Data Communication Standards

TCP/IP Protocol Stack and the Application Layer

Introduction:

To understand data communications, you must understand data communication standards. Last week we discussed several types of networks, terminal to mainframe, client-server, the Internet and cloud computing. These networks were able to interoperate, in a multi-vendor environment, because of standards or protocols which establish predefined rules for data exchange. A chrome browser can access an Apache web server and exchange information with a Microsoft Exchange server because of common protocols, such HTTP and SMTP. In addition to interoperability, standards also facilitate competition. Each vendor wants to differentiate their product from their competitors, so they develop new features. These new features are often added to the next version of the standard. Competition also lowers prices as vendors jokey for market share.

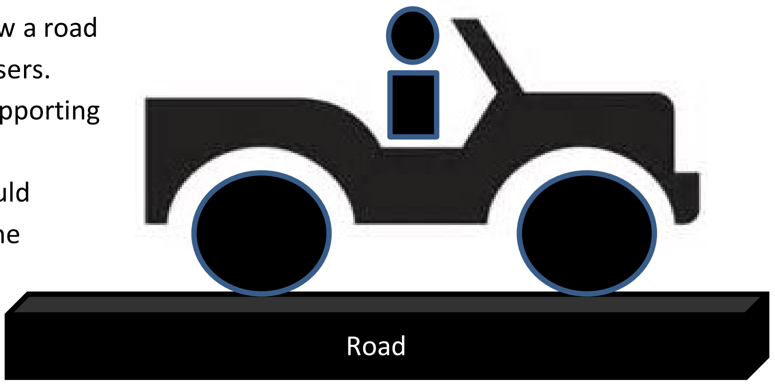
In the early days, of the client-server environment (late 1970s) there was a high degree of dissatisfaction. Different vendor's used different data formats and data exchange protocols. Businesses were unhappy because they were locked into a vendor's hardware and software when they purchased a network operating system. Users were unhappy because competitive vendors would develop new applications or features which users wanted, but were not offered by the vendor chosen by the business. Vendors too were unhappy because they had to give developers a blue print of their proprietary code for them to write applications, making them vulnerable to intellectual theft. To overcome these problems, the International Organization for Standardization (ISO) created a task force to develop an open system architecture. The project was called Open System Inter-Connection (OSI). TCP/IP was well established with a 4 layer protocol stack when the task force begin; it was expected that the OSI would replace TCP/IP. Gerald Cole in 1990 wrote:

> …we can expect to see continued usage of other open products such as the *TCP/IP protocol suite* (Transmission Control Protocol/Internet Protocol), which provides the basic needs of networking. Use of the official OSI protocols in data communications and networking will continue to gain popularity, and the OSI will continue to be utilized both in the development of OSI protocols and as a gauge for comparison with other protocols.[i]

The task force finished its work in 1993. The development of the World Wide Web in 1991 led to an exponential use of the Internet by businesses. In addition, advances in network speed and reliability made the OSI model unnecessarily complex. Consequently, the OSI architecture never replaced TCP/IP; the latter remains the standard for interconnecting networks.

Layered Protocol Architecture

The TCP/IP protocol stack today is a hybrid 5 layer model.[ii] In a layered architecture, each layer operates independent of the other layers. The layer below provides a service to the layer above and within each layer is a protocol specification.



For example, when driving your car from point A to point B, the road provides a service for your tires. The protocol specification would define how a road should be built to support the number of users. The tires provide a service for the car by supporting the weight of the care and moving it to the destination. The protocol specification would define the circumference of the tires and the density and thickness of the rubber to support the weight of the car. Lastly, the car provides a service for the driver. The car protocol would specify the frame format of the car such as size, shape and colour, and how to convert actions of the driver, such as stepping on the brake pedal to slow down, or turning the steering wheel. Lastly, the driver would include protocol specifications on using the car, high performance driving, city-driving, leisurely driving and the rules of the road. The driver, for example, wants to go in a different direction, so he/she turns the steering wheel. The steering wheel is connected to the car's tie-rod which rotates and turns the front axle. The change in direction has no effect on the job of the tires. They continue to roll and move the car regardless of the direction or road condition. This layered and independent architecture makes programming easier and modular. If you wanted to build a city-driving user-map, you don't need to know how to build a road or how to build a car. The lower layers, acting independently, provide all of the essential services for the driver to reach his/her destination.

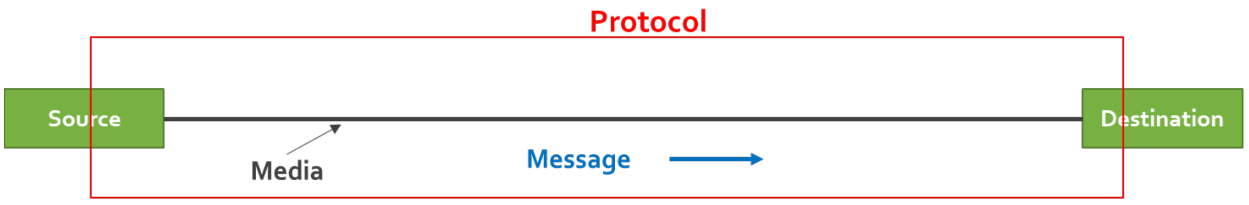
One last point, the driver must determine "how" to get to point B from point A. For example, if the driver wanted to take a leisurely drive, he/she would hop into the care, begin driving, and not really caring if he/she arrives at the destination. This is analogous to sending an email with UDP (User Datagram Protocol) which is not connection oriented and does not guarantee delivery-unreliable connection. When a driver wants to guarantee arrival at their destination, he/she will use a map or GPS which provides an established route to the destination. This is analogous to using TCP (Transport Control Protocol) to retrieve a web page which is connection-oriented establishing a route from client to server, and guaranteeing delivery. We will speak more about this later.

Layers & Protocol - Data Communication

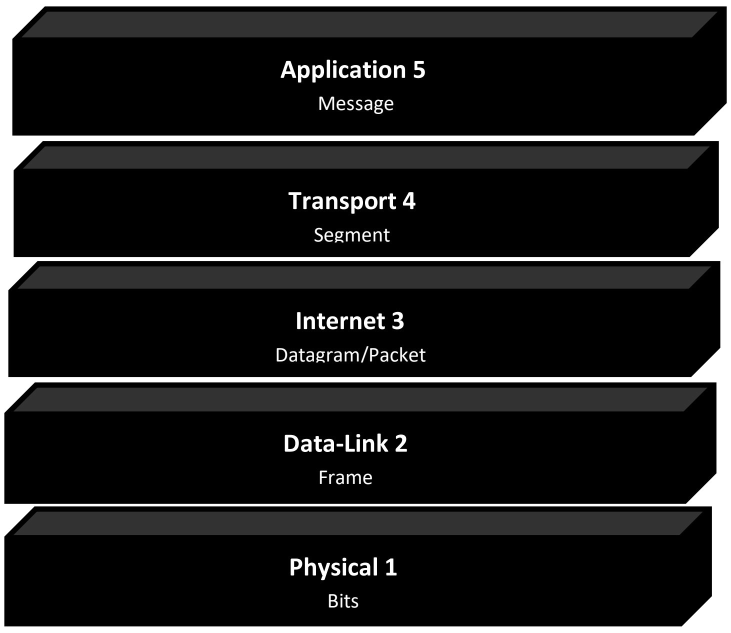
The components involved in data communication are: Source, Destination, Media, Protocol and the message exchanged between the source and the destination.

The rules that allow two entities (source, destination) to exchange some message (text, audio, video) using a physical quantity (medium-wired/wireless) is known as protocol. Protocols provide the common standardization and flexible interconnection of communication among different platforms and different kinds of computing equipment (interoperability) around the globe and locally. It is needed for establishing a connection, identification of the source and destination, transmission of data, acknowledgement of reception of data and termination of connection between the entities involved in data communication operation

Layer is a specific piece of software following some pre-defined rules of functioning in data communications. Unique functions or process are designated to different layers in a communication protocol and layers in conjunction provide the efficient communication. Layers are implemented at the source and destination and any piece of hardware which provides interfacing between the two. It is the task of the layer is to ensure that data is physically delivered to the destination without any errors.



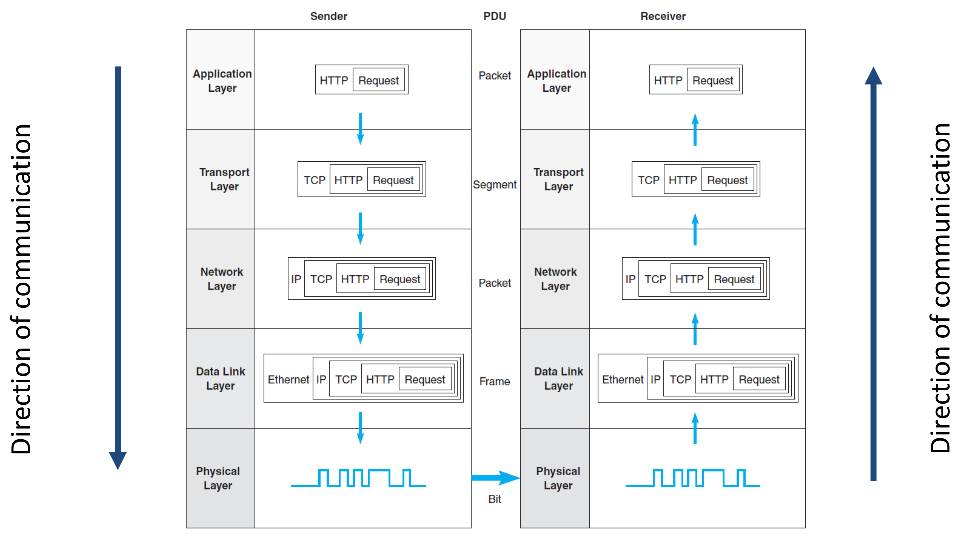
TCP/IP Protocol Architecture: In brief



The top layer of the TCP/IP protocol stack is the application layer (the layers are numbered from the bottom, the top layer is layer 5). The application layer creates the original message and determines the protocol to deliver the message. The main job of the application layer is to convert the message into a digital format. The Transport layer (layer 4) accepts the digital format as data and adds a header to the message which contains the source and destination port numbers. If TCP is the protocol, the Transport layer will add a sequence and acknowledgement number to ensure reliable delivery. If the receiving computer can't accept the speed of the transmission, the transport layer on the receiving computer will send a message to the transport layer on the sending computer to "slow down". This layer controls flow-control of the transmission.

If the message is too large to be placed on the network, the Transport will fragment the message, dividing it into smaller pieces and adding a header to each part. The transport layer on the receiving computer will reassemble the pieces back into the original message. The message is now "encapsulated" in a TCP wrapper and is called a "Segment". The segment is now passed down to the Internet layer (layer 3). It sees the message and transport header as data, and adds a header containing the source and destination IP addresses. This layer is responsible for providing routing information for packets from point A to point B, moving between switched networks. The packet is now encapsulated inside an IP wrapper and is called a "datagram/packet". The datagram/packet is now passed to the Data-Link layer. This layer adds both a header and a trailer and the datagram/packet is encapsulated in an Ethernet wrapper called a "frame". The header contains the Media Access Control address used to deliver the frame on single LAN networks. The trailer is a Frame Check Sequence number to determine if any of the bits have been corrupted in transit. The Data-Link layer can support all LAN and WAN technologies and ensures that the frame will be compatible with the network delivering it. Lastly, the Physical layer, converts the frame into bits using an encoding scheme for transiting the signals as zeros and ones using voltage or radio waves.

Message Transmission using Layers

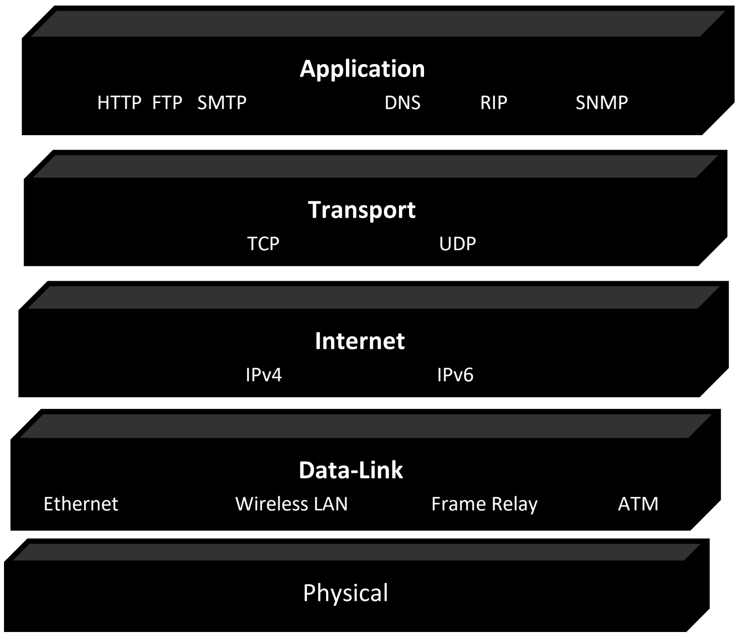


Application Layer

As previously mentioned, the Application layer defines the protocols to exchange data and specifies how applications can access the services of the other layers using sockets. A socket is an IP address and a port number, separated by a colon, such as **137.234.56.15:25.** The socket is used by the local process to make connections to destination hosts and exchange data. Different applications are assigned different port numbers; for example, incoming requests to a web server use port 80. There are a total of 65,535 ports on a PC. Ports 1-1023 are called **well-known ports** and are controlled by the Internet Corporation for Assigned Names and Numbers (ICAANN). Ports 1024-49151 are called registered ports which are registered with ICANN and reserved for specific application. Ports 49152-65535 are called dynamic or private ports. These ports can be used by programmers for all unregistered applications. The Application layer contains many protocols, and more are always being developed.

The most widely known Application layer protocols help users exchange information:

- The Hypertext Transfer Protocol (HTTP) transfers files on the World Wide Web – port 80
- The File Transfer Protocol (FTP) transfers individual files to and from servers –ports 20 and 21
- The Simple Mail Transfer Protocol (SMTP) transfers mail messages and attachments- port 25


Application
HTTP FTP SMTP DNS RIP SNMP
Transport
TCP UDP
Internet
IPv4 IPv6
Data-Link
Ethernet Wireless LAN Frame Relay ATM
Physical

Additionally, the following Application layer protocols help manage TCP/IP networks:

- The Domain Name System (DNS) protocol resolves a host name, such as senecacollege.ca, to an IP address and copies name information between DNS servers –port 53
- The Routing Information Protocol (RIP) is a protocol that routers use to exchange routing information on an IP network –port 520
- The Simple Network Management Protocol (SNMP) collects and exchanges network management information for centralized administration of network devices such as routers, bridges, and servers - port 161

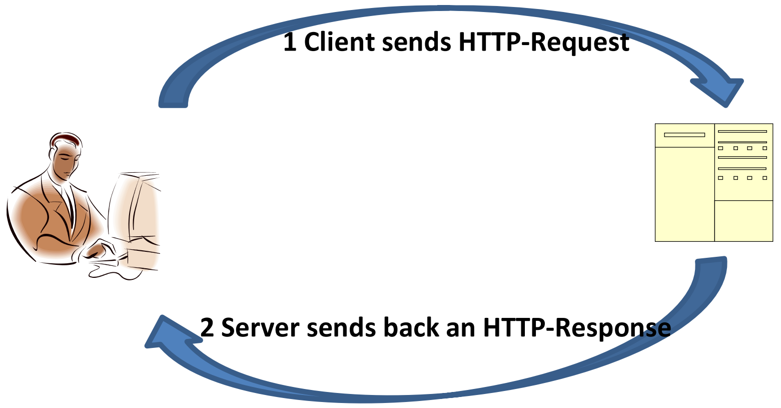
Protocol Specification

The main job of the application layer is to create the message, whether it is an email, or an HTML document. To create the message the application follows the protocol specifications so that the lower layers can provide the required services. In order to exchange information, the message must follow the following rules:

- Message Sequence – Who initiates the communication?
- Message Type – What task or command is to be performed?
- Message Syntax – What is the meaning of the message?
- Type of Connection – What type of connection is needed by the protocol, reliable or unreliable?

Message Sequence

In baking a cake, if you don't follow the recipe in sequence, the cake may not turn out very well. In the same way, network standards must govern message sequence: Who initiates the conversation? For example, the Hypertext Transfer Protocol has a simple message order. The client sends an HTTP request message to the server and the server returns an HTTP-response message which answers the client request, or sends an error message. The server can not initiate a conversation with the client.

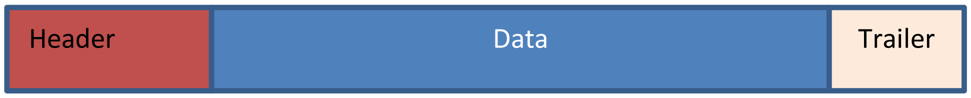

1 Client sends HTTP-Request
2 Server sends back an HTTP-Response

Message Type

When humans talk, it is usually for a purpose: What time is it? Where are the washrooms? What do you take in your coffee? There is no limitation on the questions we can ask and the order in which we ask them. Humans are intelligent, software protocols are not. The type of message must be clearly defined and limited to one specific task. For example, the most common message types of HTTP are the GET and POST messages; GET commands the server to return a web page, and POST uploads a file, usually containing online form data, to the webserver for processing. The GET message directs the server to "Get the file", the POST message says "Here is the file". (there are other commands, but for this course these are the only 2 you will need to know)

Message Syntax

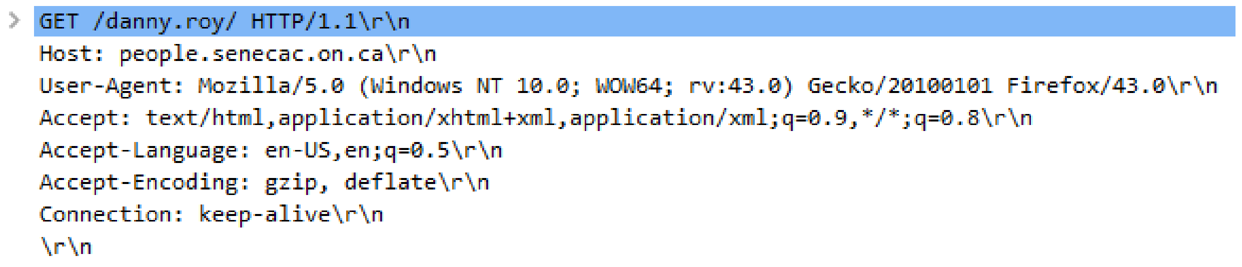
Message syntax refers to the organization of the message. In conversation, human's tend to be informal, even chaotic. Software must use very rigid and exact syntax. In general all messages have three parts: a header, data and trailer fields. The data field is the purpose of the message, created by the application layer; it contains the content to be delivered. The header is simply everything that comes before the data field and the trailer field is everything that comes after. Not all messages, however, will have a trailer field. HTTP messages, for example, always have a header field, but at times may not have a data field and trailer fields are rarely used.



The fields of the header contain information needed by the receiving computer to display the message, such a sequence numbers to ensure which packets belong to a message, the protocol used, destination and source addresses. The HTTP request message consists of 6 lines of text.

HTTP Request Message

The syntax of the HTTP request message is fairly simple. The screen shot shows the message sent to retrieve a web site called “danny.roy” on a server called people in a domain called “senecac.on.ca”.

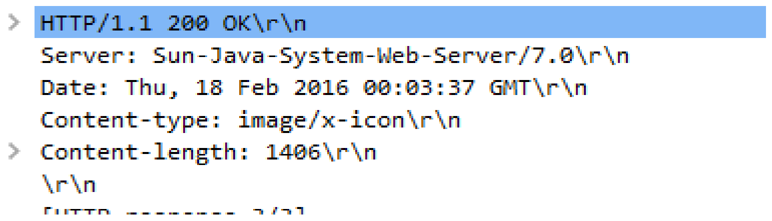


```
GET /danny.roy/ HTTP/1.1\r\n
Host: people.senecac.on.ca\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:43.0) Gecko/20100101 Firefox/43.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n
\r\n
```

- The first line specifies the GET method which requests a path to a file for retrieval and the version of HTTP used by the sender. Notice this line and all other lines end with a character return and a new line [CRLF]
- The second line specifies the host to which the request is sent people.senecac.on.ca
- The third line specifies the host browser and operating system used to send the request
- The fourth line specifies the language to be used in this case English US
- The fifth line specifies the type of compression to use, such as gzip or deflate
- The sixth line specifies the type of TCP connection in this case “keep-alive” means to keep the connection active over multiple request-response cycles. Without this, the TCP connection would end after each request-response cycle and would have to be re-established. Notice, the 2 blank lines at the end of the request; these lines are required and indicate the end of the header field. The data field is empty.

HTTP Response Message

The HTTP Response to the request contains the file requested in the data portion of the packet (not shown).



```
HTTP/1.1 200 OK\r\n
Server: Sun-Java-System-Web-Server/7.0\r\n
Date: Thu, 18 Feb 2016 00:03:37 GMT\r\n
Content-type: image/x-icon\r\n
Content-length: 1406\r\n
\r\n
```

- The first line begins with HTTP/1.1 which indicates the server has a compatible version. The 200 is a success code that the desired file is returned. The browser actually ignores this code; it is designed for humans to indicate the request was successful.
- The next line indicates the server that returned the request

- The next line gives the date and time the request was returned.
- The next heading specifies the content type which was returned, in this case an image file.
- Again, the end of the header is marked with 2 blank lines followed by the content returned in the data field (not shown). There is no trailer.

Message Encoding[iii]

Once the message has been created, the application layer converts it into a digital format. Application data may be text, image, or video. A conversion table is used to convert a byte into a unique numerical value; a process called "encoding". Each encoded value is then converted to a series of 0s and 1s. Unicode (Universal Encoding) can represent all of the written languages in the world and is the *de facto* standard for text on the web with over 86.2% of all web pages in 2016 using it.[iv]

The most common Unicode table is UTF-8 (Universal Coded Character Set + Transformation Format – 8-bit.); a variable-length encoding system using 1 to 4 "octets" (1 byte is called an octet). The first octet representing the first 128 values is identical to ASCII, making UTF-8 a superset of ASCII and safe to use with programming languages that interpret only one byte encoded characters. Some people mistakenly believe that Unicode is simply a 16-bit code with 65,536 possible characters. This is completely incorrect. UTF-8 has a character space, called code points, of 1,112,064 valid characters.

A Unicode character is represented by writing "U+" followed by a hexadecimal number. The "high-order bits" are important in Unicode. When one byte is used for ASCII values the high order bit begins with a "0". Characters above 127 in value will use more than one byte. The number of "1s" in the first byte indicates the number of octets used for encoding, followed by a "0". Each subsequent byte has a "continuation marker" of "10" at the beginning of the byte. Refer to the table below.

Bits	Decimal Range	First Code Point	Last Code Point	Bytes used	Byte1	Byte2	Byte3	Byte 4
7	0-127	U+0000	U+007F	1	**0**xxxxxxx			
11	128-2,047	U+0080	U+07FF	2	**110**xxxxx	**10**xxxxxx		
16	2,048-65,535	U+0800	U+FFFF	3	**1110**xxxx	**10**xxxxxx	**10**xxxxxx	
21	65,536-1,112,064	U+10000	U+1FFFFF	4	**11110**xxx	**10**xxxxxx	**10**xxxxxx	**10**xxxxxx

The main advantage of UTF-8, in addition to backward compatibility with ASCII, is that it is "self-synchronizing". The high order bits of every byte determine the type and number of bytes for each character. Since there is no overlap of values, a receiving computer can reevaluate a transmission by backing up at most three bytes to determine the start of a character.[v]

To encode UTF-8 you need to understand decimal to digital and hexadecimal conversion. To illustrate we will encode the, ASCII character "a", decimal value 97, and the Unicode value U+20AC, representing the Euro symbol, €.

Example 1: Converting Decimal Value to Unicode

The easiest way to convert a decimal number to binary is to divide the number by 2 and record the remainder. Repeat until the quotient or number is equal to 0. Then write the remainders in reverse order to obtain the binary number. Then convert the binary value to hexadecimal.

1 Divide number by 2 until number is equal to 0

$$\begin{array}{r} 48 \\ 2\overline{)97} \\ 8 \\ 17 \\ 16 \\ 1 \end{array} \quad \begin{array}{r} 24 \\ 2\overline{)48} \\ 4 \\ 08 \\ 8 \\ 0 \end{array} \quad \begin{array}{r} 12 \\ 2\overline{)24} \\ 2 \\ 04 \\ 4 \\ 0 \end{array} \quad \begin{array}{r} 6 \\ 2\overline{)12} \\ 12 \\ 0 \end{array} \quad \begin{array}{r} 3 \\ 2\overline{)6} \\ 6 \\ 0 \end{array} \quad \begin{array}{r} 1 \\ 2\overline{)3} \\ 2 \\ 1 \end{array} \quad \begin{array}{r} 0 \\ 2\overline{)1} \\ 1 \end{array}$$

2 Record remainders in reverse order

128	64	32	16	8	4	2	1
0	1	1	0	0	0	0	1

Leading Zero Added

3 Convert the binary value to hexadecimal

8	4	2	1	8	4	2	1
0	1	1	0	0	0	0	1

Hexadecimal 6 **1**

4 Write the Unicode code point

97 –Latin character "a"	**U+0061**

Example 2: Converting Unicode Value to UTF-8 Encoding

Unicode characters are represented by hexadecimal notation. <u>The value U+20AC is only the bit sequence or the code point to represent the Euro symbol "€"; it is not the hexadecimal UTF-8 encoded string created by the application layer.</u> To convert the code point to UTF-8 encoding, we must first

convert the code point to binary, encode the binary into a UTF-8 multi-byte sequence and then convert the sequence back to hexadecimal.

If you need to convert hexadecimal to decimal follow the steps below:

1 Convert hexadecimal bit sequence to binary. Hexadecimal is a binary “shorthand” where 4 bits are equal to one hexadecimal character from 0 to F.

2	0	A	C
0010	0000	1010	1100

2 Arrange the bits in order from right to left on the binary table

32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0

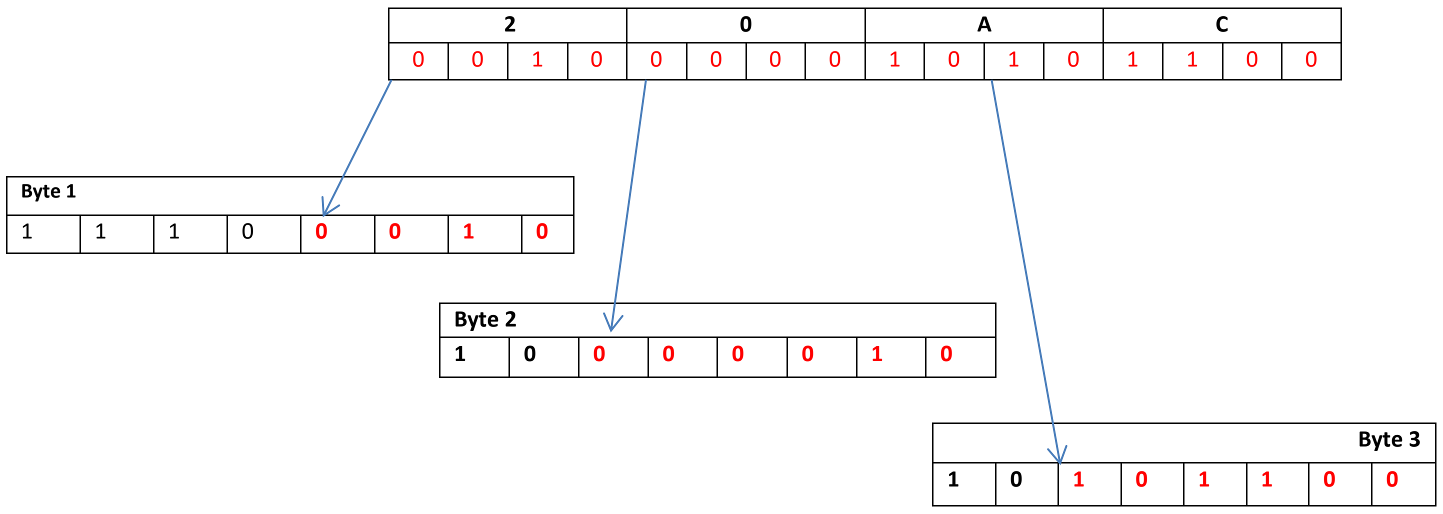
3 Add up the positions with ones

8192 + 128 + 32 + 8 + 4 = 8,364 The Unicode character U+20AC has a decimal value of 8364

1 **Convert Unicode code point to binary**

2				**0**				**A**				**C**			
0	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0

2 Encode the bits to UTF-8 encoding. Referring to the table above, the value U+20AC is between U+0800 and U+FFFF (8,364). A 3 byte sequence is needed to encode this code point. The first byte will **begin with “1110” and each continuation byte will begin with “10”**



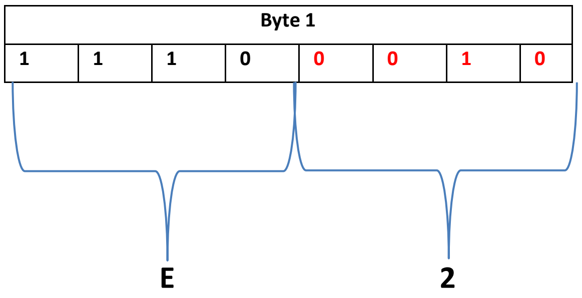
2				**0**				**A**				**C**			
0	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0

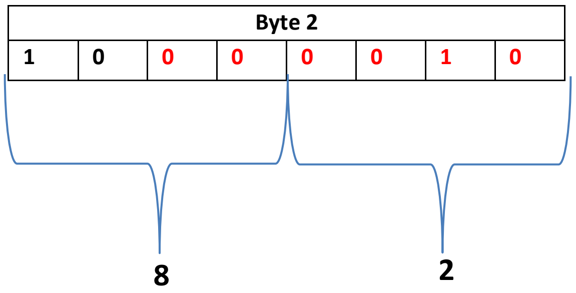
Byte 1							
1	1	1	0	**0**	**0**	**1**	**0**

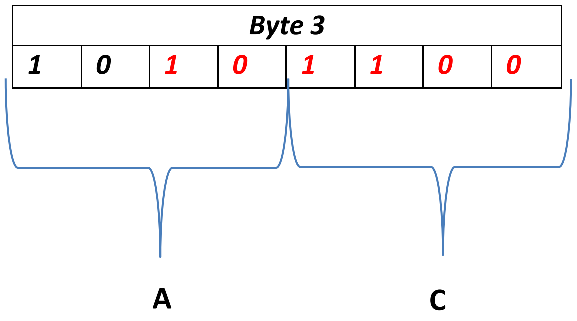
Byte 2							
1	**0**	**0**	**0**	**0**	**0**	**1**	**0**

							Byte 3
1	**0**	**1**	**0**	**1**	**1**	**0**	**0**

3 Convert the encoded multi-byte sequence to hexadecimal by grouping the bit sequence into groups of four and applying a hexadecimal value to each group







The Unicode character U+20AC has a hexadecimal UTF-8 encoding of **0xE2 82 AC**

Encoding Alternatives

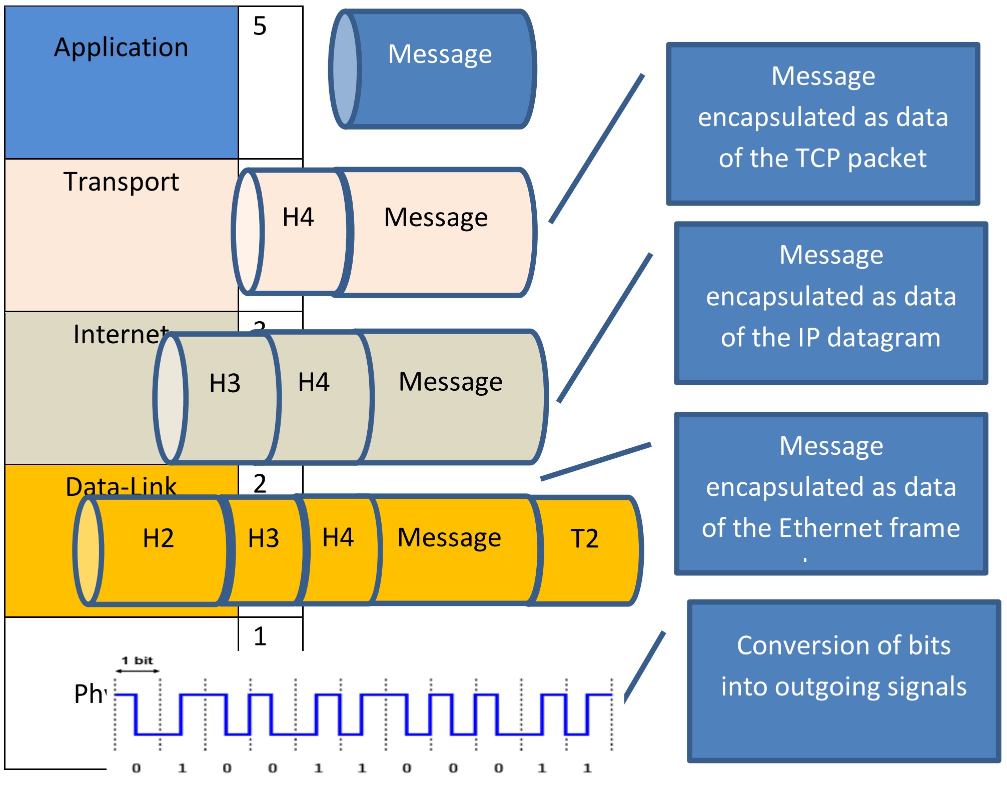
Sometimes the application layer will use a number of bits to represent alternatives in the message, such as a field representing the type of message, or the protocol to use. Suppose, you needed to represent 4 alternatives; the bit values of 00, 01, 10 and 11 could be used. Notice, for 4 alternatives 2 bits are needed. The rule is a = 2^b; "a" is the number of alternatives and "b" is the number of bits needed. How many bits are needed for 12 alternatives? You would need 2^4 which gives 16 alternatives, of which 4 will be unused. You should memorize the number of alternatives that can be represented by 4, 8 and 16 bits, since these are common field sizes. Each added bit doubles the number of possibilities. Contrarily, each bit subtracted cuts the number of alternatives in half.

Bits	Alternatives	Examples
1	2	Yes\No Male\Female
2	4	Married, Single, Divorced, Widowed
3	8	Number of Company Departments
4	16	Top 10 Google Searches
8	256	Number of Users in company
16	65,536	Number of colour levels
32	4,294,967,296	Number of IPv4 Addresses

Many applications today involve voice and video. When you speak into a microphone your voice rises and falls thousands of times per second at different frequencies. This sends pressure waves into the microphone which converts the waves into electricity. The resultant electrical signal is "analogous" to your voice, and is called an analog signal. To encode an analog signal a special electrical circuit is used called a CODEC (Encoding\Decoding). This circuit measures the amplitude of the wave and converts it into a digital value, called encoding. When the digital signal is converted back to an analog signal, to play through a speaker, the signal is decoded.

The TCP/IP Protocol Stack in Action

After the application layer has converted the message into a digital format and followed the protocol specifications the message is passed to the Transport layer to begin its outgoing transmission. The message is placed in the Transport's data field. The process of placing a message in data field of another layer is called "encapsulation". It is similar to placing an envelope inside anther envelope. The Transport layer adds a header to the TCP packet and passes the packet to the Internet layer. The Internet layer process encapsulates the packet inside an IP datagram by adding an IP header. The datagram is then passed to the Data-Link layer which encapsulates the datagram inside an Ethernet frame (or any other LAN or WAN technology). The frame is passed to the physical layer which converts the bits of the frame into signals for outgoing transmission. The physical layer does not do encapsulation. It connects the sending computer to a switch, router or host. While we think of a direct connection from application layer to application layer on the receiving computer, the connection is actually virtual. The only real connection is at the physical layer. The message travels down 5 layers (from the application layer), across the physical layer and up 5 layers to the application layer on the receiving computer.


Application
5
Message
Transport
H4
Message
Internet
3
H3
H4
Message
Data-Link
2
H2
H3
H4
Message
T2
1
1 bit
Ph
0 1 0 0 1 1 0 0 0 1 1
Message encapsulated as data of the TCP packet
Message encapsulated as data of the IP datagram
Message encapsulated as data of the Ethernet frame
Conversion of bits into outgoing signals

At the receiving computer, the message is received by the physical layer, decoded and passed to the data link layer. The frame is checked for errors and if no errors the FCS, trailer is removed. The header is checked to make sure that the MAC address matches the destination computer, and if correct, the frame header is removed and passed to the Internet layer. The Internet layer checks to make sure that the destination IP address is correct and then removes the header and passes the datagram up to the transport layer. The transport layer reassembles the packets if necessary and arranges the packets in the correct order and passes the packets to the application layer which displays the original message.

Standard Organizations:

Standards are made by fact called “de facto” or by law called “de jure”. If the majority of the market spontaneously begins using a standard, this is an example of a de facto standard. For example, no standard body legislated that UTF-8 encoding had to be used. People just started using it because of its benefits and now it dominates the web. Most standards, however, are legislated by a standard making body.

It is important for programmers to have a brief understanding of standard making bodies, because if you are working on a new project, it is common practice to design a project to conform to some standard technology. You will need to know where to find the appropriate standard for your project.

At the International level there are two standard making bodies. The International Telecommunications Union, Telecommunications division (ITU-T) is an agency of the United Nations and is primarily responsible for standards defining how telecommunication networks operate and interwork when they expand beyond national boundaries, such as satellite communication. The ITU-T is best known for its V standards for data transmission over phone lines, and X standards for data transmission over public digital networks.

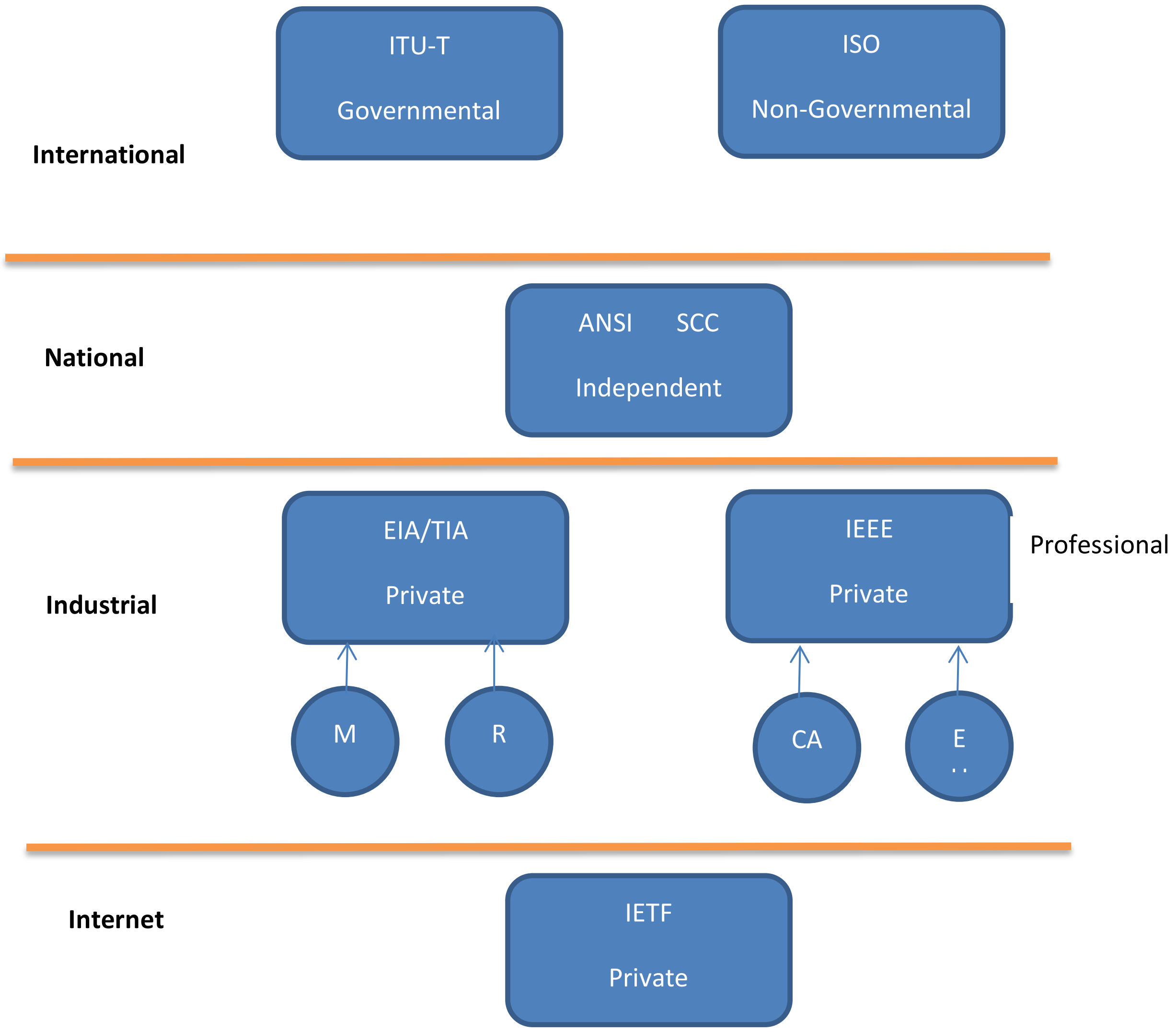
The International Organization for Standardization (ISO, notice the name does not match the acronym) is an independent non-governmental international body make up of members of other national standard bodies. The ISO has over 162 participating countries which review the standards of national bodies and adopt those that promote international trade. They make standards on all topics, from food and safety management, factory quality control to Information Security.

National standard making bodies such as American National Standards Institute (ANSI) in the United States and the Standard Council of Canada (SCC) in Canada have three principal roles. Their primary role is to promote national businesses. When technology, developed by local business, becomes a national or international standard, the Gross National Product increases because other companies pay a royalty to use the technology. As more people buy the technology the local business expands and employees more people. The second role is to protect national health and safety. Private businesses are not motivated to spend money to develop national standards for health and safety; this is an important governmental role. Lastly, national standard making bodies coordinate the development of standards by other industry or professional groups. They are always on the lookout for promising technologies.

ANSI is a private non-profit organization, but its membership consists of private and public organizations. ANSI controls the standards for fiber-optic networks and other MAN technologies. The SCC is a Canadian crown corporation and also operates independent of government, but reports to Parliament through the Minister for Industry. The SCC has always worked closely with ANSI and in 2015 signed a Memorandum of Understanding (MOU). As John Walter, the CEO of SCC stated:

SCC and ANSI are working together to break down barriers between Canada and the US. This MOU is a pivotal step toward making progress on joint Canada-US standards, testing and certification procedures. The reduction of duplicative standards, testing and certification requirements between our two countries will enhance our competitiveness.[vi]

Industrial standards are developed by the manufacturers and retailers of electronic and telecommunications equipment. The Electric Industry Alliance (EIA) and the Telecommunications Industry Association (TIA) are the principal players. Their goal is to develop public awareness, education and standards for the physical interface and electronic signaling. Often the two associations will produce joint standards EIA/TIA on interfaces, such as EIA/TIA 586A cable interface.



The Institute of Electrical and Electronic Engineers (IEEE) is the largest professional standard making body in the world. With branch members in all industrial countries IEEE standards have international compliance. The IEEE is responsible for all LAN technologies such as 802.3, Ethernet and 802.11 wireless LAN (WLAN) and 802.15 PAN (Personal Area Network) which connects Bluetooth devices to a wired network. Since the bottom two layers are OSI layers, technically LAN standards are not official until adopted by the ISO.

Lastly, the Internet is a non-governmental group with membership open to any user of the Internet. The principal agency is the IETF, Internet Engineering Task Force. This group is responsible for TCP/IP and issues standards called RTF, Request for Comment.

The process of standardization is long and often involves multiple standard making bodies. For example, Ninetendo worked with the EIA to develop a “kids proof” 6 wire console cable. Apple, who was developing a new highspeed serial port needed a rigid connector for its new “Firewire” technology. Firewire was approved by the IEEE and standardized as IEEE 1394. Apple originally saw Firewire as a replacement for SCSI, same speed high, with a plug and go connector. ANSI, who coordinates standards in the US, saw FireWire as a new way to connect peripheral devices, including other computers. This new and easy method to connect devices was seen as an opportunity to expand American business. ANSI, who sits on the ISO, lobbied to get the technology adopted as an international standard. FireWire however, today is dead; USB is the main method to connect peripheral devices.

---

[i] Gerald Cole, *Computer Networking For System Programmers*, John Wiley & Sons, New York, 1990, p.24

[ii] The OSI was conceived as a 7 layer model: application, presentation, session, transport, network , data-link and physical. The TCP/IP model was conceived as a 4 layers:, application, transport, Internet, and network interface. Today, The TCP/IP model is a hybrid 5 layer model, with the bottom 2 layers complying to the OSI standard, and the upper 3 layer complying with the TCP/IP standard. While the OSI 7 layer model is not used for interconnecting networks, it is universally used to teach networking concepts.

[iii] Read Joel Spolsky’s excellent article, “The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)” at http://www.joelonsoftware.com/articles/Unicode.html

[iv] WikiPedia.org. Unicode has replaced ASCII (American Standard Code for Information Interchange) because it can only represent Latin letters used in English with its 7 bit character space (2^7 = 128 maximum character space)

[v] The UTF-8 is standard actually uses a maximum of 6 bytes to encode characters, but the W3C has restricted the standard to 4 bytes.

[vi] John Walter, CEO, SCC from the www.scc.ca web site.