Data communication occurs between a sender endpoint and one or more receiver endpoints on a network; therefore, learning the details of endpoint identification is essential to understanding the subject. At both the data-link layer and network layer, an endpoint is identified by an address that is unique within a communication domain. The data-link layer address is commonly referred to as the Media Access Control (MAC) address, because both the de-facto wired and wireless LAN protocol suites – Ethernet and Wi-Fi – use MAC addresses to identify each endpoint.

The network layer address is commonly referred to as an IP address, because the Internet Protocol (IP) is the de-facto network layer protocol on most networks. More specifically, the most widespread network layer protocol in use today is IP version 4, often written as "IPv4". IP version 6, often written as "IPv6", is designed to be the replacement for IPv4 and has been available for some time; however, IPv6 has yet to replace IPv4 at a global level.

LAB OBJECTIVES

The objectives of this lab are:

- to give you a working knowledge of representing MAC addresses in binary and enhanced hexadecimal notations, and converting between the two.
- to give you a working knowledge of extracting the Organizational Unique Identifier (OUI) from a MAC address.
- to give you a working knowledge of representing IPv4 addresses in binary and dotted decimal notation.

LAB SUBMISSION

Use the submission template provided in the assignment inbox to perform the steps requested by this lab. Return to the assignment inbox to submit your lab.

Copyright 2013, 2014 Boston University. All Rights Reserved.

Section One - MAC Address Representation

Overview

In this section, we learn how to represent Media Access Control (MAC) addresses at the data-link layer.

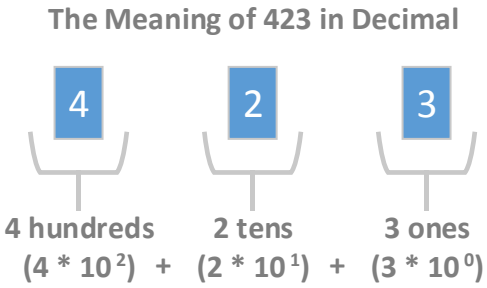
Computers and devices connect to local area networks with network adapters, which come in many forms. One common form is a card that can be internally installed onto the device or computer. This card is termed a Network Interface Controller or Network Interface Card (NIC). Another common form is as circuitry of the motherboard, so that the computer or device is network enabled without the need to install an additional card. Some adapters are external, including those that connect via a USB port. External adapters allow computers and devices to gain network connectivity without the need to internally install a NIC. Due to the ever increasing demand for virtual machines, virtual network adapters, which are adapters that exist in software only, are becoming increasingly common.

While network adapters come in many forms, it is most important that you understand that network adapters enable a computer or device to access a local area network, and that all network adapters are preconfigured with a MAC address. Manufacturers assign the address to physical adapters, while software assigns the MAC for virtual adapters. The MAC address for many physical adapters cannot be modified, though some physical adapters' addresses can be modified via software. Virtual adapters' addresses allow modification of the MAC address via software.

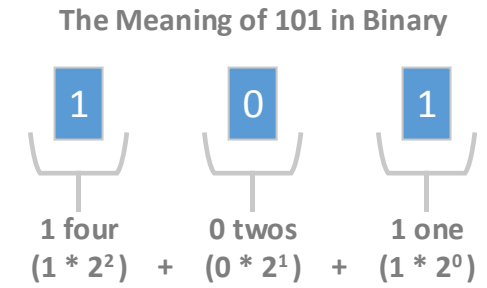
Though a MAC address consists of many details and nuances, its quintessence is a series of bits used to uniquely identify a communication endpoint. The address assigned to one network adapter will be different than all other addresses within the same communication domain. Therefore, as long as a sending node knows the MAC address of its intended recipient, the node can send a message specifically to the one recipient. A node sending to another node using a MAC address can be likened to your experience of mailing a letter. You know to whom you would like to send the letter, and so you write the correct address on the envelope and mail it. The address you put is different from all other addresses, so you know your letter will arrive at the correct place.

Before we look at more details of the 48-bit MAC addresses, let us first examine our numbering system and how this relates to binary numbers. The numbering system we are familiar with is more formally termed a decimal, or base ten, numbering system. We use ten symbols – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – to represent numbers zero through nine. For numbers higher than nine, we concatenate these numeric symbols with the recognition that each concatenated digit is multiplied by a successive power of 10. For example, the symbol "3" literally means "three", but the concatenated symbols "23" do not literally mean "two three". Rather, they mean "two times ten plus three", which is twenty-

three. Similarly, “423” does not mean “four two three”, but instead “four times one-hundred plus two times ten plus three”, which is four-hundred-twenty-three. The meaning of 423 in decimal is illustrated in the following figure.



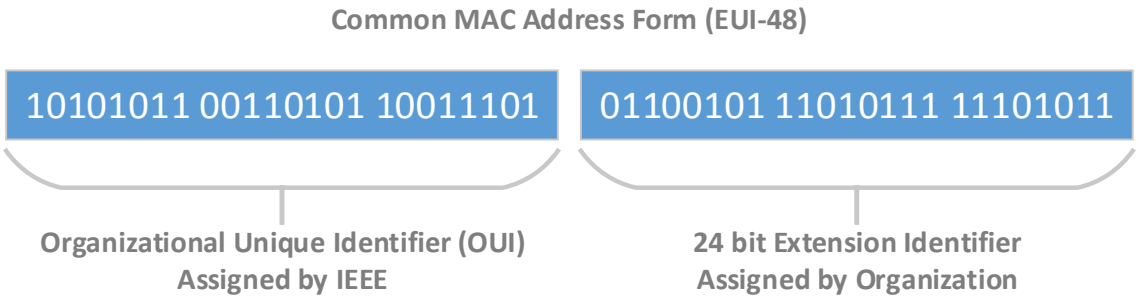
The binary numbering system is a base two numbering system. Two symbols are used – 0 and 1 – to represent numbers zero and one. For numbers higher than one, the symbols are concatenated with the recognition that each concatenated binary digit (termed a “bit”) is multiplied by a successive power of two. For example, the binary number 101 does not literally mean “one zero one”, but means “one times four plus zero times two plus one”. The meaning of 101 in binary is illustrated in the following figure.



It is important to understand that digital computers and computer networks use binary numbers to represent all data, including the data-link layer addresses for network adapters.

Network adapters are most commonly assigned 48-bit identifiers defined as EUI-48 identifiers by the IEEE Standards Association (EUI stands for Extended Unique Identifier). The first 24 bits of an EUI-48 identifier comprise the Organizational Unique Identifier (OUI). An organization that manufactures network adapters obtains this OUI from the IEEE Standards Association, which ensures that each OUI is unique. To create the MAC address for a network adapter, the organization combines their OUI with a unique 24-bit number to create the 48-bit MAC address. Thus, the uniqueness of each OUI when compared to all other OUIs, combined with the uniqueness of the 24-bit extension assigned by the organization, ensures that any MAC address assigned by a manufacturer

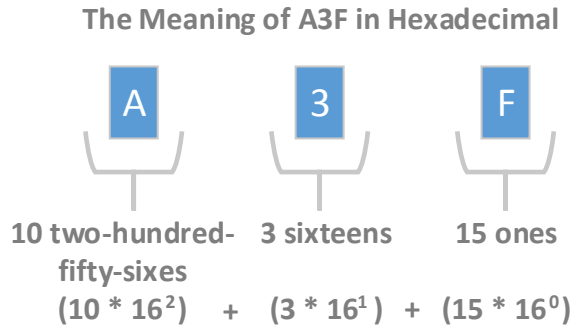
is unique throughout the world. The common MAC address form is illustrated in the following figure.



It is worth noting that there is some movement toward manufacturers assigning a 64-bit identifier, termed EUI-64, to their network adapters. However, the most common assignment today is that of an EUI-48 identifier. IPv6 supports converting EUI-48 identifiers into EUI-64 identifiers, but that is beyond the scope of this lab.

To simplify their representation, MAC addresses are commonly represented using hexadecimal notation in networking texts and tools. This is because it is difficult for human beings to work with 48-bit addresses when they are represented as a series of 48 0s or 1s.

Hexadecimal is a base sixteen numbering system. Sixteen symbols are used – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F – to represent numbers zero through fifteen. As you might now predict, when we concatenate these symbols, the implicit meaning is that each symbol is multiplied by a successive power of 16. For example, the symbol "F" literally means "fifteen", but the symbols "A3F" means "ten times two-hundred-fifty-six plus three times sixteen plus fifteen". The meaning of "A3F" is illustrated in the following figure.



MAC addresses are represented using hexadecimal digits, where pairs of hexadecimal digits are separated by a colon or a dash. For example the address 101010110011010110011101011001011101011111101011 would be represented as AB:35:9D:65:D7:EB or AB-35-9D-65-D7-EB.

Do not worry if the MAC address representation seems confusing. The focus of this section in the lab is on EUI-48 identifier representation of MAC addresses, and how to interpret and understand these addresses in binary and hexadecimal.

Steps

1. Let us start with just a portion of a MAC address, for the purpose of learning. Imagine that the first eight bits of a MAC address are 10110010. What is this number in decimal? In order to determine this, we simply fill in the following table, which has the powers of two filled in.

Power of 2		128	64	32	16	8	4	2	1
Bit									
Cumulative Amount									

All we need to do is to fill in the bit cells from our given binary number, then add the corresponding power of 2 where the bit is 1. When the bit is 0, that power of 2 is not added to the amount.

To start by entering 0 in the leftmost "Cumulative Amount" cell, then we enter a 1 in the leftmost "Bit" cell, which then adds 128 to the cumulative amount, like so:

Power of 2		128	64	32	16	8	4	2	1
Bit		1							
Cumulative Amount	0	128							

We then continue following this methodology with rest of the table cells. When doing so, we arrive at the final number 178, so we know that the binary number 10110010 is 178 in decimal.

Power of 2		128	64	32	16	8	4	2	1
Bit		1	0	1	1	0	0	1	0
Cumulative Amount	0	128	128	160	176	176	176	178	178

2. Now, give it a try by converting the binary number 01110110 to decimal by filling in the same table in step 1.

3. Now that we know how to convert the binary number to decimal, let us consider how to do the reverse. If you are given a decimal number, such as 215, how would you go

about converting it? We can make use of the similar table below, which helps us by illustrating the powers of 2, as well as the amount remaining we have left to represent as we add bits to the representation.

Power of 2		128	64	32	16	8	4	2	1
Bit									
Amount Remaining									

We only use a 1 bit if the corresponding power of 2 is less than or equal to the amount remaining, and a 0 bit otherwise.

We start by entering the full number, 215, into the leftmost "Amount Remaining" cell.

Power of 2		128	64	32	16	8	4	2	1
Bit									
Amount Remaining	215								

We now consider whether 128 is less than 215, which it is, and so we enter a 1 into the "Bit" cell under 128. And because we entered a 1, we subtract that amount from the previous amount remaining, like so:

Power of 2		128	64	32	16	8	4	2	1
Bit		1							
Amount Remaining	215	87							

We then continue this methodology through the rest of the table. After filling in the table, we find that 215 in decimal is 11010111 in binary.

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	1	0	1	1	1
Amount Remaining	215	87	23	23	7	7	3	1	0

4. Now, you give it a try by converting the decimal number 131 to binary by filling in the table.

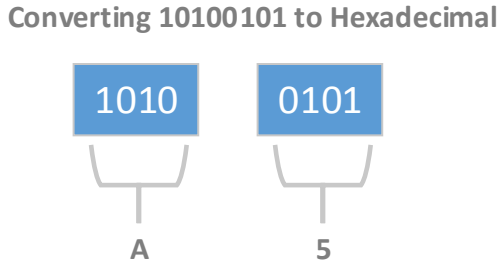
5. Now that you've had experience converting from binary to decimal and vice-versa, the only other thing you need to learn before working with full 48-bit addresses is how to convert into hexadecimal. We could choose to convert directly from decimal

to hexadecimal by using a table similar to those above, with powers of 16. However, because the powers of 16 become very large very quickly, and human beings oftentimes find it challenging to work with very large numbers, it is easier for us to first convert from decimal to binary, then convert from binary to hexadecimal.

To convert to binary to hexadecimal, you need a simple listing of 4-bit binary numbers and their hexadecimal equivalents, as follows. You will find it beneficial if you are able to eventually memorize these conversions.

0000 = 0	0001 = 1	0010 = 2	0011 = 3	0100 = 4
0101 = 5	0110 = 6	0111 = 7	1000 = 8	1001 = 9
1010 = A	1011 = B	1100 = C	1101 = D	1110 = E
1111 = F				

If you are given a binary number, such as 10100101, and are asked to convert it to hexadecimal, all you need to do is break the number into groups of 4 bits, then use the table above to obtain the equivalent hexadecimal. This process is illustrated in the figure below.



The corresponding hexadecimal number is A5.

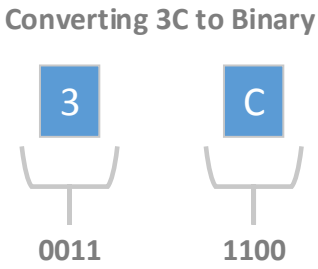
You may be curious as to why we can convert groups of binary digits directly into hexadecimal, while we cannot convert groups of decimal digits directly into binary or hexadecimal. The reason is that 16, the base of hexadecimal, is a power of 2, which is the base of binary. 10, the base of decimal, is not a power of 2, so we cannot take the same shortcut when working with decimal numbers.

6. Now it's your turn. Use the binary to hexadecimal table to convert the binary number 01101111 to hexadecimal.

7. We can use a similar lookup table to convert hexadecimal to binary, just by reversing the process. That is, given two hexadecimal digits, we locate the corresponding 4 binary bits, and concatenate them to derive the full binary number.

0 = 0000	1 = 0001	2 = 0010	3 = 0011	4 = 0100
5 = 0101	6 = 0110	7 = 0111	8 = 1000	9 = 1001
A = 1010	B = 1011	C = 1100	D = 1101	E = 1110
F = 1111				

For example, to convert the hexadecimal number 3C, we find that 3 corresponds to 0011, and that C corresponds to 1100. When we concatenate them, our full binary number is 00111100. It is really that simple! This process is illustrated in the following figure.



8. Try it yourself now by converting the hexadecimal number A2 into binary.

9. Let us review what you have learned thus far. You are now able to convert decimal to binary, binary to decimal, binary to hexadecimal, and hexadecimal to binary. You now have the tools you need to work with full MAC address representations. Congratulations!

   Let us now look at a full 48-bit MAC address in the enhanced hexadecimal format commonly used in networking texts and tools.

   AC:95:19:BC:E2:9A

   What is this number in binary? To answer this, all we need to do is apply what we have already learned to each hexadecimal digit using our handy conversion table, then concatenate each group of 4 bits to arrive at the full binary number.

0 = 0000	1 = 0001	2 = 0010	3 = 0011	4 = 0100
5 = 0101	6 = 0110	7 = 0111	8 = 1000	9 = 1001
A = 1010	B = 1011	C = 1100	D = 1101	E = 1110
F = 1111				

   First, we start by filling in the table below.

A	C	9	5	1	9	B	C	E	2	9	A
1010	1100	1001	0101	0001	1001	1011	1100	1110	0010	1001	1010

Now, we simply concatenate all groups of bits together.

10101100100101010001100110111100111000101001101 0

Wow! Look at the size of that number. As you can now see, it is much easier for us as human beings to work with the enhanced hexadecimal representation than the binary number directly. However, because computers and networks deal directly with the binary numbers, there are times when we need to be able to convert the MAC into binary to debug a problem with a real-world network.

For readability purposes, the lab instructions will list most binary numbers in groups of 8. Thus:

10101100100101010001100110111100111000101001101 0

will be represented as:

10101100 10010101 00011001 10111100 11100010 10011010

It is also requested that you represent straight binary numbers in groups of 8, when completing the labs, as illustrated above, to aid in the readability of your answers. This convention is for readability only. Please keep in mind that computers and devices use the raw bits and do not use characters or spaces in their encoding of binary numbers.

10. Now, give it a try yourself with this full 48-bit address, by converting each hexadecimal digit to 4 bits, and then concatenating all the groups of bits.

    D9:87:3A:BE:F2:15

11. Recall that the first 24-bits in an EUI-48 identifier are normally the Organizational Unique Identifier (OUI) assigned by the IEEE organization. What would the OUI be in the address given in step 9 (AC:95:19:BC:E2:9A)? Simple! It would be AC:95:19 in the enhanced hexadecimal representation, or 10101100 10010101 00011001 in binary.

12. What would the OUI be for the address given in step 10 (D9:87:3A:BE:F2:15) in both the enhanced hexadecimal representation and in binary? List them both out.

13. With all you have learned, you are now able perform more complete exercises on 48-bit MAC addresses, which you may well need to do when using networking tools to solve real-world problems. More importantly, you need this applied knowledge when working conceptually with networks. It is best to have ample practice when doing so. For the following MAC address:

A9:BC:12:E9:6C:DF

convert it into its binary representation using the table introduced in step 9, and give its 24-bit OUI in enhanced hexadecimal and in binary.

14. Given the following 48-bit address represented in binary, convert it to enhanced hexadecimal representation, and give its 24-bit OUI in enhanced hexadecimal and binary representation. Convert the number by first grouping the bits into groups of four, and then using the binary to hexadecimal lookup table on each group.

    00100100 11010111 10011011 01100101 00000010 00111111

    We are done learning how to work with the most common type of MAC address. Congratulations on your progress!

Section Two - IPv4 Address Representation

Overview

In this section, we will learn how to represent IPv4 addresses at the network layer.

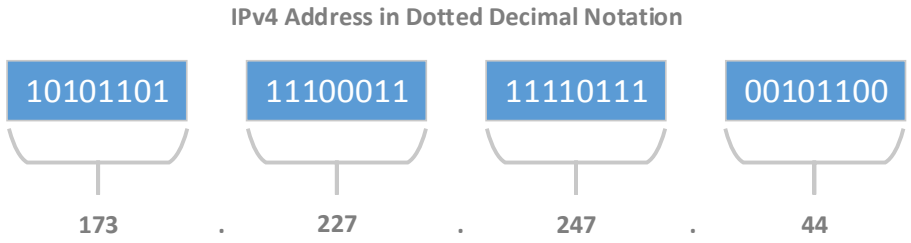
We learned previously that computers and devices connect to LANs with network adapters, and that each adapter is assigned its own MAC address. When a computer needs to send a frame to another computer on its LAN, it finds out the MAC address for the adapter on the computer, then addresses the frame to that MAC address. One requirement imposed by Ethernet and Wi-Fi is that the intended recipient must be directly accessible by the sender. That is, the sender will simply address a frame to the receiver under the assumption that the frame has the capacity to reach the recipient directly.

If all computers were connected directly to each other, identifying senders and receivers solely by their MAC address might suffice. However, computers and devices today need to communicate with other computers across cities, states, and even nations, and this type of communication requires functionality beyond that which Ethernet and Wi-Fi provide. One major role of the network layer is to make provision for this type of inter-network communication. IPv4, the current de-facto network layer protocol, makes this provision through a formulaic means of packet forwarding across networks. One integral part of IP's packet forwarding formula is the assignment of a unique address, termed an IP address, to each communication endpoint. In practical terms, this means that each network adapter is assigned an IP address, in addition to a MAC address.

It may be confusing to you as to why a network adapter would be assigned two different, unique addresses. After all, isn't one unique address sufficient? Within the five layer Internet model, the answer is no. The reason lies in the way that layers work. Each layer on the sender communicates only with the same layer on the receiver. That is, the data-link layer on the sender communicates with the data-link layer on the receiver, and the network layer on the sender communicates with the network layer on the receiver. Any address used at the data-link layer is not used by the network layer, and any address used at the network layer is not used by the data-link layer. Otherwise, they would not be two layers, but one. So, we conclude that the address used at the data-link layer is used to uniquely identify senders and receivers at the data-link layer only, and the address used at the network layer is used to uniquely identify senders and receivers at the network layer only.

There is much to learn about the significance and use of the IPv4 address, but for now, we will focus on learning its format and representation. An IPv4 address is 32 bits in length, for example, 10101101 11100011 11110111 00101100. As you might expect, as

human beings we find it difficult to practically work with binary numbers of this length, and so have adopted a standard representation easier for us to understand. Simply put, we divide the number into 4 sequences of 8-bits, then represent each 8-bit sequence as a decimal number. Each 8-bit sequence is termed an *octet* in this context, and the decimal number for each octet is divided by a period. For example, the IPv4 address given in this paragraph, 10101101 11100011 11110111 00101100, is represented as 173.227.247.44. This notation is referred to as *dotted decimal* notation, which is a presentation format whereby each of two or more decimal numbers are separated with a period. The example IPv4 address is illustrated in the following figure.



Note that although the notation for representing *exactly four* decimal numbers, each separated by a period, is *dotted-quad* notation, in practice almost all writers describing IPv4 addresses use the more generic term. To follow suit, the phrase "dotted decimal" will be used throughout this section.

Steps

15. First, let us practice converting an IPv4 address in dotted decimal notation to binary. You have previously learned how to convert from decimal to binary, so the general methodology is nothing new to you. All we need to do is convert each of the four decimal numbers to binary, using the handy conversion tables below, then concatenate the results. For example, if we have the IPv4 address 192.168.5.33, we convert each number in turn like so.

192 ➔ 11000000

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	0	0	0	0	0
Amount Remaining	192	64	0	0	0	0	0	0	0

168 ➔ 10101000

Power of 2		128	64	32	16	8	4	2	1
Bit		1	0	1	0	1	0	0	0
Amount Remaining	168	40	40	8	8	0	0	0	0

5 ➔ 00000101

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	0	0	0	1	0	1
Amount Remaining	5	5	5	5	5	5	1	1	0

33 ➔ 00100001

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	1	0	0	0	0	1
Amount Remaining	33	33	33	1	1	1	1	1	0

Each number separately thus evaluates to 11000000.10101000.00000101.00100001. When we remove the periods, we can say that the IPv4 address of 192.168.5.33 is 11000000 10101000 00000101 00100001 as a raw binary number.

You might wonder why we bother converting addresses in dotted decimal notation to binary. After all, the dotted decimal notation is easier for us to work with. The reason is that computers and network adapters do not view the address in dotted decimal notation, and operate on the raw 32-bit binary number. We will learn some precise ways that computers use to operate on IPv4 addresses, and it is critical that we view and operate on the address in its raw form when doing so.

16. Now, you give it a shot by converting the following IPv4 address in dotted decimal notation to binary. Use the table provided in the submission template for each octet.

    137.254.120.50

17. We also need some practice in converting IPv4 addresses in raw 32-bit format into dotted decimal notation. We simply reverse the process to do so. We separate the 32-bit number into 4 octets, convert each octet to a decimal number, then separate each decimal number with a period.

    For example, if we are given the IPv4 address:

    11010000 01000000 01111001 10100001

    we convert each octet to decimal using the conversion tables below.

11010000 ➔ 208

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	1	0	0	0	0
Cumulative Amount	0	128	192	192	208	208	208	208	208

01000000 ➔ 64

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	0	0	0	0	0	0
Cumulative Amount	0	0	64	64	64	64	64	64	64

01111001 ➔ 121

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	1	1	1	0	0	1
Cumulative Amount	0	0	64	96	112	120	120	120	121

10100001 ➔ 161

Power of 2		128	64	32	16	8	4	2	1
Bit		1	0	1	0	0	0	0	1
Cumulative Amount	0	128	128	160	160	160	160	160	161

Now that we know the decimal number for each octet, we simply separate each with a period to arrive out our final dotted decimal notation: 208.64.121.161. So the IPv4 address 11010000010000000111100110100001 is 208.64.121.161 in dotted decimal notation.

18. Now, you give it a try with the following IPv4 address displayed in its raw 32-bit format. Convert it to its dotted decimal notation using the same conversion tables as in step 17.

    11010001 00001111 00001101 10000110

Congratulations on your progress! You have learned how to convert to and from binary, decimal, and hexadecimal, in order convert into and out of important MAC and IPv4 address representations. This is an important step on your journey to understand and work intelligently with modern networks!