How the Internet Works

Andrew McNabb
http://www.mcnabbs.org/andrew/

September 2, 2004

The Mystical Internet

Most drivers have only a basic understanding about how cars work. They know how to use steering wheels, pedals, and sometimes clutches, but they know little about engines, starters, and transmissions. Carburators, catalytic converters, and differentials are mystic buzzwords. Many people feel there is no need to know anything about cars other than how to drive them, but even a moderate understanding provides several benefits. For example, a more knowledgeable car owner can communicate with mechanics, perform basic repairs and maintenance, and avoid unnecessary damage. A driver may decide to learn how cars work for these practical reasons or simply to satisfy a desire for knowledge.

Computers and networking are much less mature fields than auto mechanics. As a result, consumers have a weaker understanding of how they work. At the same time, however, they have a greater need for comprehending computers and networking. Knowledge about how the Internet works will help users to solve problems, be aware of security issues, and avoid confusion as they use the Internet.

Understanding the Internet is about understanding its protocols. Starting from the bottom and moving up are Ethernet, IP, UDP and TCP, DNS, SMTP, and HTTP. While these are only a fraction of the networking protocols in existence, they are the most commonly used and the most essential in the infrastructure of the Internet.

Protocols

A *protocol* is a precisely defined procedure for communication. When several systems speak the same protocols, they can transfer information. Internet protocols began in the late 1970s and have been developing over time. Specifications have been coordinated by the Internet Engineering Task

Force (IETF), which publishes Requests for Comment (RFCs) containing detailed descriptions of protocols.

The protocols of the Internet are layered, each protocol building on others. This allows for flexibility, since new systems can be developed without completely reinventing the wheel each time. Imagine what it would be like if every time publishers created a new magazine or catalog, they had to organize a fleet of horses and riders (or invent an automobile), pave roads, and design a nationwide addressing system. Fortunately there would be less junk mail, but each issue of a magazine would cost hundreds of dollars. Since publishers use the postal system, and the postal system uses cars and trains, and cars and trains use highways and rail systems, they can send magazines quickly and cheaply. The same principle applies to the Internet. If everyone can agree on the same fundamental protocols, then communication from one end of the world to the other is cheap and fast (usually under a fifth of a second, round trip).

Ethernet

Ethernet (officially named IEEE 802.3) is the most common protocol used at the physical and datalink level. It dictates exactly what is allowed to go over the wire and how computers communicate at a very low level. Ethernet is popular because it is easy to administer and inexpensive [Peterson and Davie, 2000, pages 116, 124]. It allows computers on the same *Local Area Network* (LAN), all connected directly together, to communicate with each other.

Electronics tend to struggle with natural languages like English, Serbian, and Japanese. Given only a wire and electricity, it is difficult to transmit information such as a sentence without a clearly defined encoding. For example, before communication systems were able to transmit the sound of the human voice, operators communicated over wires using Morse Code, which encoded letters as series of long and short beeps. Analogously, computers generally encode all information

as series of 0s and 1s. In network communication, they must encode the 0s and 1s as electric signals using two voltage levels. It would be intuitive if a 1 were represented by a high voltage and 0 by a low voltage, in what is known as *non-return to zero* (NRZ) encoding. However, as is so often the case, the intuitive solution turns out to have major weaknesses like "baseline wander" and lack of "clock recovery" (ask your favorite electrical engineer for more information). It turns out

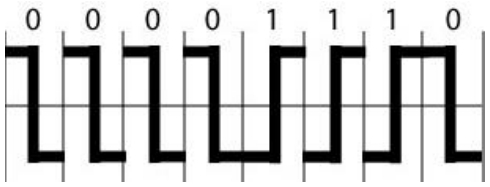

Figure 1: The Manchester Encoding [WildPackets, 2004]

that an encoding based on the voltage *change* is much more reliable than one based on the voltage *level* [Peterson and Davie, 2000, pages 80–83]. The *Manchester encoding* represents a 1 as a low to high voltage change and 0 as a high to low change (see Figure 1). Since the signal has a transition in the middle of every bit-time, the signal stays synchronized and undistorted [WildPackets, 2004].

Just as computers must use an encoding to represent 1s and 0s, they must find a way to represent entire messages. An Ethernet *frame* is a short, manageable message. Each frame includes headers, which give information about the frame. An Ethernet frame has a maximum length of about 200 English words (1518 bytes). When a longer message needs to be sent, the system breaks it into smaller chunks, puts a header on each of them, and treats them as independent frames (see Figure 2).

An Ethernet network is like a disorganized conversation. Everyone waits for a break in the chatter and then shouts out what they have to say. If two people try to speak at the same time, neither one is intelligible, so they give up briefly and wait for another break in the conversation to try again. All computers in an Ethernet network are physically or logically on the same wire. When one of them needs to send a message, it waits for 9.6 microseconds of silence on the wire. It then sends the frame *preamble*, which is eight bytes of alternating 1s and 0s, ending in two 1s. The

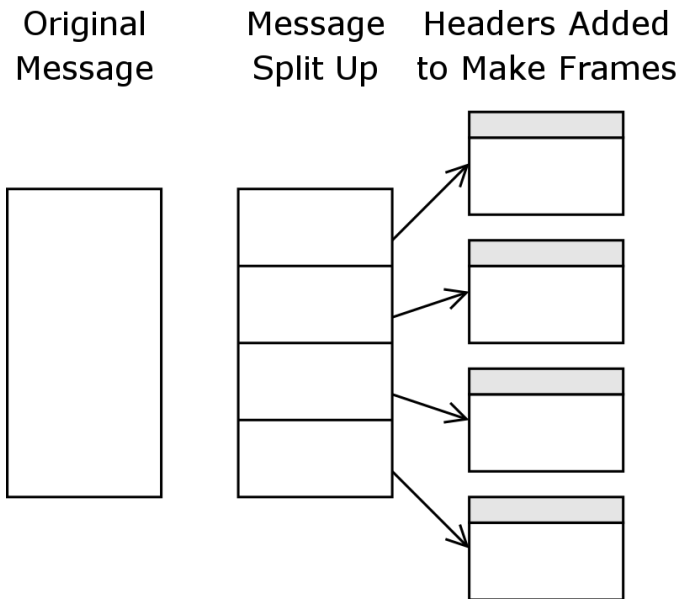

Figure 2: A large message is split into manageable frames.

preamble allows everyone on the network to synchronize and start listening. It is the equivalent of shouting, “Hey! I have something to say.” The computer then sends the rest of the frame, constantly listening for any interruptions which would force it to announce (by sending a lot of 1s) that the transmission failed, wait a few microseconds, and start all over. When the wire finally goes silent, another computer can take over [WildPackets, 2004].

The encodings and transmission etiquette make up the physical aspect to the Ethernet protocol. The rest of Ethernet is devoted to datalink coordination, solving the problem of how to know who should listen to a particular frame and who should just ignore it. After the preamble, the frame contains a destination address followed by a source address, which are analogous to the “to” and “from” lines on a postcard. These fields specify the six-byte *MAC Address* (Medium Access Control Address), the pre-assigned unique hardware address of each network device, of the recipient and sender. A typical MAC address looks like 00:01:02:e4:2d:4c[1]. Since every device knows its own MAC address, it can quickly look at each frame and decide whether or not to listen to it. Note that although most systems will reject all frames addressed to other devices, anyone

[1]MAC addresses are written for humans in hexadecimal (base 16) with colons separating each byte.

is free to enter *promiscuous* mode and look at every single frame on the wire. In addition to the destination and source addresses, the frame header includes a field which shows what type of data is included in the body of the frame.

Ethernet is the first building block of the Internet. It allows physically connected computers to communicate, even if only in a very primitive way. All of the other protocols will build on Ethernet.

IP and Routing

The *Internet Protocol* (IP) is the protocol which allows messages, known as *packets*, to find their way across more complex networks like the Internet. Unlike Ethernet, which works by broadcasting messages to everyone including the recipient, IP allows messages to be routed to the correct destination.

Every computer on an IP network has a four-byte *IP Address*, which is similar to a MAC address except that instead of being built into the hardware it is configured for each system. A typical IP address looks like 128.187.168.21[2]. In addition to the IP address, computers on an IP network must have the following configured: default gateway, network mask, and broadcast address [Linux Programmer's Manual, 2000].

An IP packet contains an IP header, just as an Ethernet frame contains an Ethernet header. The most important fields in the IP header are the source and destination IP addresses and the protocol, which tells what type of data is included in the body of the packet. The header also contains information such as the header length, the total length of the packet, the identification number, the time-to-live, and the checksum, which are all essential for communication but less relevant at this

[2]IP addresses are written in *dotted decimal* form, meaning that each byte is written in decimal (base 10), and the bytes are separated by periods.

level of analysis [Stevens, 1998, pages 883–885].

Sending an IP packet is more difficult than just broadcasting it on the wire, but it is still relatively simple. There are two main cases: either the recipient is on the same LAN, or it isn't. When the computer is configured for the network, it is told which range of IP addresses are on the local network and which are not[3], so it can easily distinguish the two cases.

In the case that the recipient is on the same LAN, the *Address Resolution Protocol* (ARP) is what lets IP use Ethernet. If a computer with the IP address 10.0.0.4 and the MAC address 01:02:03:04:05:06 needs to send a message to the computer at the IP address 10.0.0.5, it needs to figure out what MAC address corresponds to it. To do this it broadcasts an *ARP Request* which reads, "Who has 10.0.0.5? Tell 01:02:03:04:05:06." The other computer picks up the ARP request, realizes that it has the IP address in question, and sends back an *ARP Reply* stating, "10.0.0.5 is at 0a:0b:0c:0d:0e:0f." Then the sender puts an Ethernet header on the IP packet, with a destination MAC address of 0a:0b:0c:0d:0e:0f, and puts the frame on the wire. Finally the recipient picks up the frame and processes the packet [Ethereal, 2004].

If the recipient is not on the same LAN, then the sender must relay the message through a *gateway*. A gateway[4] is a device which sits on two separate networks and relays packets between the two. The sending computer is configured with the IP address of the default gateway, so it sends out an ARP request to find the gateway's MAC address. Finally it makes an Ethernet frame out of the IP packet and sends it to the gateway, which it trusts to make sure the packet finds its way to the correct destination.

The Internet is a complex network of networks, and routing packets is no simple task. Systems such as the *Border Gateway Protocol* (BGP) allow routers to discover the topology of the network so that they know where to send the packet next. As a packet makes its way from the source to the

[3] The *network number*, which may look like 192.168.1.0, and the *subnet mask*, which is often 255.255.255.0, together indicate which addresses are on the local network.

[4] A *firewall* is a special type of gateway which discards packets it doesn't like, generally for security.

```
traceroute to slashdot.org (66.35.250.150)
 1  192.168.1.1 (192.168.1.1)  2.863 ms
 2  10.190.144.1 (10.190.144.1)  13.955 ms
 3  12.244.116.97 (12.244.116.97)  11.645 ms
 4  12.244.66.137 (12.244.66.137)  17.349 ms
 5  12.244.72.162 (12.244.72.162)  21.500 ms
 6  gar1-p360.sc1ca.ip.att.net (12.122.2.242)  34.013 ms
 7  tbr1-p012102.sffca.ip.att.net (12.122.2.246)  39.524 ms
 8  ggr2-p300.sffca.ip.att.net (12.123.13.190)  56.764 ms
 9  att-gw.sf.cw.net (192.205.32.110)  39.255 ms
10  dcr2-loopback.SantaClara.savvis.net (208.172.146.100)  42.841 ms
11  bhr1-pos-0-0.SantaClarasc8.savvis.net (208.172.156.198)  50.261 ms
12  csr1-ve243.SantaClarasc8.cw.net (66.35.194.50)  38.903 ms
13  66.35.212.174 (66.35.212.174)  50.943 ms
14  slashdot.org (66.35.250.150)  53.416 ms
```

Figure 3: Abridged output from a traceroute to slashdot.org, demonstrating the routing of packets. Each line represents the next hop along the route [Traceroute, 2000].

destination, it is forwarded from router to router until it finally arrives [Peterson and Davie, 2000]. Most packets sent across the Internet make it to their destination in 15 to 20 hops, all within about 200 milliseconds. Figure 3 shows the path of a packet. Note that the routers don't necessarily use Ethernet, so while IP usually runs over Ethernet, an IP packet crossing the Internet may temporarily be sent on top of some other protocol.

IP makes it possible for computers on opposite sides of the world to communicate, but it doesn't do everything. For example, IP makes no guarantee that a packet will actually arrive safely, so sometimes packets will get lost and need to be retransmitted.

UDP and TCP

Ethernet sends messages to computers on the local network, and IP sends messages to computers across the Internet, but even IP is too low-level of a protocol for applications to use. With IP, the packet gets delivered to the remote computer, but how does the computer know which program should process it? The UDP and TCP protocols work over IP and enable applications to communicate.

The *User Datagram Protocol* (UDP) is a very simple connectionless protocol that allows programs

to send *datagrams* (data telegrams), which are short self-contained messages. UDP only ensures that datagrams are not corrupted and that they go to the correct program when they reach the remote computer. Like IP, UDP does not guarantee that datagrams will always arrive—it is possible for them to be lost in transit [Stevens, 1998, NetworkSorcery, 2004].

UDP uses *port numbers* to deliver datagrams to the appropriate program. When a program is written, the author will often pick a default port number to use. Many sites on the Internet have lists of officially or unofficially reserved port numbers. The UDP header for each datagram includes fields for the source port and the destination port. Each UDP application informs the computer which port it needs to use. When a datagram arrives, the system looks at the UDP header and delivers the datagram to the program which is registered to the same port number as appears in the destination port field.

The *Transmission Control Protocol* (TCP) is a complex protocol which works over IP. Unlike UDP, which is connectionless and sends independent datagrams, TCP is a connection-oriented protocol which sends streams of data split up into segments (which are invisible to the application). TCP is a reliable protocol—it ensures that no data get lost. It also provides flow control and congestion avoidance; in other words, TCP pays attention to the status of the network and decides what size of segments to send and how frequently to send them. For most applications TCP is preferable to UDP [Stevens, 1998].

Since TCP defines explicit connections, a session must be established before any real data get sent. While the server program is listening on its port, the client program on the other computer opens a port and requests a connection to the remote system and port. TCP performs a *three-way handshake* to establish the connection and then transmits data back and forth between the two programs. Finally, when one of the programs requests that the connection be closed, the two computers perform a TCP connection termination process [Peterson and Davie, 2000].

UDP is widely used in applications where small amounts of lost data are irrelevant, such as stream-

ing audio and video, where by the time a retransmission could arrive it would be too late to be useful. Overall, TCP is much more commonly used than UDP. TCP is used for transferring web pages, sending email, and many other applications.

DNS

The *Domain Name System* is the protocol which converts human-readable *domain names*, such as “www.yahoo.com” or “mork.uni.uiuc.edu” to their corresponding IP addresses, 66.94.230.42 and 128.174.26.251. Without DNS, the Internet would be much less usable. Imagine having to memorize the four-byte number for every web site.

DNS is a hierarchical system made up of *domains*. The most common *top-level domains* in the US are “com,” “org,” “edu,” “gov,” and “net.” There are also top-level domains for other countries, such as “uk,” “jp,” and “yu.” Underneath these top-level domains are second-level domains such as “yahoo.com” and “byu.edu” [ICANN, 2003].

DNS servers are spread throughout the Internet, and they cache (or remember) DNS entries. Several special DNS servers, known as *root servers*, keep track of which servers are authoritative for which domains. Using the DNS protocol, servers communicate with each other and clients make requests to servers. DNS uses both TCP and UDP to transfer information.

While DNS is fascinating, in this context it is most important to understand that DNS is the reason domain names work, and that without them everyone would have to remember the full IP addresses.

Email and SMTP

Email on the Internet is sent using the *Simple Mail Transport Protocol* (SMTP). SMTP uses TCP for its transport, so the protocol hierarchy is SMTP on TCP on IP on Ethernet. SMTP defines commands that are sent back and forth between the client and the server. Mail messages also include standardized headers that show the path the message has taken in addition to who the sender is, who the recipient is, the date of sending, the subject of the message, etc.

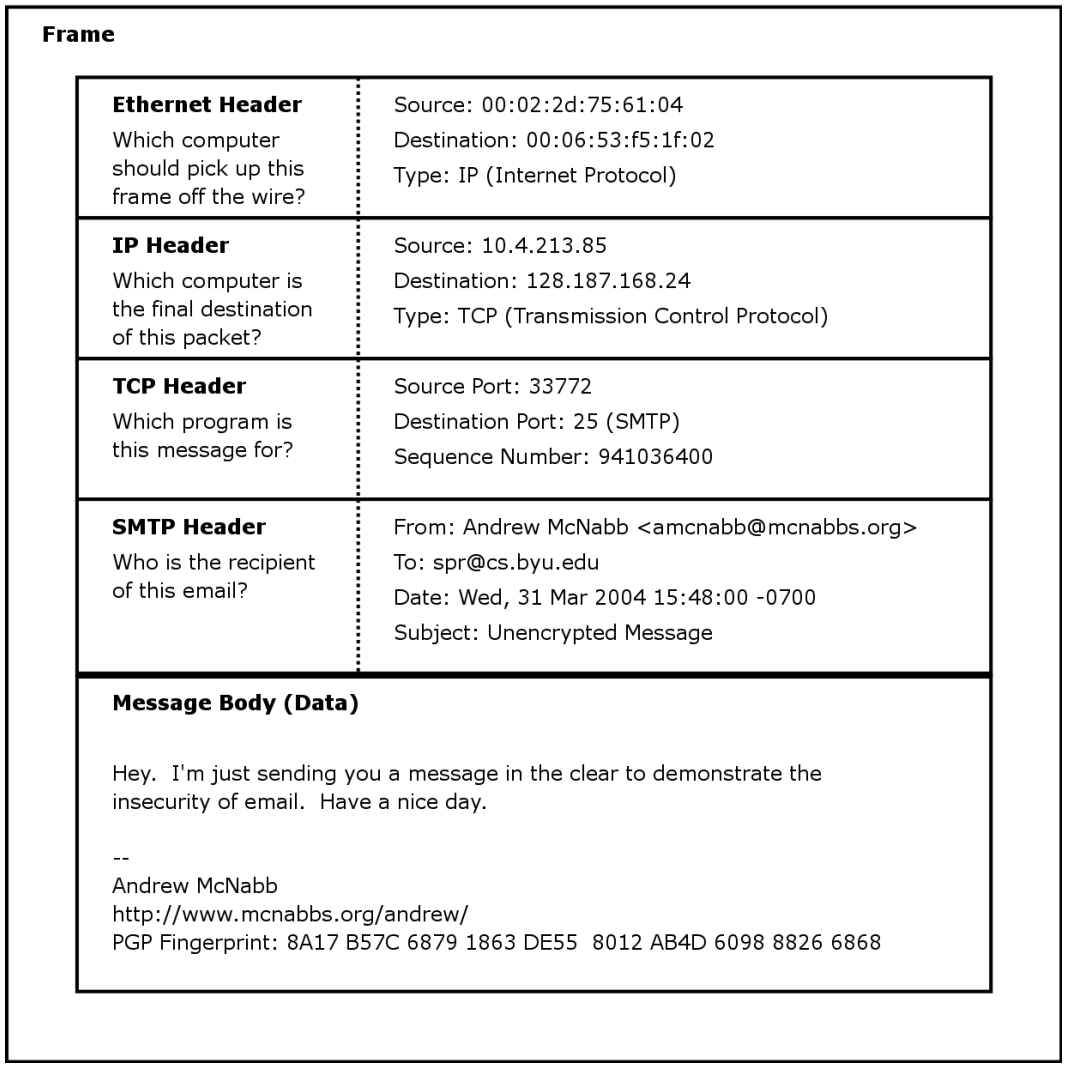

Figure 4: An Ethernet frame containing an SMTP session's TCP segment which contains an email message.

When a computer wants to send an email message, it uses DNS to find the appropriate mail server

for the given email address. It then establishes a TCP connection with port 25 of the server, since 25 is SMTP's port. The client and the server then make introductions by issuing a "HELO" or "EHLO" command. To send the actual message the client uses three commands: "MAIL FROM" to identify the sender, "RCPT TO" to specify the recipient, and "DATA" to send the contents of the message [NetworkSorcery, 2004]. Figure 4 shows an Ethernet frame containing part of an SMTP session. This TCP segment holds the message body, which is one part of the process of sending a message [Ethereal, 2004].

SMTP is not a secure protocol. Anyone who sees the Ethernet frame sees the entire contents of the email message. Also, no authentication is performed, so anyone can send a message appearing to be from anyone else.[5] Most people who use email aren't careful about what they send and are too trusting of everything they receive.[6] Users should learn to sign and encrypt their email, using systems such as PGP [Kaufman et al., 2002].

HTTP

The *Hyper Text Transfer Protocol* (HTTP) is the protocol used by web browsers to download web pages. It is similar to SMTP in that it works over TCP and uses commands.

When a user requests a web page, the browser connects to the web server at port 80 (which is used for HTTP) and requests the web page. Inside the web page are references to files like images and style sheets, so after downloading the page and processing it, the browser issues additional requests. If there are several files to download it will probably establish several connections so that it can download the files in parallel. After everything is downloaded it renders and displays the

[5] Sending a fake email message is incredibly easy. Just connect to an SMTP server and give "HELO," "MAIL FROM," "RCPT TO" and "DATA" commands and send a message with normal-looking headers. Fake messages are generally easy to trace by those who understand the mail headers, such as network administrators, but most people can't tell the difference.

[6] Viruses always send fake email messages, and trusting users are happy to open them and propagate the problem.

page.[7]

Requests over HTTP are most often made using the "GET" command. Along with the command, the client will usually send some HTTP headers, which specify its capabilities. When the server receives the command, it sends a response beginning with HTTP headers which describe the file, followed by the file itself [NetworkSorcery, 2004].

Like SMTP, HTTP is not a secure protocol. Anyone can read what is going over the wire, including passwords and credit card numbers. Because of this, a variant, known as *Secure HTTP* or HTTPS was developed, which sends HTTP traffic over the *Secure Sockets Layer* (SSL) protocol. In most cases, information sent over HTTPS is safe from eavesdroppers.

Conclusion

The Internet is a network of networks which amazingly manage to communicate despite the wide variety of operating systems and communication technologies that comprise it. The Internet is flexible and useful, even with its weaknesses. People who understand how cars work are better equipped to maintain them than mere drivers. Similarly, computer users who have a solid understanding of the Internet are better at solving problems and maintaining their privacy than people who merely point and click.

References

[Ethereal, 2004] Ethereal (2004). Ethereal: A network protocol analyzer. http://www.ethereal.com/. Ethereal was used to provide specific examples of network

[7] Sometimes browsers will display the page before all of the files are completely downloaded and updates the page as it receives more information.

communications.

[ICANN, 2003] ICANN (2003). Top-level domains. http://www.icann.org/tlds/.

[Kaufman et al., 2002] Kaufman, C., Perlman, R., and Speciner, M. (2002). *Network Security: Private Communication in a Public World*. Prentice Hall, Upper Saddle River, NJ.

[Linux Programmer's Manual, 2000] Linux Programmer's Manual (2000). Ifconfig(8): configure a network interface. /usr/bin/man 8 ifconfig.

[NetworkSorcery, 2004] NetworkSorcery (2004). RFC Sourcebook. http://www.networksorcery.com/enp/default0601.htm.

[Peterson and Davie, 2000] Peterson, L. L. and Davie, B. S. (2000). *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, San Francisco, CA.

[Stevens, 1998] Stevens, W. R. (1998). *UNIX Network Programming*, volume 1. Prentice Hall, Upper Saddle River, NJ.

[Traceroute, 2000] Traceroute (2000). Traceroute: print the route packets take to network host. ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[WildPackets, 2004] WildPackets (2004). Technical compendium: Technology, engineering, networking. http://www.wildpackets.com/compendium/main/index.html.