Packets and Encapsulation

UNIX can support a variety of physical networks, including Ethernet, FDDI, token ring, ATM (Asynchronous Transfer Mode), wireless, and serial-line-based systems. Hardware is managed within the link layer of the TCP/IP architecture, and higher-level protocols do not know or care about the specific hardware being used.

Data travels on a network in the form of packets, bursts of data with a maximum length imposed by the link layer. Each packet consists of a header and a payload. The header tells where the packet came from and where it is going. It can also include checksums, protocol-specific information, or other handling instructions. The payload is the data to be transferred.

The name of the primitive data unit depends on the layer of the protocol. At the link layer it is called a frame, at the IP layer it is called a packet, and at the TCP layer it is called a segment. We will use the term packet as a generic term that encompasses all of these cases.

As a packet travels down the protocol stack in preparation for being sent, each protocol adds its own header information. Each protocol's finished packet becomes the payload part of the packet generated by the next protocol. This nesting is known as encapsulation. On the receiving machine, the encapsulation is reversed as the packet travels back up the protocol stack.

For example, a UDP packet being transmitted over Ethernet contains three different wrappers or envelopes. On the Ethernet wire, it is framed with a simple header that lists the source and next-hop destination hardware addresses, the length of the frame, and the frame's checksum (CRC). The Ethernet frame's payload is an IP packet, the IP packet's payload is a UDP packet, and the UDP packet's payload is the actual data being transmitted.

<table>
<thead>
<tr>
<th>Ethernet Header</th>
<th>IP Header</th>
<th>UDP Header</th>
<th>Application Data</th>
<th>Ethernet CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 bytes</td>
<td>20 bytes</td>
<td>8 bytes</td>
<td>100 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

UDP packet (108 bytes)
IP packet (128 bytes)
Ethernet frame (146 bytes)

The Link Layer

Ethernet Framing Standards

One of the main chores of the link layer is to add headers to packets and to put separators between them. The headers contain the packet's link-layer addressing information and checksums. The separators ensure that receivers can tell where one packet stops and the next one begins. The process of adding these extra bits is known generically as framing.

Two different standards of 10 Mb/s Ethernet framing are in common use: DIX Ethernet II and the IEEE 802.2 LLC SNAP. UNIX hosts generally use Ethernet II, as do Cisco routers. Novell and IPX networks normally use 802.2. Ethernet and 802.2 differ in some fields of the frame header but do not conflict, so
receivers can determine unambiguously which format is being used by each individual packet and decode the header appropriately.

The framing that a machine uses is determined both by its interface card and by the interface card’s driver. On PCs running Windows, you can choose which style of framing that you want, but on UNIX systems you usually cannot. Both types of framing interoperate just fine from UNIX’s perspective.

**Packet Addressing**

Like letters or email messages, network packets must be properly addressed in order to reach their destinations. Several addressing schemes are used in combination:

- MAC (media access control) addresses for use by hardware
- IPv4 and IPv6 network addresses for use by software
- Hostnames for use by people

A host’s network interface may have a link-layer MAC address that distinguishes it from other machines on the physical network, an IP address that identifies it on the global internet, and a hostname that is used by humans.

The lowest level of addressing is dictated by network hardware. For example, Ethernet devices are assigned a unique 6-byte hardware address at the time of manufacture. The 6-byte Ethernet address is divided into two parts: the first three bytes identify the manufacturer of the hardware, and the last three bytes are a unique serial number that the manufacturer assigns. System admins can often identify at least the brand of machine that is trashing the network by looking up the 3-byte identifier in a table of vendor IDs. A current vendor table is available from:

http://www.iana.org/assignments/ethernet-numbers

At the next level up from the hardware, Internet addressing (more commonly known as IP addressing) is used. One 4-byte IP address is assigned to each network interface. IP addresses are globally unique and hardware independent.

The mapping between IP addresses and hardware addresses is implemented at the link layer of the TCP/IP model. On networks that support broadcasting, i.e. networks that allow packets to be addressed to “all hosts on this physical network”, a protocol called ARP, allows mappings to be discovered automatically, without assistance from a system admin.

Since IP addresses are long, seemingly random numbers, they are hard for people to remember. UNIX systems allow one or more hostnames to be associated with an IP address so that users can type

```
$ ssh anchor.cs.clemson.edu
```

instead of
$ ssh 128.139.242.1

This mapping can be set up in several ways, ranging from a static file, i.e. /etc/hosts, to the world-wide Domain Name System. Keep in mind that hostnames are just a shorthand way of writing IP addresses. The /etc/hosts file on one of our Solaris machines is

```
# Internet host table
# 127.0.0.1       localhost       loghost
::1     localhost       loghost
172.19.48.213   CPSC424-White   # Added by DHCP
```

**Ports**

IP addresses identify machines, or more precisely, network interfaces on a machine. They are not specific enough to address particular processes or services. TCP and UDP extend IP addresses with a concept known as a *port*. A port is a 16-bit number that supplements an IP address to specify a particular communication channel. Standard UNIX services such as email, FTP, and the remote login server all associate themselves with well-known ports in the file /etc/services. To help prevent impersonation of these services, UNIX systems restrict access to port numbers under 1,024 to root.

A partial list of /etc/services on the same Solaris system is

```
# Copyright 2008 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
#ident "@(#)services 1.34 08/11/19 SMI"
#
# Network services, Internet style
#
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                       # Secure Shell
telnet          23/tcp
```
<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
<th>Protocol</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtp</td>
<td>25/tcp</td>
<td>mail</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>37/tcp</td>
<td>timserver</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>37/udp</td>
<td>timserver</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>42/udp</td>
<td>nameserver</td>
<td></td>
</tr>
<tr>
<td>whois</td>
<td>43/tcp</td>
<td>nicname</td>
<td># usually to sri-nic</td>
</tr>
<tr>
<td>domain</td>
<td>53/udp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bootps</td>
<td>67/udp</td>
<td></td>
<td># BOOTP/DHCP server</td>
</tr>
<tr>
<td>bootpc</td>
<td>68/udp</td>
<td></td>
<td># BOOTP/DHCP client</td>
</tr>
<tr>
<td>kerberos</td>
<td>88/udp</td>
<td>kdc</td>
<td># Kerberos V5 KDC</td>
</tr>
<tr>
<td>kerberos</td>
<td>88/tcp</td>
<td>kdc</td>
<td># Kerberos V5 KDC</td>
</tr>
<tr>
<td>hostnames</td>
<td>101/tcp</td>
<td>hostname</td>
<td># usually to sri-nic</td>
</tr>
<tr>
<td>pop2</td>
<td>109/tcp</td>
<td>pop-2</td>
<td># Post Office Protocol - V2</td>
</tr>
<tr>
<td>pop3</td>
<td>110/tcp</td>
<td></td>
<td># Post Office Protocol - Version 3</td>
</tr>
</tbody>
</table>

**Address Types**

At both the IP layer and the link layer, there are several different types of addresses:

- **Unicast** -- addresses that refer to a single host, i.e. network interface
- **Multicast** -- addresses that identify a group of hosts
- **Broadcast** -- addresses that include all hosts on the local network
- **Anycast** -- addresses that resolve to any one of a group of hosts

Multicast addressing facilitates applications such as video conferencing in which the same set of packets must be sent to all participants. Multicast is largely used on today’s Internet to stream television and movies. IPv6 broadcast addresses are really just specialized forms of multicast addressing.

Anycast addresses bring load balancing to the network layer by allowing packets to be delivered to whichever of several destinations is closest in term of network routing. You might expect that they would be implemented similarly to multicast addresses, but in fact they are more like unicast addresses.

The value assigned for IP multicast on the Internet is 01:00:5E. The link-layer broadcast address is all 1s in binary or ff:ff:ff:ff:ff in hexadecimal. At the IP layer, multicast addresses begin with a byte in the range 224 to 239. Broadcast addresses have a host part that is all 1s.