While IPv4 is still by far the language of the Internet and is used significantly more than IPv6, some organizations have begun to transition to IPv6. There is sufficient vendor and standards support to make this transition possible, and organizations recognize that as the intended replacement for IPv4, IPv6 offers many benefits. Therefore it is important for you to understand some fundamental IPv6 concepts so that you are prepared to work with modern networks that have transitioned, or will soon transition, to IPv6.

You may legitimately ask why IPv6 is needed at all. After all, if IPv4 is working fine for many organizations, why not just keep using IPv4? We are in the final stages of IPv4 address exhaustion, because IPv4 addresses are only 32 bits in length, and because IPv4 addresses have not been allocated optimally or equitably to all organizations worldwide. So although organizations have managed to operate worldwide using IPv4 up until now, the unprecedented growth of the Internet will not allow for an IPv4 only world much longer. There has been and continues to be an explosion of the use of personal devices such as smart phones and tablets, and each of these need an IP address. Previously underdeveloped markets worldwide are beginning to demand millions of new IP addresses. The now digital world will soon need a different protocol, and IPv6 is the intended solution.

Although IPv6 is the intended replacement for IPv4, it is likely that IPv4 will be used for decades to come. We will not likely experience a light switch effect of “turning off” IPv4 and “turning on” IPv6. One reason for this is because organizations use port address translation (PAT) to map hundreds or thousands of private addresses to a single public address, thereby delaying the often touted IPv4 address exhaustion due to IPv4’s 32-bit address. Without PAT, we would have exhausted all public IPv4 addresses a long time ago and organizations would have been forced to switch to a replacement sooner, but as of today many organizations are operating just fine using IPv4. Therefore many organizations will continue to use IPv4, possibly for a long time to come. Some hardware and software is very long lived, even unto decades, and in many cases it may be cost prohibitive or impossible to upgrade such legacy systems to IPv6, even if the organization as a whole has migrated to IPv6. So you can rest assured that your IPv4 *and* IPv6 knowledge and skills will be useful for decades to come.

One of the many powerful features of IPv6 is that it has been designed to fully interoperate with IPv4, so that organizations that use IPv4 exclusively can communicate with organizations that use IPv6 exclusively, and vice-versa. Also, organizations can use both IPv4 and IPv6 simultaneously, making the transition to IPv6 easier.

LAB OBJECTIVES

The objectives of this lab are:

- to learn how to represent IPv6 addresses in canonical text representation.
- to learn how to remove leading zeros from IPv6 addresses in canonical text representation.
- to learn how to zero compress IPv6 addresses in canonical text representation.
- to learn how to create and represent IPv4-mapped IPv6 addresses.

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.

Section One – IPv6 Address Representation

OVERVIEW

IPv6 addresses are 128 bits in length, and this length demands a human-readable representation that is more concise than the standard IPv4 dotted decimal notation. The accepted IPv6 representation is 8 blocks of 4 hexadecimal digits, each separated by a colon. Since each hexadecimal digit is capable of representing 4 bits, a block of 4 of them can represent 16 bits. And we need 8 blocks of these to represent the full 128 bit address.

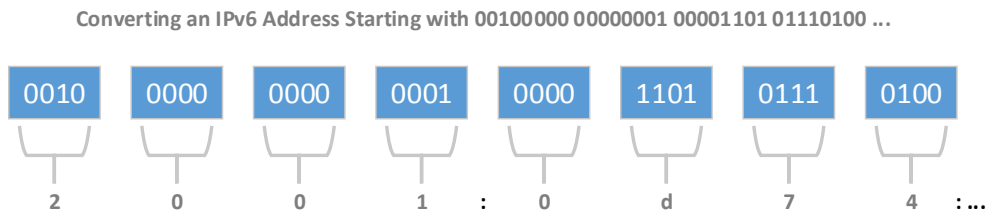
STEPS

1. Let us work through an example of IPv6 representation. Imagine that an IPv6 address in binary is 00100000 00000001 00001101 01110100 00111000 00101101 00000000 00010010 00000000 00000000 00000000 00000001 00010010 00110100 01010110 10101010. Wow! That is a long binary number! Computers have no problem with binary numbers of that length, but we would surely make mistakes if we tried to work with and perform calculations on such long binary numbers. Instead, we represent this same IPv6 address as follows:

   2001:0d74:382d:0012:0000:0001:1234:56aa

   Each hexadecimal digit represents 4 bits. For example, the first 8 bits of the address in binary are 00100000, so the first 4 bits, 0010, are represented with a 2 in hexadecimal, and the second 4 bits, 0000, are represented with a 0 in hexadecimal. For the next 8 bits, 00000001, 0000 is represented with a 0 in hexadecimal, and 0001 is represented with a 1 in hexadecimal. Thus the first 16 bits are represented as 2001 in hexadecimal. After a block of 4 hexadecimal digits, by convention, we insert a colon (:) to separate it from another block of 4 hexadecimal digits.

   The following figure illustrates this methodology applied to the first 32 bits of the same IPv6 address.



Notice that each group of 4 bits is converted to hexadecimal, and that a colon is inserted after each block of 4 hexadecimal digits. While the figure only illustrates the first 32 bits, the same methodology can be applied to all 128 bits to arrive at the human readable representation of 2001:0d74:382d:0012:0000:0001:1234:56aa.

The human readable representation shown above, with 8 sets of 4 hexadecimal digits, is defined in RFC 2373, titled "IP Version 6 Addressing Architecture", and in RFC 5952, titled "A Recommendation for IPv6 Address Text Representation". An RFC is a document that is a "Request for Comments" by the Internet Engineering Task Force (IETF), one of the organizations responsible for defining and improving many network protocols. RFC 5952 terms this representation as the "canonical text representation", and this term will be used throughout the remainder of this lab to refer to this human readable representation.

2. Convert the following IPv6 address given in binary to its canonical text representation shown in step 1.

   11010011 00000101 11111100 10101010 00000000 11000000 11100111 00111100
   01010000 11000001 10000101 00001111 00100100 11011011 10100011 01100110

3. RFCs 2373 and 5952 define two rules on how to shorten the length of canonical text representations. We will work with the first rule in this step, and the second rule in a later step. The first rule is that in each group of 4 hexadecimal digits, leading 0 digits can be omitted. For example, if a group contains the digits "0001", the group can be shortened to "1" by eliminating the leading 0 digits. If a group consists of all 0 digits, however, the rule states that one 0 must still be present. Therefore, a group that contains "0000" can be shortened to "0". Other examples include reducing "0010" to "10" and "0A9B" to "A9B". In all cases we simply eliminate the leading 0 digits, if there are any.

   In step 1, we determined that the canonical text representation for address:

   00100000 00000001 00001101 01110100 00111000 00101101 00000000 00010010
   00000000 00000000 00000000 00000001 00010010 00110100 01010110 10101010

   is 2001:0d74:382d:0012:0000:0001:1234:56aa.

   If we apply the first rule to eliminate all leading 0 digits, the leading 0 bits are eliminated, 2001:~~0~~d74:382d:~~00~~12:~~000~~0:~~000~~1:1234:56aa, and the result is thus 2001:d74:382d:12:0:1:1234:56aa.

4. Now you give the rule a try on the following address:

   00000000 10001111 11000110 00000001 00001011 00111110 11111110 00011000

11101110 01110000 00111001 11111110 01010100 11000001 00000001 11100111

Represent the address in its canonical text notation, making sure to eliminate all leadings 0 digits as described by the rule introduced in step 3.

5. RFCs 2373 and 5952 define a second rule termed "zero compression" that further reduces the size of a particular canonical text representation. In exactly one place in a canonical text representation of an address, the characters "::" can substitute for two or more consecutive groups of 4 hexadecimal digits that consist of all 0 digits.

   For example, the following address in binary notation:

   11001111 00110011 00110011 00000000 00000000 00000000 00000000 00000000
   00000000 00000000 10001100 01011100 00000000 00000000 00000000 00000000

   is represented as CF33:3300:0000:0000:0000:8C5C:0000:0000 in full canonical text representation. If we apply zero compression to the first three consecutive groups of 4 hexadecimal digits consisting of all 0s, the canonical text representation then becomes CF33:3300::8C5C:0000:0000. Let us view this zero compression graphically in the figure below.

   Notice that the three consecutive groups of all 0 digits are eliminated and are simply represented by the characters "::".

   Note that this rule can be applied to *only one* instance of consecutive groups of all 0 digits. Why? Because if two or more consecutive groups were replaced with the "::" characters, it would no longer be possible to reconstruct the full address reliably due to the introduced ambiguity. For example, the aforementioned address CF33:3300:0000:0000:0000:8C5C:0000:0000 has two different consecutive groups of all 0 digits, with the consecutive groups highlighted in blue. If we were to zero compress both of them, the result would be CF33:3300::8C5C::. Why does this result not work? Simple! In viewing this compress form, we have allowed for the following two full representations:

   - CF33:3300:0000:0000:8C5C:0000:0000:0000
   - CF33:3300:0000:0000:0000:8C5C:0000:0000

There is no way for us to determine which is correct simply by viewing the compressed form of the address, CF33:3300::8C5C::. For this reason, RFC 5952 mandates that only one consecutive group of all 0 digits can be replaced with the characters "::", because then there is no ambiguity.

What happens if there are multiple consecutive groups of all 0 digits in the address? RFC 5952 indicates the following order of precedence:

1. Zero compress the longest of the consecutive groups of all 0 digits.
2. If two or more consecutive groups of all 0 digits are of the same length (and are the longest consecutive groups), zero compress the first (leftmost).

With our example address CF33:3300:0000:0000:0000:8C5C:0000:0000, we zero compress the first of the consecutive groups of all 0 digits, because the first has 3 groups, and the second only has 2 groups. Therefore CF33:3300**::**8C5C:0000:0000 *is* the correct representation, and CF33:3300:0000:0000:0000:8C5C**::** *is not* the correct representation.

Another point worth mentioning is that a single group of all 0 digits does not qualify for zero compression. The most we can do for a single group of all 0 digits is to remove the leadings 0 digits. That means that a single group of all 0 digits, 0000, would become simply become "0", but could not be replaced with the "::" characters.

Lastly, why bother with zero compression at all? After all, how likely is it that a real address will have consecutive groups of 16-bits that are all 0s? Many special addressing IPv6 forms make use of many consecutive 0 bits. As just one example, when IPv4 addresses are embedded into IPv6 addresses, the IPv6 address begins with 80 0 bits. There are other special forms as well where long strings of 0 bits are used, so it is not uncommon to see IPv6 addresses with many consecutive groups of 16-bits that are all 0s.

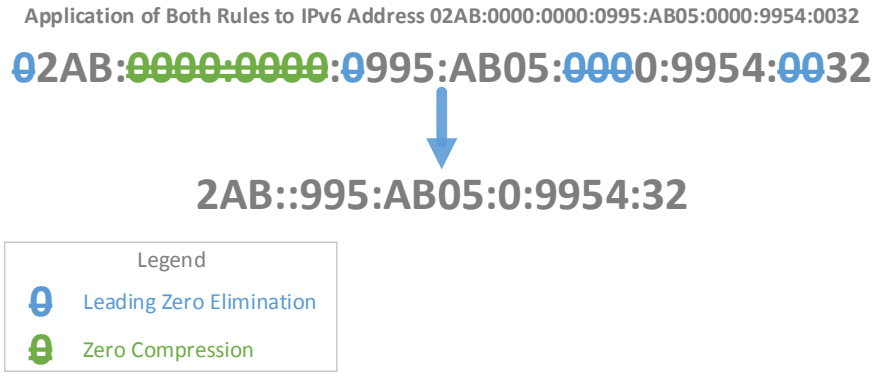
6. Now you give this second rule a try on the following address:

   11001100 01110110 00000000 00000000 00000000 00000000 00111011 11100111
   00000000 00000000 00000000 00000000 00000000 00000000 11110010 10000110

   Represent the address in its canonical text notation, making sure to zero compress the appropriate consecutive groups of all 0 digits, as described by the rule introduced in step 5.

7. The two aforementioned rules can be combined to obtain the shortest canonical text representation of an IPv6 address. For example, imagine that the full canonical text representation of an IPv6 address is 02AB:0000:0000:0995:AB05:0000:9954:0032.

We can zero compress the address, which gives us a result of 02AB::0995:AB05:0000:9954:0032, then we can remove leading 0 digits, which gives us a result of 2AB::995:AB05:0:9954:32. Application of both rules is illustrated in the following figure.



Notice that the second and third groups, which consist of all 0 digits, were compressed, and all leading 0 digits were removed.

According to RFC 5952, when representing an IPv6 address in its canonical text representation, both rules must be applied. However, because IPv6 has not yet been widely adopted as of this writing, it remains to be seen whether the compressed format or the full format will be preferred.

8. Given the following IPv6 address in binary:

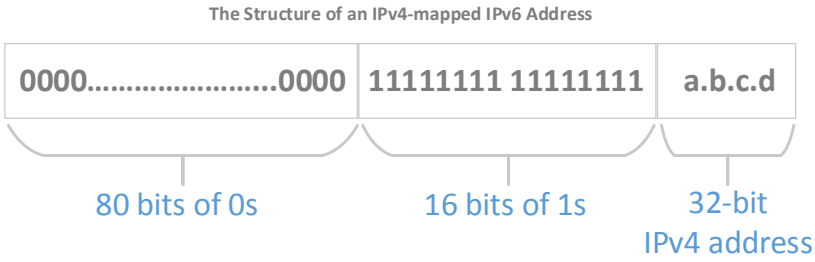
   00000000 00000000 00000000 00000000 00000000 00000000 00000000 10101011 00001001 00010000 00000000 00000000 11110001 01010110 00001101 01010101

   determine its canonical text representation, making sure to remove leading 0 digits and to apply zero compression, as illustrated in step 7.

9. A significant factor in the success of IPv6 deployment worldwide is how well it interoperates with IPv4, because the world's existing infrastructure consists of billions of devices and network links that use IPv4. It is likely that some devices will continue to use IPv4 for decades to come, and we do not want cut off communication to and from these devices. The designers of IPv6 recognized that devices that operate using IPv6 must be able to communicate with devices that exclusively use IPv4, and therefore defined a standard method of converting an IPv4 address into an IPv6 address. This type of address is termed an *IPv4-mapped IPv6 address.*

   The method of creating an IPv4-mapped IPv6 address from an existing IPv4 address is straightforward. The first 80 bits of the IPv6 address are each set to 0, the next 16

bits are each set to 1, and the IPv4 address itself is embedded in the last 32 bits. This concept is illustrated in the figure below.



Let us look at an example. Imagine that a node has the IPv4 address 177.212.15.33 and we want to create its corresponding IPv4-mapped IPv6 address.

- First, we start with 80 0 bits, which corresponds to 5 groups of 4 hexadecimal digits: 0000:0000:0000:0000:0000.
- Second, we append the 16 1 bits, which corresponds to 1 group of hexadecimal digits: 0000:0000:0000:0000:0000:FFFF.
- Third, we append the IPv4 address: 0000:0000:0000:0000:0000:FFFF:177.212.15.33.
- Fourth, we zero compress the result: ::FFFF:177.212.15.33.

Wait a minute! Is that a typo? We are not used to seeing an IPv6 address in this way, with hexadecimal mixed with dotted decimal notation. Actually, we are permitted to represent IPv4 addresses embedded in IPv6 addresses in dotted decimal notation, and it helps the IPv4 address stand out as an IPv4 address within the IPv6 address. Note that it is not illegal to represent the address in all hexadecimal, in which case the result would be ::FFFF:B1D4:F21, but it is conventional to represent the IPv4 address in dotted decimal. So our final result for the IPv4-mapped IPv6 address is ::FFFF:177.212.15.33.

10. Imagine that a computer is assigned IPv4 address 245.19.1.99. Create its corresponding IPv4-mapped IPv6 address as demonstrated in step 9, making sure to zero compress the result.