How the Internet Works

TCP/IP Protocol Suite

We now consider the TCP/IP protocol suite in detail. Note: we will use *Internet* (with a capital "I") to denote the Connected TCP/IP Internet, and internet (with a small "i") when talking about standalone TCP/IP internets that aren't connected to the rest of the world. The Internet protocol suite covers (mostly) layers 3, 4, and 5, where layer "5" means everything in OSI layers 5-7. At the physical and datalink layers, the TCP/IP protocols don't define any standards. Indeed, as we shall see, the protocols have been designed to operate over a large number of layer 2 protocols.

The *Internet Protocol* (IP) is a network layer protocol.

Hosts and gateways process packets called *Internet datagrams* (IP datagrams).

IP provides connectionless, best-effort delivery service.

The *Transmission Control Protocol* (TCP) is a transport layer protocol that provides reliable stream service between processes on two machines. It is a sliding window protocol that uses acknowledgments and retransmissions to overcome the unreliability of IP.

The *User Datagram Protocol* (UDP) provides connectionless datagram service between machines.

Application protocols include:

SMTP: The Simple Mail Transfer Protocol is used to send mail from one machine to another.

Telnet: Provides remote login service. It allows a user on one machine to log into another machine on the network.

FTP: The File Transfer Protocol copies arbitrary files (e.g. binary, data, and source) from one machine to another.

HTTP: Basic Web protocol.

ssh: secure shell and file access.

Internet Addressing

Host identifiers are classified as *names*, *addresses*, or *routes*, where:

A name suggests *what* object we want.

An address specifies *where* the object is.

A route tells us *how* to get to the object.

In the Internet, names consist of human-readable strings such as *eve*, *percival*, or *cs.wpi.edu*.

Addresses consist of compact, 32-bit identifiers. Internet software translates names into addresses; lower protocol layers always uses addresses rather than names.

Internet addresses are hierarchical, consisting of two parts:

network: The network part of an address identifies which network a host is on.
Conceptually, each LAN has its own unique IP network number.

local: The local part of an address identifies which host on that network.

Later, we'll examine a technique called *subnetting* that adds a third level to the hierarchy. With subnetting, the local part may consist of a "site", which is further broken down in to local network number, local host.

Conceptually, the Internet consists of a collection of physical networks, each of which is assigned a unique number. As datagrams travel from one gateway to another, each gateway routes the datagram based on the network number in the datagram's destination address.

Only the gateway on the same network as the destination uses the local part of the address in forwarding a datagram. That is, when the datagram reaches a gateway that connects to the destination address, the gateway uses the local part of the address to forward the datagram to the appropriate host.

Traditional Address Classes

The Internet designers were unsure whether the world would evolve into a few networks with many hosts (e.g., large networks), or many networks each supporting only a few hosts (e.g., small networks). Thus, Internet addresses handle both large and small networks. Internet address are four bytes in size, where:

1. Class A addresses start with a "0" in the most significant bit, followed by a 7-bit network address and a 24-bit local part.
2. Class B addresses start with a "10" in the two most significant bits, followed by a 14-bit network number and a 16-bit local part.
3. Class C addresses start with a "110" in the three most significant bits, followed by a 22-bit network number and an 8-bit local part.
4. Class D addresses start with a "1110" in the four most significant bits, followed by a 28-bit group number.

Note: The use of fixed-sized addresses makes the routing operation efficient. In the ISO world, addresses are of varying format and length and just extracting the address from the packet may not be straightforward.

Internet addresses can also refer to broadcast addresses. The all 1's address is used to mean "broadcast on this network". Of course, if the underlying network technology doesn't support broadcasting, one can't broadcast Internet datagrams either.

Network addresses are written using dotted decimal notation. Each address consists of 4 bytes, and each byte is written in decimal form. Sample addresses:

- *wpi.wpi.edu*: 130.215.24.6 (class B)
- *owl.wpi.edu*: 130.215.8.139 (class B)
- *wpi.edu*: 130.215 (a network address)
- *rialto.mcs.vuw.ac.nz*: 130.195.5.15 (class B)
- *gwen.cs.purdue.edu*: 128.10.2.3 (class B)
- *c.nyser.net*: 192.33.4.12 (Class C)
- *pescadero.stanford.edu*: 36.8.0.8 (class A)
- *su-net-temp*: 36 (network address)

Note: Internet addresses refer to network connections rather than hosts. Gateways, for instance, have two or more network connections and each interface has its own IP address. Thus, there is not a one-to-one mapping between host names and IP addresses.

Internet addresses are hierarchical addresses. Datagrams are initially routed only by network number, and only the gateway connected to the destination network uses the local part while performing the routing operation.

What happens to a host's internet address if it moves from one network to another? Its Internet address must change. Now we get a better appreciation for why one wants to distinguish between a machine's name and its address. Physical address is constant, network address must change.

Network Byte Order

One problem that arises when interconnecting different machines is that different machines represent integers in different ways:

- *Big Endian* machines such as IBM and Sun computers store the most significant byte of a 32-bit integer in the lowest memory address of the word (e.g. to the left).

  That is, the integer 0x01020304 is laid out in memory as bytes 0x01, 0x02, 0x03, and 0x04.

- *Little Endian* machinessuch as Intel store the most significant byte at the highest address.

  That is, the integer 0x01020304 is laid out in memory as bytes 0x04, 0x03, 0x02, 0x01.

- Other machines (such as DEC-10s) use 36-bit words to hold integers.

As with all network protocols, the protocols specify the meanings of all bits in each field, right down to specifying the bit and byte order. The Internet defines a *network standard byte order* that is used when referring to the fields of Internet datagrams, and the Internet specifies the use of Big Endian form.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <ctype.h>

main(int argc, char **argv)
{
    struct sockaddr_in sin;
    struct servent *ps;
    struct hostent *ph;
    char sbHost[128];
    long l;

    gethostname(sbHost, 128);
    ph = gethostbyname(sbHost);
    ps = getservbyname("finger", "tcp");
    bcopy(ph->h_addr, (char *)&l, sizeof(long));
    printf("%s: %x:%d\n", sbHost, l, ps->s_port);
```

```
    printf("%s: %x:%d (network order)\n", sbHost, ntohl(l), ntohs(ps->s_port));
}

% ./byteorder   (on Intel/AMD)
sequoia.wpi.edu: f05c382:20224
sequoia.wpi.edu: 82c3050f:79 (network order)

% ./byteorder   (on Sun)
crane.wpi.edu: 82c30412:79
crane.wpi.edu: 82c30412:79 (network order)
```

Mapping Between Internet and Physical Addresses

Suppose we have two machines A and B connected to the same network, and A wants to send an internet datagram to B. A must know B's data link layer address in order to send frames to B.

The problem of mapping Internet addresses to physical addresses is known as the *address resolution problem.*

There are two classes of physical addresses, typified by the following examples. The key distinction is whether the physical address is small enough that it can be encoded in the local part of an internet address.

Ethernet addresses are large (48-bit) fixed-size addresses.

ProNET-10 addresses are small (8-bit) fixed size addresses.

The proNET-10 is a 10Mbps LAN ring network that uses 8-bit source and destination addresses. The network administrator assigns the physical address of each new station added to the ring, and no two stations on a ring share the same address.

Moreover, a site administrator is free to choose the local part of the IP host address, setting it to be the same as the LAN's station number. For example, a machine with a network interface of station number 54 could have an internet address of 128.204.6.54 or 192.12.53.54. Thus, mapping an Internet address to a physical address consists of extracting the relevant bits from the IP address.

Unfortunately, address resolution is more complex for networks such as Ethernets:

1. Each Ethernet device has its own unique address. Replacing a host's Ethernet card changes its physical address.
2. Physical address are 6 bytes long, too large to encode within an Internet address.
3. New machines can be added to the network with no disruption of service. Thus, adding new hosts should not require reconfiguring existing hosts to inform them of the new machine.

ARP

The *Address Resolution Protocol* (ARP) is a protocol that allows hosts to dynamically map Internet addresses to physical addresses:

1. The requesting machine only needs to know the target machine's IP address.
2. It sends out a special ARP request frame using the Ethernet's broadcast capability. Thus, every machine on the LAN will receive the ARP request.
3. The ARP request asks "what is the Ethernet address of Internet address X"?
4. Each machine receives a copy of the broadcast message, and the machine having the desired IP address responds with its Ethernet address.

Of course, a machine doesn't send out an ARP packet each time it wishes to send an IP datagram. Instead, each machine maintains a cache of recently used mappings, and an ARP request is only sent if the desired mapping is not already in the cache.

ARP request packets also contain the sender's IP and Ethernet address pair. Why? To eliminate the need for a second ARP request. If *machine A* wishes to communicate with *machine B*, there is high probability that *B* will need *A*'s Ethernet address as well.

Since every machine receives every ARP request (which is broadcast), how about adding the source address in each ARP request to the cache? It turns out that this is not a terribly good idea. Although a network may consist of hundreds of machines, a given host is unlikely to actively communicate with more than a few at any one time. Thus, adding every mapping to the local cache is likely to waste memory, and may cause the flushing of entries that will be used again soon to make room for entries that will never be used.

Compromise: Upon receipt of an ARP request from a machine whose IP address is already in the local ARP cache, update the information for that entry. This handles the case of a machine whose Ethernet address changes; ARP entries with the old value will be overwritten with the new value.

From a layering point of view, ARP sits below IP, but above the data link layer.

ARP Details

Conceptually, ARP consists of two parts:

the software responsible for finding the physical address of an IP address (e.g., a client), and

the software responsible for answering ARP requests from other machines (e.g., a server).

When sending an IP datagram, the sender searches its local ARP cache for the desired target address:

If we find a match, we are done.

Otherwise, send out a broadcast ARP request and wait for the response.

In practice, waiting for a response is somewhat tricky, because the target machine may be down, the request might become lost and need to be retransmitted, and so forth.

ARP packets are encapsulated in Ethernet frames. Why is the 16-bit type field needed? So that the Ethernet device driver software can distinguish frames carrying ARP packets from those carrying IP datagrams. ARP packets are passed to the ARP module, while IP packets are handed to the software that processes IP.

ARP packets have been designed in a general way so that the protocol can be used over many different network technologies. ARP packets have the following format:

1. The 2-byte *H-Type* field gives the type of the hardware address we are interested in (e.g., 1 for Ethernet).
2. The 2-byte *P-Type* field gives the type of the higher level protocol address we are interested in (e.g., 0x0800 for IP). Note, it is two bytes long, just like the Ethernet type field. Is this a coincidence?
3. A 1-byte *H-Len* field specifying the length of the hardware address (6 bytes would be the length for Ethernet).
4. A 1-byte *P-Len* field specifying the length of the target protocol address (4 for IP).
5. A 16-bit *Code* field specifying the operation desired (e.g., REQUEST or RESPONSE).
6. The sender's Ethernet address (Sender HA) (if known).
7. The sender's Internet address (Sender PA) (if known).
8. The target's Ethernet address (Target HA) (filled in response).
9. The target's Internet address (Target PA) (filled in response).

Reverse ARP

ARP handles the case of determining the hardware address that corresponds to an IP address. When is it necessary to map hardware addresses back into IP addresses?

When a diskless machine first boots, it doesn't know its own IP address (and can't read it from a local disk!). How can a booting station get started?

Have the booting client contact a server to obtain its Internet address. Three problems:

1. How can the client communicate with a server, if it doesn't yet know its own IP address? Use a special protocol that uses only Ethernet frames. This bootstrapping protocol will only be used during the booting phase; once we have an IP address, subsequent communication uses IP protocols.
2. Which server should it ask? Use Ethernet broadcasting to ask all machines; only the server will respond.
3. What identifier can the client use to identify itself to the server? (And consequently, so the server knows which machine is booting and wants its IP address?)

   When a diskless workstation boots, its Ethernet address is the only piece of information available to it *before* it has booted.

The protocol that maps hardware addresses to Internet addresses is called *Reverse ARP*, or RARP. The RARP server maintains a database of physical address to Internet address mappings. The actual format of RARP messages is similar to those of ARP:

The Ethernet frame type is set to type RARP (0x8035), and

RARP defines two new message types, "RARP request" and "RARP response". The remaining fields are the same as in ARP.

Note: We now see one of the primary benefits of broadcasting: locating servers.

However, because broadcasting is resource intensive, (every machine on the local network must process the message, even if only to determine that it isn't interested in it) broadcasting should be used sparingly.

Dynamic Host Control Protocol (DHCP)

RARP has largely been replaced by DHCP.

Allows dynamic host configuration so parameters such as default gateway and DNS server do not need to be manually configured.

Also allows a pool of IP addresses to be maintained by a DHCP server and assigned as hosts request them. Both used in LAN environment and also by ISPs.

A *lease* is associated with an IP address that must be renewed when the lease expires. DHCP servers may or may not maintain "stickiness" between an Ethernet and IP address.

DHCP server maintains a lookup based on Ethernet address sent as part of client request. At WPI this address must be registered so not just any machine can use the campus wired or wireless network.