Computer Network Layers
CIS748 Class Notes

Alex S.[*]

1 The Layers

Normally, the task of communication from computer to computer is broken up into *layers*. Each device attached to the network has a corresponding 'stack' of these layers, where each layer conceptually talks to the corresponding layer on the other computer. Each layer only needs to know how to interface to the layers above and below it.

The 'standard' layered model used for illustrative purposes only is the OSI Model[1]. It consists of seven layers:

7 Application Layer (Provide end-user services, like e-mail)

6 Presentation Layer (Data compression, and other data conversion)

5 Session Layer (Authentication/Authorization)

4 Transport Layer (Guarantee end-to-end data transfer—from machine to machine)

3 Network Layer (Routing, accounting).

2 Data Link Layer (Transmit/recieve packets, resolve hardware addresses)

1 Physical Layer (Physical cable, medium, air)

There is an interesting quote from *UNIX System Administration Handbook*: "Some think a financial layer and a political layer should be added to these. There is no magic about the number seven; seven committees were involved in the specification, and one layer was created for each."

In any case, there have been some attempts to implement the OSI model, and some sorta actually worked, etc., but it never managed to catch on.

[*]`alex@theparticle.com`

[1]`http://en.wikipedia.org/wiki/OSI_model`

1.1 PDU, Protocol Data Unit

Layers communicate using a thing called a PDU, or Protocol Data Unit. Each layer may have a different PDU format, etc. It is only required that corresponding layers be able to understand their PDU (the *interface* between layers). For this purpose, we often attach a number to the PDU, to know what layer it belongs to. For example, a 3-PDU would corresponds to the 3rd layer PDU, and so on for any N-PDU.

Layer N needs to be able to talk to layer $N + 1$ and layer $N - 1$, but not to any other layer.

1.2 Encapsulation

Layers are arranged in a way that each layer only has to know about layers that are directly below or directly above. All data from the higher layers is wrapped (and subsequently unwrapped at the destination) so that, for example, the data-link layer doesn't have to know whether you're browsing the web or transferring files—all it sees are just a bunch of bits that need to be transferred.

1.3 Services

We can also view the layered model as higher layers getting services from lower layers and lower layers providing a service to higher layers.

1.3.1 Service Access Points

An entity at layer N may service more than one entity at layer $N + 1$. Layer N may use a *Service Access Point* address to determine which entity at a higher layer will get serviced. An example of that are Port addresses in TCP/IP.

2 OSI Model

Even though the OSI Model isn't the actual 'real' model, it is still useful for illustration purposes as an idealized network model. It consists of seven layers (descriptions taken from Wikipedia):

2.1 Application Layer

Layer 7, the highest layer: This layer interfaces directly to and performs common application services for the application processes. The common application services provide semantic conversion between associated application processes. Examples of common application services include the virtual file, virtual terminal (for example, Telnet), and "Job transfer and Manipulation protocol" (JTM, standard ISO/IEC 8832).

2.2 Presentation Layer

Layer 6: The Presentation layer relieves the Application layer of concern regarding syntactical differences in data representation within the end-user systems. MIME encoding, encryption and similar manipulation of the presentation of data is done at this layer. An example of a presentation service would be the conversion of an EBCDIC-coded text file to an ASCII-coded file.

2.3 Session Layer

Layer 5: The Session layer provides the mechanism for managing the dialogue between end-user application processes. It provides for either duplex or half-duplex operation and establishes checkpointing, adjournment, termination, and restart procedures. This layer is responsible for setting up and tearing down TCP/IP sessions.

2.4 Transport Layer

Layer 4: The purpose of the Transport layer is to provide transparent transfer of data between end users, thus relieving the upper layers from any concern with providing reliable and cost-effective data transfer. The transport layer controls the reliability of a given link. Some protocols are stateful and connection oriented. This means that the transport layer can keep track of the packets and retransmit those that fail. The best known example of a layer 4 protocol is TCP.

2.5 Network Layer

Layer 3: The Network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the Transport layer. The Network layer performs network routing, flow control, segmentation/desegmentation, and error control functions. The router operates at this layer – sending data throughout the extended network and making the Internet possible, although there are layer 3 (or IP) switches. This is a logical addressing scheme - values are chosen by the network engineer. The addressing scheme is hierarchical.

2.6 Data Link Layer

Layer 2: The Data link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical layer. The addressing scheme is physical which means that the addresses (MAC) are hard-coded into the network cards at the time of manufacture. The addressing scheme is flat. Note: The best known example of this is Ethernet. Other examples of data link protocols are HDLC and ADCCP for point-to-point or packet-switched networks and LLC and Aloha for local area networks. This is the layer at which bridges and switches operate. Connectivity is provided only among locally attached network nodes.

2.7 Physical Layer

Layer 1: The physical layer defines all electrical and physical specifications for devices. This includes the layout of pins, voltages, and cable specifications. Hubs and repeaters are physical-layer devices. The major functions and services performed by the physical layer are:

- establishment and termination of a connection to a communications medium.
- participation in the process whereby the communication resources are effectively shared among multiple users. For example, contention resolution and flow control.
- modulation, or conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. These are signals operating over the physical cabling – copper and fibre optic, for example. SCSI operates at this level.

3 TCP/IP

The TCP/IP can be thought of consisting of four layers:

4 Application Layer (End-user application programs)

3 Transport Layer (Communication among programs on a network)

2 Network Layer (Communication, addressing, and routing)

1 Link Layer (Network hardware and device level)

The TCP/IP does most of the things the OSI model does, except it does it in four layers, and is actually a lot more robust and real than the OSI idea.

TCP/IP is setup in such a way that it doesn't really matter which network is used underneath. The Link Layer can use physical wire, wireless, serial lines, etc.

4 IEEE Standards

In 1985 the Computer Society of the IEEE started a project called Project 802, to set standards to enable intercommunications between equipment from variety of manufacturers. It is a specification for the data-link and physical layer of major LAN protocols.

The 802 standard split the data-link layer into 2 layers:

- Logical Link Control (LLC): IEEE 802.2

- Media Access Control (MAC): IEEE 802.3 (CSMA/CD), IEEE 802.4 (Token Bus), IEEE 802.5 (Token Ring), IEEE 802.5 (DQDB—distributed-queue dual-bus network), IEEE 802.11 (Wireless).

The physical layer is also split into a few layers, depending on the type of network we're dealing with. Generally, we'll have two primary things: PMI (Physical Medium Independent and PMD (Physical Medium Dependent). Those are generally broken up into PLS (Physical Layer Signaling), AUI (Attachment Unit Interface), MAU (Medium Attachment Unit), MDI (Medium Dependent Interface).

5 Computer Networks Intro (Review; Other Notes)

5.1 Serial Lines

Serial lines are generally the most basic and most common communication medium you can have between computers and/or equipment. They generally consist of relatively cheap wire sending bits across at relatively slow speeds. That's all there is to them.

Because of their simplicity, serial ports are available on pretty much all sorts of hardware (except many new laptops). Many port and jack configurations are supported (there are round jacks (DIN-8), 9 pin jacks (DB-9), 25 pin jacks (DB-25), etc.) Most times, only a few of the wires are actually used. In fact, the most basic serial port communication between two computers uses only 3 wires: (transmit, receive, and ground). Sometimes an RJ-45 is used for serial communications—and there are various standards on how pins map to which wires.

The transmission can use hardware flow control or software flow control. The hardware flow control generally requires more wires to be connected. Software flow control is harder on the CPU.

5.2 Null Modem

There is a concept of a 'null' modem. This is basically when the serial cable isn't connected to a device, but to another computer. It is the same wire as a regular serial cable, except a few wires have been crossed over... namely the send/receive wires—along with a few hardware flow control wires if those are used).

5.3 Terminal Devices

In the UNIX world, there is a thing called a 'terminal'. It is basically a screen/keyboard connected to the 'computer' via a serial cable. You've seen a similar idea when you use Telnet (or SSH) to log into Solaris/Linux computers (except connection doesn't travel over serial cables—but it still uses terminal devices).

In fact, there is a program called `getty` which presents a login prompt on a particular serial port. Something you could try: connect two computers via a null-modem cable, do a `getty` on the serial port on one computer, and on another computer, you can cat the serial port device to read/write it and actually be logged into the other computer through a serial port (and obviously use all the command line programs).

UNIX has a bunch of *virtual* terminals, and when the system starts up, `getty` is executed on a bunch of those terminals. That's why you see a login prompt on the screen (your screen/keyboard are just treated as command line terminals as far as the computer are concerned).

There is a program called `stty` which you can use to adjust the properties of your terminals. For example, to change the speed of a serial port, or to enable/disable control flow options, specify special characters, number of rows/columns, etc.

5.4 IP over Serial Lines

It turns out, you can use TCP/IP over serial lines. This shouldn't come as a surprise, as many of you might still be using dial-up Internet. Basically, there are two protocols that do the job: SLIP and PPP.

Both have the basic idea of encapsulating IP packets, and transmitting them (with some flow control) over the serial line. The process would work something like this: the dial-up server is setup to pickup a phone line when it rings and start a `getty` on it. You dial up (ATDT), see a terminal (or your dial-up networking software sees the login), it proceeds to login (via the simple UNIX like login prompt), it then runs PPP or SLIP to start a remote protocol engine—which then configures the already established serial link for transfer of IP packets.

5.4.1 SLIP

SLIP (Serial Line Internet Protocol) is probably the simplest one, and at one point, you'd use it if you wanted to setup a dial-up connection to the Internet from your Linux box.

5.4.2 PPP

PPP (Point-to-Point Protocol) is a huge bloated standard that is capable of quite a bit more than just sending packets. When you dial into the your ISP, this is what you're using.

5.4.3 Others

There may be other such encapsulation protocols used—so you can't just dial up and expect things to work (via standard networking configuration). Some ISPs require you to have special "dialer" software that enables your Windows box to use *other* encapsulation protocol. AOL was notorious for that.

5.5 TCP/IP

TCP/IP is a whole bunch of various protocols stuck into one:

5.5.1 IP

Internet Protocol, or IP, is used to transport RAW data from one machine to the next. This involves hopping from computer to computer until the data gets there.

5.5.2 ICMP

Internet Control Message Protocol, or ICMP, is used as a helper protocol for IP. It provides assistance with error messages, routing, and echo requests.

5.5.3 ARP

Address Resolution Protocol, or ARP, provides services to resolve logical network addresses to hardware network addresses. Generally, figuring out which IP addresses correspond to which MAC addresses.

It basically screams out to the network "does anyone know the hardware address for this IP". Then some machine (if it is alive) will respond, "Yes, here I am."

5.5.4 UDP

User Datagram Protocol, or UDP, sends a packet from one *program* to another *program*. It's unreliable, and connectionless.

5.5.5 TCP

Transmission Control Protocol, or TCP, sends a stream of data from one *program* to another *program*. It is reliable, and connection oriented (establishes a 'session').

5.6 Segmentation and Packets

TCP/IP breaks up all traffic into packets. A packet is just a structure with a header, and some data. When referring to a packet on a LAN, we often use the term "frame" (even though when a packet travels in a frame it has a bit more info attached to it).

Every network has MTU, or Maximum Transfer Unit. This is the maximum size of a packet that can traverse the network. The IP layer, breaks up your data into packets of MTU size, transfers those to the destination machine, where they are reassembled.

Every packet, obviously has to have source IP, destination IP, size of data, etc. When it is encapsulated in some other protocol, those are also included. For example, a UDP packet carrying 100 bytes of data on an Ethernet would have:

- 14 bytes Ethernet Header (includes things like MAC addresses)

- 20 bytes IP Header (includes things like source/destination IP addresses)
- 8 bytes UDP Header (includes things like source/destination port addresses)
- 100 bytes of user data
- 4 bytes Ethernet Trailer (terminates the frame)

So while we were just sending 100 bytes via UDP, over the Ethernet, that sent 146 bytes.

5.7 Internet Addresses

Since we are primarily concerned with TCP/IP, let's discuss network addresses. An IP addresses is just a 32 bit number, written out with each byte (0-255) separated by dots. For example, 192.168.0.1 is an IP address.

There are several classes of networks (and thus addresses). Each address belongs to a class. Part of the IP address indicates the network address. For example, N.N.N.H has 3 bytes dedicated to the network address, and 1 byte to the host address. Various classes of networks:

- Class A: Starts with 1-126. Major networks. N.H.H.H
- Class B: Starts with 128-191. Large networks. N.N.H.H
- Class C: Starts with 192-223. Small networks (easy to get). N.N.N.H
- Class D: Starts with 224-239. Multi-cast addresses.
- Class E: Starts with 240-254. Experimental addresses.

In any case, most networks that most people are concerned with are B and C.

5.8 Routing

Routing is the process of directing a packet through the network to the destination computer. You can view it as asking for directions. You are sent to deliver something to an unknown city. You ask for directions, and someone may point you towards a general direction (or towards someone who may known the general direction). You soon get to the city, and you start asking for more specific information concerning the street address, etc. Same idea works for routing.

For routing to work, the operating system maintains a routing table. It is just a table of network masks and destinations. Whenever the kernel sees a packet, it figures out which network it corresponds to, and sends the packet to that network—that's all there is to it. There are also 'default' routes; when everything else fails (you don't know what to do with a packet) you send it off to the default route.

There is static routing and dynamic routing. Static means that you setup the routing table manually, at boot-time, etc., and it stays around. For small networks that is quite efficient.

Dynamic routing involves the use of routing daemons like `routed` or `gated` to discover the network topology and how to reach distant networks. There are many approaches to this, which we won't get to at this time.