Chapter 7

Networking

The field of networking is vast and to cover everything we would need several courses on the subject. This section is meant to give you a basic introduction to the current networking model and to some of the common networking protocols you will see as a system administrator.

7.1 The 7 Layer OSI Model

The **OSI Model** is a layered, abstract model that describes network communication. By using this layered abstraction, each layer can function (somewhat) independently of the others. For example, consider the HTTP protocol, which is a layer 7 protocol that your web browser uses to communicate with web servers. This protocol can implement something like “get this page: index.html” without having to worry about how the data will actually be transmitted over the network.

7.1.1 James Bond and the 7 Layer OSI

Before we discuss the function of each layer in the OSI model, its helpful to think about what we are actually trying accomplish. I found this example of James Bond relaying a secret message between two people to be useful. This example was written by Dick Lewis <richard@lewistech.com>, Lewis Technology Inc. <http://www.lewistech.com/>

James Bond meets Number One on the 7th floor of the spy headquarters building. Number One gives Bond a secret message that must get through to the US Embassy across town.

Bond proceeds to the 6th floor where the message is translated into an intermediary language, encrypted and miniaturized.

Bond takes the elevator to the 5th floor where Security checks the message to be sure it is all there and puts some checkpoints in the message so his counterpart at the US end can be sure he’s got the whole message.

On the 4th floor, the message is analyzed to see if it can be combined with some other small messages that need to go to the US end. Also, if the message was very large, it might be broken into several small packages so other spies can take it and have it reassembled on the other end.

The 3rd floor personnel check the address on the message, determine who the addressee is, and advise Bond of the fastest route to the Embassy.

One the 2nd floor, the message is put into a special courier pouch (packet). It contains the message, the sender and destination ID. It also warns the recipient if other pieces are still coming.

Bond proceeds to the 1st floor where Q has prepared the Aston Martin for the trip to the Embassy.

Bond departs for the US embassy with the secret packet in hand. On the other end, the process is reversed. Bond proceeds from floor to floor where the message is decoded.

7.1.2 The 7 Layers Explained

Now that we've seen an example, lets examine each layer individually. We will start at the top layer (highest level of abstraction) and work down to the bottom.

Layer 7: Application This layer at which applications access the network. This includes many of the high level protocols like HTTP for website, FTP for file transfers, and SMTP for e-mail.

Layer 6: Presentation This layer translates data from the Application layer into an intermediate format for use during network transmission. This is also the layer that data compression techniques and encryption protocols such as SSL are implemented.

Layer 5: Session This layer is responsible for inter-host communication and manages the connections (sessions) between two hosts. It is responsible for establishing a new connection when one is needed, manage active connections, and terminate connections when they are no longed needed.

Layer 4: Transport This layer is responsible for end-to-end connections and data transfer. This layer handles error recognition in received data. It also segments/desegments large messages that are transmitted/received. The most popular protocols in this layer are TCP and UDP. TCP is 'connection-oriented' and provides reliable data transfer by resending data until delivery is acknowledged. UDP is 'connection-less' and sends data without guaranteed delivery.

Layer 3: Network This layer is responsible for logical addressing and path determination. The best known example of a layer 3 protocol is the internet protocol (IP). This implements routing and logical addressing through the use of IP addresses, which will be discussed later.

Layer 2: Data-Link This layer is responsible for physical addressing and packaging data for transfer between two network entities. The most common layer 2 protocol is Ethernet. This layer is often split into two sub-layers: Logical Link Control (LLC), and Media Access Control (MAC). This is the layer at which switches and bridges operate. Communication on this layer is provided only for locally attached network devices.

Layer 1: Physical This is the lowest layer of the abstraction and is responsible for the actual electrical signal transmission. This layer includes things like the pin layouts of network cards and network cable specifications. A well know example of a layer 1 protocol is the 802.11{a,b,g} wireless specifications.

While each of these layers is important in our networking systems, you will likely be spending most of your networking time working with the Layer 3 IP protocol and a variety of Layer 7 protocols.

7.2 IP Addressing

IP (internet protocol) is the most common network layer protocol in use today. There are several versions of IP in existence, though only IPv4 and IPv6 are the only ones that are actually used. IPv4 is by far the most dominant IP version in use on the internet today, so most of this discussion will involve IPv4 (or just IP) addresses.

7.2.1 Representation

IPv4 uses 32 bits to represent on address. This allows for a total of $2^{32} = 4,294,967,296$ unique addresses. However, many of these IP addresses are reserved for special purposes such as private networks. While this may seem like a lot of addresses, the number of unallocated addresses is decreases at an increasing rate. This has helped to stimulate the movement to IPv6. IPv6 uses 128 bit addresses, allowing for $2^{128} \approx 5 \times 10^{28}$ addresses. While this seems ideal, IPv6 is still in the early stages of development and has not been widely adopted, so we will focus on IPv4 for now.

While IP addresses are 32-bit binary numbers, they are rarely written out in binary form. There are several human readable notations, but the **dot-decimal notation** is by far the most popular. In this notation, addresses are written as 4 octets in decimal form separated by periods. For example, `www.berkeley.edu` has the IP address `169.229.131.92`.

7.2.2 Allocation

The allocation of IP addresses is handled by the **Internet Assigned Numbers Authority**. IP addresses are, at least ideally, hierarchically structured. Thus, IANA assigns large blocks of IP addresses to **Regional Internet Registries**. These organizations are responsible for large geographical regions. For example, the IPs from `62.0.0.0 - 62.255.255.255` are assigned to **RIPE**, the RIR for Europe, and the Middle East. The RIR is then responsible for assigning IP addresses to organizations within the region. The RIRs also maintain publicly searchable **WHOIS** databases that contain information about IP assignment.

7.2.3 Private Addresses

Not all of IPs in the full IP range are publicly assigned by IANA. Certain ranges have been designated **Private Networks**. These ranges cannot be routed outside the private network and hosts within the private network cannot communicate directly with hosts outside the private network (though this can be solved through the use of **Network Address Translation**, which will be discussed later). The four designated ranges are:

IP Range	IPs in network
`10.0.0.0 - 10.255.255.255`	`16,777,216`
`172.16.0.0 - 172.31.255.255`	`1,048,576`
`192.168.0.0 - 192.168.255.255`	`65,536`
`169.254.0.0 - 169.254.255.255`	`65,536`

Since these IPs cannot be routed outside the local private network, these ranges can be used by any group for a private network. The most common example is the average Cable/DSL customer with a home network of several computers sharing an internet connection. If you are on such a network, take a look at your IP address. You will likely find that that your IP lies within one of these private ranges (most likely a `192.168` address). In order to communicate with the Internet, your ISP assigns a public IP address to your DSL/Cable Modem. More likely than not, your modem does not connect directly to another computer, but instead to a router. This router manages your home network by assigning hosts IPs in a private address range and doing the network address translation needed to allow those hosts to communicate with the Internet.

You can see the non-routability of these addresses in this context. Data going to/from addresses in the `192.168` private range cannot leave go past the route that manages that network. Host `192.168.1.101` can communicate with `192.168.1.102` directly, but in order for either host to communicate outside the private network, it will take on the public address of the modem. Having this sort of isolation allows for any number of private networks to exist independently. The number is only limited by the limited availability of public IPs to give private networks in order for them to communicate with the outside. Of course, there is no limit if you have networks that don't need to communicate with outside networks.

7.2.4 Loopback

In addition to the private ranges above, there is an super private range called **loopback range**. This range consists of all IPs between `127.0.0.0` and `127.255.255.255`. This range is reserved for localhost communication. Data sent to these addresses never leave the source host, and will appear as incoming data to that host (i.e. a loopback connection).

7.3 Network Address Translation (NAT)

Network Address Translation (called **NAT** or **IP Masquerading**) is a process that involves rewriting the source and/or destination IP and/or port or packets as they pass through the NAT host. The role of NAT host can be played by any machine capable or running the required software. This is often done by a router in something like a home network, but can also be done by an actual host connected to a simple switch.

There are multiple types of NAT, including **one-to-one NAT** and **overloaded NAT**. **One-to-one NAT** is a simple IP address translation and doesn't involve any port mapping. This might be used if you happen to own several public IPs and you want each to correspond to certain machines on your internal network. In this situation, there is a one-to-one mapping between the public IPs and the private network IPs, so all the NAT machine has to do is rewrite the IP as it travels by.

The more commonly used (and more complicated) NAT is **overloaded NAT**. This type is used to allow multiple hosts on a private network to access an outside network using the same public IP. In this scheme, when a host on the internal network communicates with the outside network, it takes on the public IP assigned to the NAT and a port that is decided by the NAT. That data is then sent over the outside network to the remote host. When the remote host sends data back, it uses the rewritten IP and port as a destination and sends it back over the network. When this data gets back, the NAT uses the port to translate back to the correct internal IP and port.

To see an example of how this works, lets consider a simple network with a couple of hosts on the private network that connect through a NAT router. We'll say `Host A` is host on the private network with IP `192.168.1.101`. The NAT router, `Host B`, has two network interfaces, one connected to the private network with IP `192.168.1.1`, and one connected to the outside world with public IP `192.58.221.204`. Lets also say that the remote host, `Host C` has IP `64.233.187.99` (which happens to be `www.google.com`.

Any packet sent on the network must have both a source and destination IP address and port. We will represent them as `IP:port`, for example, `192.58.221.243:80` would be port `80` on IP `192.58.221.243`. Now lets follow a simple transaction between `Host A` and the webserver on `Host C`. We will follow the data packets, keeping track of source and destination IP and port.

When `Host A` sends the request for data to `Host C`, it will specify the standard web server port `80` as a destination port and one of the high numbered ports, say `48085`, as a source port. The packet information is:

```
Source: 192.168.1.101:48085
Destination: 64.233.167.99:80
```

This packet is then routed on to the network gateway, `Host B`. The NAT sees that the destination is outside the private network, so it will undergo translation. The NAT will come up with a high numbered port on the public connection, say `34533`, and rewrite the source IP and port. The NAT keeps an internal table of these translations, something like:

Source IP	Original port	Modified port
192.168.1.110	48085	34533

After going through translation, the packet leaves `Host B` with information:

```
Source: 192.58.221.204:34533
Destination: 64.233.167.99:80
```

The packet is then routed through the internet to `Host C`. Since the destination is a web server, there will be data to be sent back. The response is sent back using the source information of the received packet, so the return data packet leaves `Host C` with information:

```
Source: 64.233.167.99:80
Destination: 192.58.221.204:34533
```

The packet is routed through the internet until it reaches `Host B`. When `Host B` receives the packet, it checks the destination port against the NAT translation table to see if the packets should be rerouted onto the private network. In this case, it sees that the destination port, `34533`, is in the NAT table as a modified port. The packet headers are then rewritten with the data associated with that NAT table entry and routed to the correct host on the private network. Thus, the packet leaves `Host B` on the private network with information:

```
Source: 64.233.167.99:80
Destination: 192.168.1.101:48085
```

It heads back to `Host A` as if nothing had ever happened.

Its important to note that this has interesting effects on some higher level protocols. For example, If you have ever used a file sharing client (for legitimate purposes of course), you may have trouble with the 'NAT Firewall', with the suggestion that you use **Port Forwarding**. The problem comes because the automatically generated entries in the NAT translation table are only created for connections initiated locally, while these programs often require remote hosts be able to initiate connections remotely. If the remote host attempts to start a connection with a host on the private network, it will get stopped at the NAT because there is no translation table entry. **Port Forwarding** is simply adding permanent entries to the NAT translation table.

7.4 Name Resolution (DNS)

While every host on the Internet has a unique IP addresses, as users, we rarely come in contact with these addresses. When you open a web browser, you don't type in `66.102.7.99`, you type in `www.google.com`. These names, called **domain names**, are easier for us to remember than IP addresses, so we typically use them instead. However these name are only for our convenience. The lower layer protocols don't work with IP addresses, not domain names. The **Domain Name System** (DNS) provides the mechanism for resolving domain names to IP addresses.

DNS acts like the phone book on your cell phone. When you open your phone book and call "John", your phone doesn't connect to the network and try to dial "John". That means nothing to the telephone network. All phones care about are actual phone numbers. Your phone book (like DNS) stores the relationship between a convenient name and the number it represents. When you call "John" your phone book looks up "John's" phone number and dials that. DNS works the same way. When you try to connect to `www.google.com`, the DNS resolver on your machine will first look up the IP address for the `www.google.com`, find that its `66.102.7.99`, then continue communication with the IP address. DNS stores other information associated with domain names, but for now we will use the IP address as our example for the DNS resolution mechanism.

DNS is organized hierarchically. IANA manages the root level and **top level domains** like `.com`, `.edu`, `.net`, `.org`, etc. Names that are associated directly under these top label domains, like `berkeley.edu`, must be registered with IANA. When a name is registered, a **subdomain** can be created. Control of the names in this subdomain is delegated from IANA to the owner of the subdomain. For example, domain names like `www.berkeley.edu` and `nuc.berkeley.edu` are managed by UC Berkeley instead of IANA, since Berkeley owns the `berkeley.edu` subdomain. Control can be further delegated as deeper levels of subdomains are created. For example, UC Berkeley can create subdomains like `nuc.berkeley.edu` for the nuclear engineering department and `ocf.berkeley.edu` for the OCF computing center. Since these groups get control of their subdomains, the nuclear engineering department can add a hostname for `sheridan.nuc.berkeley.edu` without needed to contact the university or IANA. Similarly, OCF can add a hostname for `tsunami.ocf.berkeley.edu` without needing to deal with the university or IANA.

Since the delegation of authority is hierarchical, it makes sense that hostnames are stored hierarchically. Each domain/subdomain (and the root level) has a **DNS server** associated with it. The DNS server stores the mapping between domain name and IP address for the names that that domain has authority over. If the domain delegates authority to a subdomain, the DNS server will contain a link to the DNS server for that domain. For example, the DNS server for `berkeley.edu` contains mapping for names like `www.berkeley.edu` and `grad.berkeley.edu`. For a name like `tsunami.ocf.berkeley.edu`, although this is under the `berkeley.edu` domain, authority for the `ocf.berkeley.edu` subdomain has been delegated to the DNS server for `ocf.berkeley.edu`. Thus, the `tsunami.ocf.berkeley.edu` will not be stored in the `berkeley.edu` DNS server, but instead in the `ocf.berkeley.edu` DNS server.

The actual resolution mechanism is quite simple. When resolving a domain name, the resolver starts by querying a domain DNS server for the authoritative DNS server for the next subdomain in the address. When it gets a response that the request is not a subdomain, it requests the address for that host. If the host exists, you will the get IP. If the host doesn't exist, you will get a "failed to resolve hostname" type error. You've probably seen this type of error when you mistype a URL in your web browser.

Lets looks at an example DNS resolution by trying to resolve the IP address for my favorite host `tsunami.ocf.berkeley.edu`. The resolved with start at the root level server and working its way down. The conversation would look something like:

```
Resolver (to root DNS): Where is the DNS server for .edu?
root DNS: Here's the address.
Resolver (to .edu DNS): Where is the DNS server for berkeley.edu?
.edu DNS: Here's the address.
Resolver (to berkeley.edu DNS): Where is the DNS server
         for ocf.berkeley.edu?
berkeley.edu DNS: Here's the address.
Resolver (to ocf.berkeley.edu DNS): Where is the DNS server for
         tsunami.ocf.berkeley.edu?
ocf.berkeley.edu: tsunami.ocf.berkeley.edu is not a subdomain.
Resolver (to ocf.berkeley.edu DNS): OK, what is the address for
         tsunami.ocf.berkeley.edu?
ocf.berkeley.edu: 192.58.221.223
```

At the end of the resolution, the resolver is left with the address of the host name it started with.

7.5 Address Resolution Protocol (ARP)

IP addresses provide a way for logically addressing hosts on a network. Within the logical network layer, every data packet knows both its source host and its destination host. While this creates a logical connection, transmission of data 'over the wire' is done using physical addresses. These physical addresses, usually MAC (Media Access Control) addresses, uniquely identify a piece of network hardware, like an ethernet card. Before a packet can be sent over the network, the logical IP addresses need to be resolved to physical hardware addresses.

You may wonder why you would need a hardware addresses if you already have the IP addresses. Well, the IP addresses only give the addresses of the end-to-end connection hosts. The problem is that 99% of the time, you will be communicating with hosts that aren't on your local network. The data will probably need to be routed over the internet to reach the destination host. Here is where the hardware addresses come in. The data packet will always have the same logical source and destination IP addresses, but the lower level ethernet frame that is transmitted from one physically connected host to another will have the physical source and destination addresses of those two hosts.

For example, consider the how you might communicate between my commonly used host `tsunami`, which is located in Berkeley, CA, and the the UCLA webserver, located, as you might expect, in Los Angeles, CA. As far as the network layer (IP) is concerned, data travels from me (`192.58.221.223`) to the webserver (`169.232.33.135`). However, to actually get from `tsunami` to `www.ucla.edu`, the data will have to hop across several networks, as can be seen from the following traceroute (note that timing data has been dropped for simplicity):

```
aoaks@tsunami:~$ traceroute www.ucla.edu
traceroute to www.ucla.edu (169.232.33.135)
 1  fast2-9.inr-230-spr.Berkeley.EDU (192.58.221.1)
 2  g3-14.inr-201-eva.Berkeley.EDU (128.32.255.109)
 3  ge-1-2-0.inr-002-reccev.Berkeley.EDU (128.32.0.36)
 4  hpr-oak-hpr--ucb-ge.cenic.net (137.164.27.129)
 5  svl-hpr--oak-hpr-10ge.cenic.net (137.164.25.8)
 6  lax-hpr--svl-hpr-10ge.cenic.net (137.164.25.12)
 7  ucla--lax-hpr1-ge.cenic.net (137.164.27.6)
 8  border-1--core-2-10ge.backbone.ucla.net (169.232.4.102)
 9  core-2--csb1-1-ge.backbone.ucla.net (169.232.8.5)
10  www.ucla.edu (169.232.33.135)
```

The actual translation from logical addresses to physical addresses is performed using the **Address Resolution Protocol** (ARP) method. This is actually a relatively simple method. When the resolution starts, the source host knows its own IP address and physical address, and possibly the destination information (we'll see why in a second). For now, assume it doesn't have the destination information. In order for a host (say with IP `192.168.1.101` and HW `00:A0:CC:DE:CD:68`) to find the destination (say with IP `192.168.1.104` and HW `00:A9:DE:A8:C5:92`), the source host will broadcast (send to all hosts on the local network) an ARP request. This says something to the effect of "Who has IP `192.168.1.104`? Tell `192.168.1.101` at `00:A0:CC:DE:CD:68`." Since this goes out to all hosts on the local network, they can all record the information about the relationship between that source IP and hardware address in a local cache for use later (this is how a host may know the destination information before asking). When the host `192.168.1.104` sees this request, it responds that host (note that it doesn't broadcast) with an ARP response like "I have IP `192.168.1.104`. My hardware address is `00:A9:DE:A8:C5:92`." The source host can now store this relationship in its cache and proceed.

What is also often done is ARP announcements, called **Gratuitous ARP**. A host will send out an ARP request, not intending to receive a reply, but simply to update the ARP caches on any hosts that get the announcement. This is often done by operating systems during start up to help resolve some problems that typically occur. A typical example would be when an ethernet card on a host is replaced. Now the ARP relationship between IP and hardware address has changed, but other hosts on the network still have the old relationship stored in their ARP cache. By broadcasting a gratuitous ARP at start up, these old ARP relationships are updating in the other hosts without them having to wait for the changed hosts to make an actual ARP request.

7.6 Reverse ARP / DHCP

We have discussed the case of a host resolving a logical IP address into a physical hardware address. Now lets consider the opposite resolution. A host comes on the network with a hardware address and wants to know what IP it should have. This is a common case for hosts on a private network like a home network. Back in the day, this was handled, logically enough, by **Reverse ARP**, which would use a server to provide mappings between physical hardware addresses and logical IP addresses. RARP has been obsoleted by **Dynamic Host Configuration Protocol** (DHCP), which in addition to providing an IP, can also push all sorts of configuration information like netmask, default gateway, DNS servers, time servers and more.

DHCP is a client/server protocol, so in order to work, the network must have a DHCP server running on it. This is often handled by a router with DHCP software installed on it, but it can also be handled by a host on the network (most use the same host as the NAT host). The network administrator configures the DHCP server with the data it should push to clients that request an IP, like netmask, default gateway, DNS server, etc. As for IP assignments, DHCP offers several mechanisms, but the two most used are manual and dynamic, which are often used together.

In manual DHCP, the hardware address is stored in a list on the DHCP server and given a static IP to use. When that hardware address requests an assignment from the DHCP server, it will always get the same IP address. This is convenient if you don't want to have to configure all of the clients manually, but still want them to have specific IP addresses.

In dynamic DHCP, the network administrator assigns authority of a pool of free IP addresses to the DHCP server for distribution. When a client requests an assignment, the DHCP server will lease out one of the free IPs in the pool. The lease time is predetermined by the server and usually finite. Lease times can range from an hour to several months (or infinite, if desired). Before the lease expires an host that is still using its lease will need to renegotiate for a new lease.

While it is possible to have multiple DHCP servers on the same network, you need to be careful when allocating free IP pools. You must make sure that the free IP pools assigned to each host have no overlaps or encounter problems. One DHCP server may give out an overlapping IP and it won't let the other DHCP server know. Then if the another DHCP server attempts to give out the same IP, you will have two hosts with the same IP on the network.

Now lets follow the typical conversation that takes place when a DHCP client requests a lease. The client starts by broadcasting a **DHCP discovery** message to the local network. This says something like "This is my hardware address, what is IP address should I use?" Since this is broadcast to the entire local network, any DHCP servers that are listening will get this discovery signal.

The DHCP server (or servers if there are multiple on the network) will send a **DHCP offer** message to the client (not broadcast). The response says something like "I'm offering this IP address and a lease. Do you want to take it?" This offer contains an IP, lease information, and a few key configuration values. The IP could either be for a static IP if the client is in the manual DHCP table, or the server will reserve one of the free IPs in the dynamic pool.

Once the client receives an offer (or several if there are several servers), it will look through them and choose one. Once it picks one, it will broadcast a **DHCP request** message indicating which IP it chose. It says something like "I've decided I want to use this IP address that was offered to me by this DHCP server." Since this is broadcast, all of the DHCP servers that offered an IP will get a response from the client. When the DHCP server that offered the lease gets the response back, it will move into the final phase of the transaction. Any other DHCP servers will get the response, see that the client chose a different server, and withdraw the lease, returning the IP to the free pool if it was reserved.

Now that the chosen DHCP server has received a request from the client, it will go into the final stage of configuration. The server will send a **DHCP acknowledgment** message back the client (not broadcast). This says something like "I've acknowledged the assignment of your IP address to your hardware address. Here is your lease information again and any extra configuration information you may need." While the DHCP offer only contained a few select configuration details, this will contain all of the configuration options that the DHCP server is supposed to distribute. When the client receives this, it applies the given values and configuration is complete.