited to being either a 1 (one) or a 0 (zero), and each digit is called 1 bit (short for binary digit). Typically, you count either 4 or 8 bits together, with these being referred to as a **nibble** and a **byte**, respectively.

The binary numbers are placed in a value spot, starting at the right and moving left, with each spot having double the value of the previous spot. The below table shows the decimal values of each bit location in a nibble and a byte. Remember, a nibble is four bits and a byte is eight bits. In network addressing, we often refer to a byte as an **octet**.

Nibble Values	Byte Values
8 4 2 1	128 64 32 16 8 4 2 1

What all this means is that if one digit (1) is placed in a value spot, then the nibble or byte takes on that decimal value and adds it to any other value spots that have a 1. And if zero (0) is placed in a bit spot, you don't count that value. For example, if we have a 1 placed in each spot of our nibble, we then add up `8 + 4 + 2 + 1` to give us a maximum value of 15. Another example of our nibble values is `1010`, which means that the 8 bit and the 2 bit are turned on and equal a decimal value of `10`. If we have a nibble binary value of `0110`, then our decimal value is `6` because the 4 and 2 bits are turned on.

But the byte values can add up to a value that's significantly higher than 15. This is how—if we count every bit as a one (1), then the byte binary value looks like this (remember, 8 bits equal a byte):

We then count up every bit spot because each is turned on. It looks like this, which demonstrates the maximum value of a byte:

```
128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255
```

It is strongly advised that you memorize the below table before braving the IP sections in further lessons since this lists all available subnet masks.

Binary Value	Decimal Value
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

Hexadecimal addressing is completely different than binary or decimal—it's converted by reading nibbles, not bytes. By using a nibble, we can convert these bits to hex pretty simply. First, understand that the hexadecimal addressing scheme uses only the numbers 0 through 9. And because the numbers 10, 11, 12, and so on can't be used (because they are two-digit numbers), the letters A, B, C, D, E, and F are used to represent 10, 11, 12, 13, 14, and 15, respectively.

Suppose you have something like this: `0x6A`. (Some manufacturers put `0x` in front of characters so you know that they're a hex value, while others just give you an `h`. It doesn't have any other special meaning.) What are binary and decimal values? To correctly answer that question, all you have to remember is that each *hex* character is *one nibble* and *two hex* characters together make a *byte*. To figure out the binary value, first put the hex characters into two nibbles and then put them together into a byte. `6 = 0110` and `A (which is 10 in decimal) = 1010`, so the complete byte is `01101010`.

To convert from binary to hex, just take the byte and break it into nibbles. Here's how you do that: Say you have the binary number `01010101`. First, break it into nibbles—`0101` and `0101`—with the value of each nibble being 5 because the 1 and 4 bits are on. This makes the hex answer `0x55`. And in decimal format, the binary number is `01010101`, which converts to 64 + 16 + 4 + 1 = 85.

Hexadecimal Value	Binary Value	Decimal Value
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Ethernet Addressing

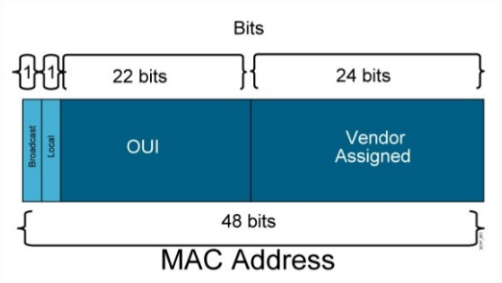
In order to communicate or transfer the data from one computer to another computer, we need some addresses. In Computer Network various types of addresses are introduced; each works at a different layer. **Ethernet addressing** uses the **Media Access Control (MAC)** address burned into each and every *Ethernet NIC*. **MAC Address** is a physical address which works at **Data Link Layer**.

MAC Addresses are unique 48-bits hardware number of a computer, which is embedded into the network card (known as Network Interface Card) during the time of manufacturing. MAC Address is also known as the Physical Address of a network device. In IEEE 802 standard, Data Link Layer is divided into two sublayers:

- Logical Link Control (LLC) Sublayer
- Media Access Control (MAC) Sublayer

MAC address is used by the **Media Access Control (MAC)** sublayer of the **Data Link Layer**. MAC Address is world wide unique since millions of network devices exist and we need to uniquely identify each.

MAC Address is a 12-digit hexadecimal number (6-Byte binary number), which is mostly represented by Colon-Hexadecimal notation. First 6-digits (say 00:40:96) of MAC Address identifies the manufacturer, called OUI (Organizational Unique Identifier). IEEE Registration Authority Committee assigns these MAC prefixes to its registered vendors. Some firms may have more than one MAC address.



MAC Address

Here are some OUI of well-known manufacturers:

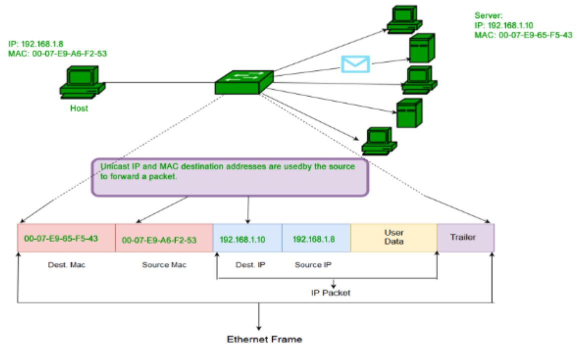
```
CC:46:D6 - Cisco
3C:5A:B4 - Google, Inc.
3C:D9:2B - Hewlett Packard
00:9A:CD - HUAWEI TECHNOLOGIES CO., LTD
```

The rightmost six digits represent the Network Interface Controller, which is assigned by the manufacturer.

Types of MAC Addresses

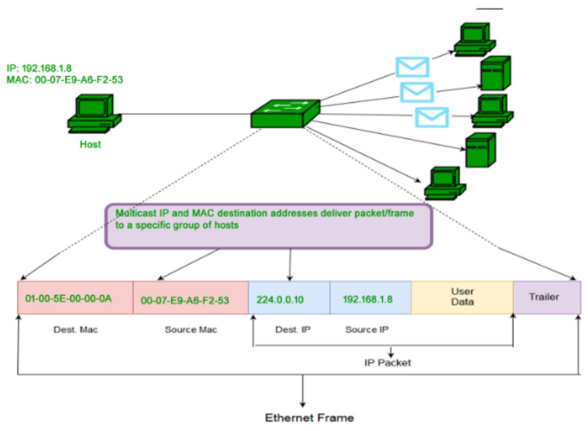
1. Unicast

A Unicast addressed frame is only sent out to the interface leading to specific NIC. If the LSB (least significant bit) of the first octet of an address is set to zero, the frame is meant to reach only one receiving NIC. MAC Address of source machine is always Unicast.



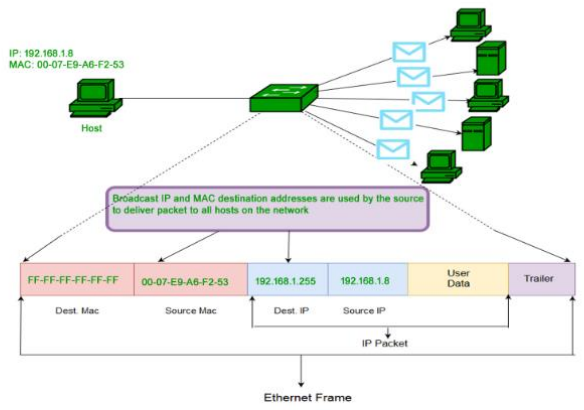
2. Multicast

The multicast address allows the source to send a frame to a group of devices. In Layer-2 (Ethernet) Multicast address, LSB (least significant bit) of the first octet of an address is set to one. IEEE has allocated the address block 01-80-C2-xx-xx-xx (01-80-C2-00-00-00 to 01-80-C2-FF-FF-FF) for group addresses for use by standard protocols.



3. Broadcast

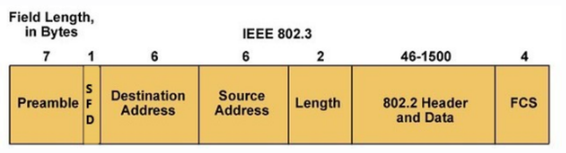
Similar to Network Layer, Broadcast is also possible on the underlying layer (Data Link Layer). Ethernet frames with ones in all bits of the destination address (FF-FF-FF-FF-FF-FF) are referred to as the broadcast address. Frames which are destined with MAC address FF-FF-FF-FF-FF-FF will reach to every computer belong to that LAN segment.



Ethernet Frames

When transmitting data over Ethernet, the **Ethernet frame** is primarily responsible for the correct rulemaking and successful transmission of data packets. Essentially, **data** sent over Ethernet is **carried by the frame**. An Ethernet frame is between 64 bytes and 1,518 bytes big, depending on the size of the data to be transported.

The Ethernet frame structure is defined in the **IEEE 802.3** standard. Here is a graphical representation of an Ethernet frame and a description of each field in the frame:

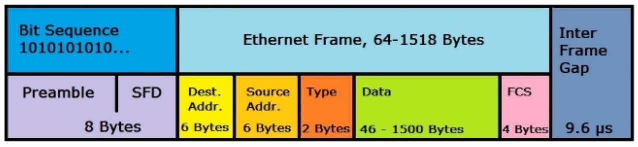


The Ethernet frame

- **PREAMBLE** – Ethernet frame starts with 7-Bytes Preamble. This is a pattern of alternative 0's and 1's which indicates the starting of the frame and allows the sender and receiver to establish bit synchronization. Initially, **PRE (Preamble)** was introduced to allow for the loss of a few bits due to signal delays. But today's high-speed Ethernet doesn't need Preamble to protect the frame bits. **PRE (Preamble)** indicates the receiver that frame is coming and allow the receiver to lock onto the data stream before the actual frame begins.
- **Start of frame delimiter (SFD)** – This is a 1-Byte field which is always set to 10101011. SFD indicates that upcoming bits are starting of the frame, which is the destination address. Sometimes SFD is considered the part of PRE, this is the reason Preamble is described as 8 Bytes in many places. The SFD warns station or stations that this is the last chance for synchronization.
- **Destination MAC Address** – This is a 6-Byte field which contains the MAC address of the machine for which data is destined.

Ethernet II Frames

Standard IEEE 802.3 basic frame format is discussed in detail in the preceding section. Now let's see the extended Ethernet frame (also called Ethernet II Frame) header:



The Ethernet II frame

An **Ethernet II frame** must be at least 64 bytes for collision detection to work, and can be a maximum of 1,518 bytes. The packet starts with a **preamble** that controls the synchronization between the sender and receiver and a **"Start Frame Delimiter (SFD)"** that defines the frame. Both values are bit sequences in the format "10101010 ..." in which the actual frame contains information about the source and destination addresses (MAC format), control information (in the case of Ethernet II the type field, later a length specification), followed by the transmitted data record. A **frame check sequence (FCS)** is an error-detecting code that closes the frame (except for the preamble and SFD). The packet is completed by an **"InterFrame Gap"**, which defines a 9.6 µs transmission pause.

Ethernet II uses the classic frame structure with a type field ("Type") which defines various protocols of the network layer. In the OSI model, the network layer is important for connecting and providing network addresses. The type field was replaced by a length specification in later frame formats.

Tip:

- In the type field, Ethernet II determines which switching protocols are used. This is important for segmenting the data stream and preventing data congestion.