THE INTERNET AND WORLD WIDE WEB

The goal of this lesson is to set in place a general understanding of the history of the Internet and World Wide Web (WWW) and, more importantly, to explore how these amazing technologies work. In the process of exploring the fundamental concepts, we provide an important foundation of vocabulary relating to the Internet and WWW. It is interesting to note that many people, including prominent news commentators, use the terms Internet and WWW interchangeably as if they are the same thing. But they are very different!

1. VERY BRIEF HISTORY OF COMPUTERS

In the 1600s, computing machines made from mechanical components (metal levers and gears) could do simple arithmetic (add, subtract, multiply, divide). Some of them had cool names like the *Stepped Reckoner* and the *Difference Engine*. In the 1800s, Charles Babbage built the *Analytical Engine*, which is widely recognized as the first mechanical computer that could do logical operations more like electronic computers that would be invented about 100 years later. In the mean time, mechanical "computers" actually had some practical uses. Perhaps you recall seeing purely mechanical cash registers (cha-ching!) in old movies set in the early-mid 1900s.

Wars are powerful motivators for technology. The necessity for calculating long-distance trajectories for artillery shells in World War II lead to the research that inspired the first electronic computers. The ENIAC (*Electronic Numerical Integrator And Calculator*) is the most widely known among the very first digital computers. It was developed at the University of Pennsylvania around 1946, contained 18,000 vacuum tubes, weighed 60,000 pounds (30 tons), and occupied as much floor space (1,600 square feet) as a moderately sized house.

In the 1950s, solid-state transistors started replacing vacuum tubes to control electric current, and the size and cost of computers started to decrease. By the 1960s, smaller computers called *minicomputers* were inexpensive enough (around $20,000) so that most colleges and universities could afford one. Ironically, minicomputers were about the size of refrigerators, so they were not even close to "mini" by today's standards. But the era of minicomputers brought important advances such as *time sharing,* where multiple programs could be run simultaneously on one computer by sharing processor time. Prior to time sharing, a computer could only run one program at a time.

In the late 1970s, the Apple II was the first widely popular *desktop computer*, from which the notion of a *personal computer* arose. The Apple II was revolutionary because it featured many important advances such as the removable floppy disk drive, a mouse to move the cursor on the screen, and the use of a point-and-click "windows" user interface (as opposed to merely typing at a command line). The Apple II is basically the predecessor of modern Mac computers used widely today. In the early 1980s, the IBM PC (Personal Computer) burst onto the scene featuring MS DOS (MicroSoft Disk Operating System). Microsoft-based PCs would soon dominate the PC market. Microsoft even patented terms such as "Windows" even though they weren't the first to use them in computer operating systems. By the 1990's, around 95% of PCs were running Microsoft Windows, even though Apple/Mac PCs were more advanced in some ways. The story of how Bill Gates' Microsoft Company came to dominate the desktop PC market is quite interesting, but is beyond the scope of this discussion.

The rest of this story is recent history. Computers have become so small that we carry them in our pockets (phones), wear them on our wrists (watches, fitbits), and even have them embedded in our cars, TVs, and kitchen appliances. It's to the point that it's hard to live without them. For desktop computing, Microsoft dominance has gradually receded (although still leading worldwide) with increasing usage of Mac OS and also Linux (which is free). For mobile computing, the iOS (Apple) and Android (Google) operating systems are by far the most widely used. Tablet-style computers live in the blurry realm between mobile and desktop devices and use a variety of operating systems including iOS, Windows, and Google's Linux based Chrome OS (not to be confused with Google's Web browser also called Chrome).

2. BRIEF HISTORY OF THE INTERNET

In the 1960s, the *Advanced Research Projects Agency (ARPA)*, funded by the USA Department of Defense, was developing technology to allow computers to communicate over very long distances because of the ongoing cold war and the threat of nuclear attack. Indeed, it would certainly be desirable for key government agencies and military installations to be able to maintain computer-based communications should a nuclear war break out.

Going online in 1969, **ARPANET** – ARPA's network – was essentially the first version of the *Internet*. As shown on the left of Figure 1, ARPANET originally linked only four remote locations at research universities in the western United States: UCLA, UC Santa Barbara, the University of Utah, and the Stanford Research Institute. The technology used was the refrigerator-sized minicomputer of the 1960s. Pictured on the right in Figure 1 is a Honeywell DDP-516 minicomputer (12K of memory), which was used for the "Interface Message Processors." They were essentially the first Internet *routers,* a term we will soon define in more detail.

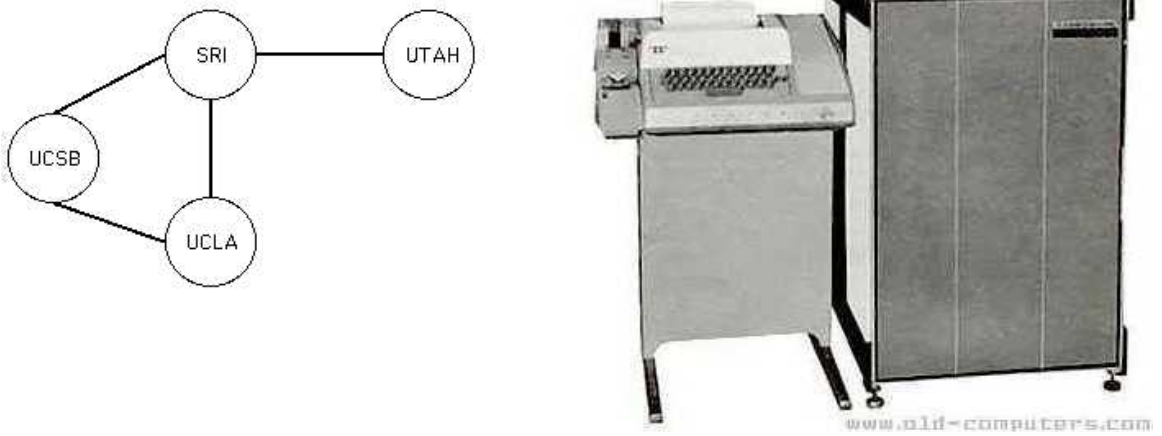

Figure 1 Left: ARPANET host locations in 1969 Right: Original ARPANET routers

Original plans allowed for ARPANET to be capable of expanding to 128 host computers. That must have seemed like a big number at the time, perhaps even excessive, but they were limited by the technology of the time, and clearly couldn't foresee the sequence of events that would cause the Internet to explode to hundreds millions of hosts within about 30 years.

In the early 70's, other large research universities and government research centers like NASA connected to ARPANET. Distant connections through satellite links soon followed. The first international connection to ARPANET came in 1973 when Norway's powerful seismological research computer facility was connected to ARPANET through a satellite station in Sweden. The research of interest to the US government at the Norwegian seismological facility was focused on detecting underground nuclear detonations.

Figure 2 shows the growth of the Internet through roughly its first 35 years. Prior to the mid 1980s, mostly only high-level researchers and scientists were using the Internet, and it grew fairly slowly. In 1986, through funding from the *National Science Foundation (NSF)*, new Internet infrastructure called NSFNET went online with the goal of connecting a broader range of academic research institutions. Before long, virtually every college and university was hooked up to what was then being termed the Internet. Notice in Figure 2 that the number of Internet hosts grew by a factor of 100 during the five-year period from 1984 to 1989 when NSFNET went online.

Notice that it again grew by another factor of 100, this time in the shorter period between 1989 and 1992. After that, you can see that it started doubling every year. That's literally exponential growth. The discussion below identifies the 3 biggest factors that contributed to the exponential growth that began around 1990. They are not the only factors, but it's safe to say that if any of these 3 things had not happened, Internet use would not have achieved exponential growth.

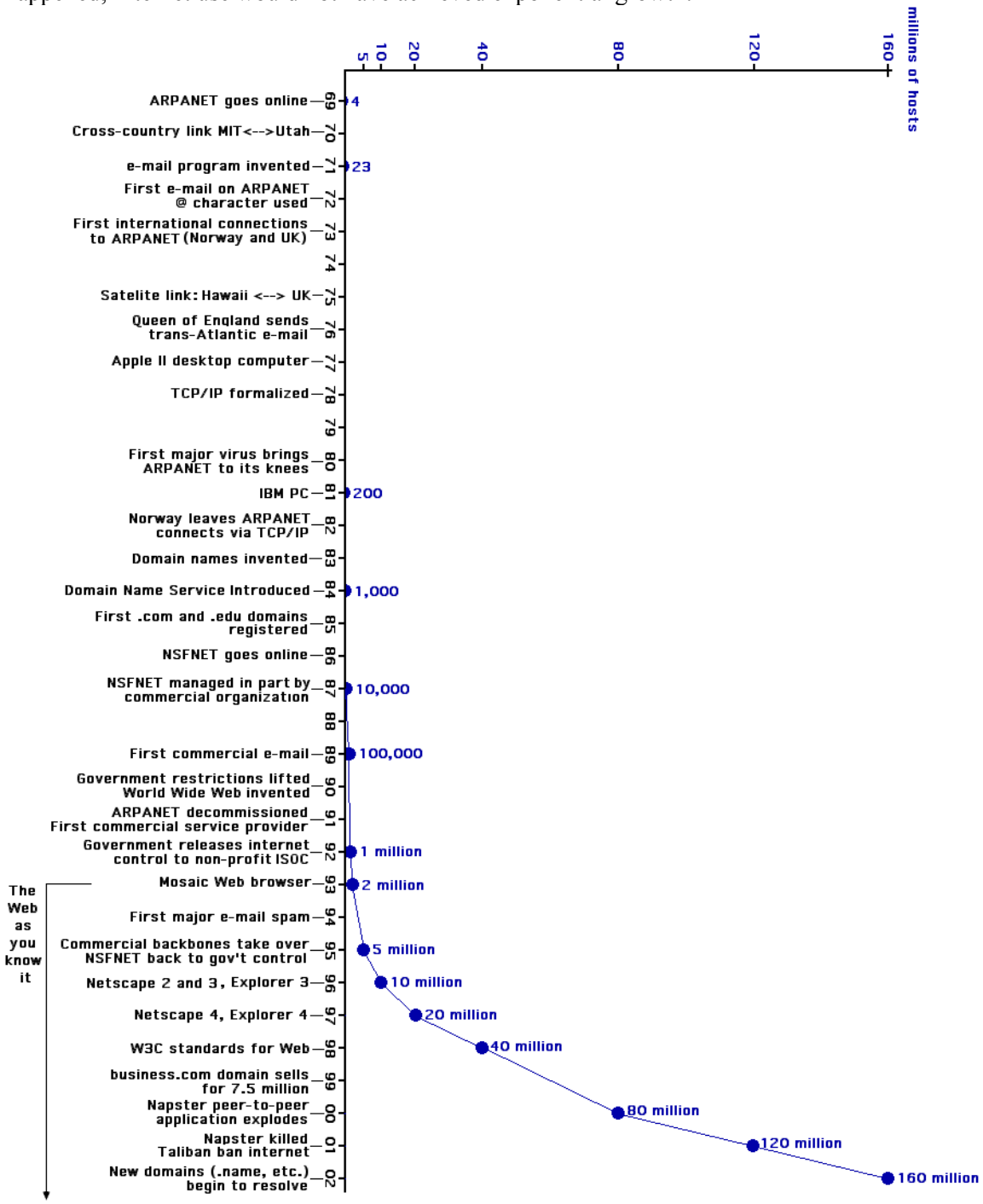

Figure 2 Internet Growth Timeline – roughly its first 3 decades

First was the proliferation of affordable personal computers. It's quite obvious that the Internet would not have been able to explode if computers weren't available for the masses. By the early 1990s,

PCs had already been on the scene for over 10 years. They were becoming smaller, more powerful, and more affordable. People increasingly had them in their homes, and most schools and colleges made them available to students. Moreover, the NSFNET project had extended Internet connections to educational institutions, so they were now online. It was no longer primarily only researchers and scientists that had access to the Internet.

The second major factor was the US government releasing control of the Internet. Prior to this, you actually had to formally apply to a government agency to get Internet access. It was becoming too big and complicated for the US government to maintain. In 1992, loose oversight of the Internet was transferred to a non-profit that eventually became the *Internet Society (ISOC)*, an international non-profit organization that sets technical policy to benefit the health and vitality of the Internet. Creating Internet infrastructure was then open to corporations and driven by demand in the market place, independent from government bureaucracy. Large telecommunications companies quickly invested large sums in Internet infrastructure around the world. People started small companies hoping to cash in, and many did. Private Internet Service Providers (ISPs) began selling Internet access subscriptions to the general public. The Internet was now out of the schools and colleges, and available in people's homes.

The third factor in the Internet's explosion was the World Wide Web. The WWW didn't arrive until the 1990's, some 20+ years after the Internet. We aren't yet ready to explain the difference, but for now suffice it to say the advent of Web pages and Web browsers changed the way we consume information. Prior to the WWW and information-at-your-fingertips, most people didn't have a compelling reason to buy Internet access for their homes or businesses. The WWW motivated children and grandparents alike to get online, and the rest is recent history.

The Internet growth chart in Figure 2 conveniently stops counting "Internet hosts" around the beginning of the cell phone era. Traditionally a host was simply a computer with Internet access. Now we speak of "devices" and that includes things like phones, cars, and household appliances just to name a few. A whale or dolphin that's been planted with a GPS sensor and is cruising around the ocean in the wild even qualifies. This vast collection of inter-connected devices is increasingly being called the *Internet of Things* (IoT). In the first decades of the Internet, it was instructive to track its growth as traditional computers connected. These days, it's perhaps more interesting to track the implications on society rather than focusing on the numbers of devices connected. The Internet of Things is starting to permeate the way we live.

Now that Internet infrastructure is in the commercial domain, you may be wondering about the legacy of the governmental agencies that were instrumental in creating it. ARPA was completely out of the Internet business by 1991, and is now called DARPA (the D is for Defense). If you search for "darpa projects" or see the news section of their Web site at *darpa.mil*, you will see that they are still actively researching cool technologies that have defense implications. The NSF was also completely out of the Internet business by 1998, a few years after NSFNET was taken offline. Again, the news section of their site at *nsf.gov* shows a wide variety of research areas they currently fund. The government was instrumental in creating the Internet, but it never would have reached it's potential if the government had continued to control and regulate it.

3. THE INTERNET

The Internet can be broken down into five layers that perform separate functions, but also work together to make it reliable and efficient. Figure 3 depicts the *5-layer Internet Model*. This section discusses these layers, starting from the bottom.

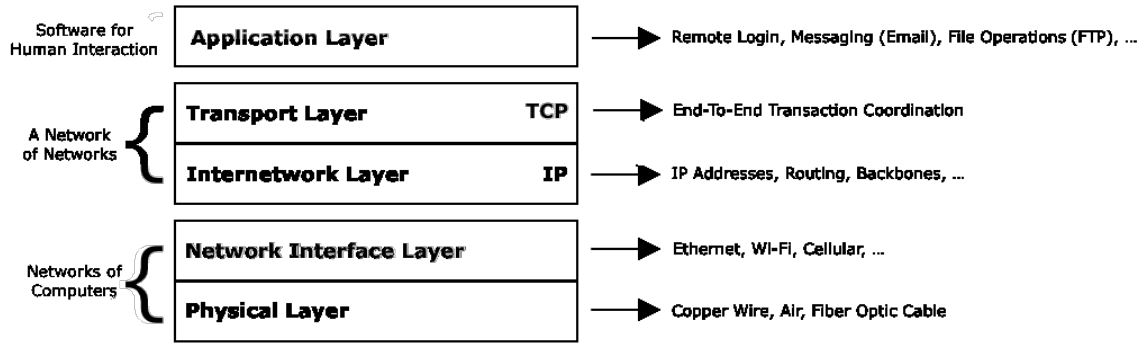

Figure 3 The 5-layer Internet Model

The Physical Layer

In the early years of the Internet, most data was transmitted through copper telephone cables, where digital computer data is represented as ordinary electricity. Electrical current through copper wire is described in terms of voltage, electrical resistance, and current.

Before long, data was also being transmitted through the air as Electro-Magnetic Radiation (EM). EM is called different things depending upon its wavelength. The EM spectrum goes from low energy (big wavelength) radio waves, to infra-red, to the light your eyes see in the visible spectrum, to ultra-violet, to high energy (small wavelength) X-rays that can reveal your bones through flesh. EM is described in terms of wavelength, amplitude, and frequency.

Computer data sent through the air is generally in the low-energy, radio wave spectrum. It might be surprising to you that computer data is also transmitted around the visible spectrum, but not through the air. (Indeed, it would be very weird too actually see computer data in the air.) Rather, EM in or near the visible spectrum is sent through fiber optic cables, which are bundles of many strands of glass, each about the thickness of a human hair.

There's no point in discussing the Physical layer in any greater detail, because these things are of more interest to physicists and electrical engineers. But it is pretty cool to realize that a chunk of digital data produced by a computer might go around the world in a fraction of a second, traveling through each of copper, air, and glass fiber along the way.

The Network Interface Layer

Simply put, this layer deals with networks – computers reliably transmitting data to each other. When two or more computers can communicate reliably, they form a *network*. We will use the term *network layer* or simply *networking* in this discussion, rather than the more formal *network interface layer*.

Most any form of communication requires specific rules to make it uniform and reliable. That is certainly true for written communications, which have specific rules for structure and grammar. A communication rule for a computer network is usually called a *protocol*. A pre-computing definition for a protocol is "points on which agreement has been reached by negotiating parties." For computer networks that translates to "a formal description of message formats and the rules two or more computers must follow to exchange those messages."

Details of specific networking protocols are well beyond the scope of this discussion. Instead, we'll focus on the names, key terms, and how the various types of networks are used. You have no doubt

heard of some of the most common networking technologies described below, but some may be new to you.

Ethernet is the most common networking protocol for hard-wired *Local Area Networks* (LANs). An Ethernet LAN is typically the size of all the computers in a room or a relatively small building. A hard-wired network at a business or lab at an educational institution is almost always Ethernet. An Ethernet cable very closely resembles an ordinary copper telephone cable, but is slightly thicker and plugs directly into a computer.

Ethernet is historically the most common network type, so it is often used as a benchmark to compare against other network types. Network *bandwidth* (data transfer rate) is usually measured in Mbps (megabits per second). Currently, standard Ethernet networks transfer data at 1000 Mbps (Gigabit Ethernet) while older 100 Mbps Ethernet networks are being phased out. For perspective, Gigabit Ethernet transfers 1 billion bits (0s and 1s) every second.

Wi-Fi refers to several networking protocols used to create wireless (through the air) LANs. The two most common types are identified by their wave frequencies: 2.4 GHz and 5 Ghz. The 2.4 GHz band has a range of about 90 meters through the air, and can pass through solid objects like walls, although that decreases the range. The 5 Ghz band can transmit data more quickly (faster network), but can't pass through walls and has a shorter range of about 30 meters.

Virtually any store that has electronics sells relatively inexpensive Wi-Fi transmitters for your home or office. The area of coverage is often called a *hotspot*. If you wander into a hotspot with your device, the Wi-Fi network may or may not require a password to join. It is becoming increasingly common to cluster hotspots with overlapping coverage areas to create saturation networks that cover whole city blocks, for example.

> The term Ethernet arises from an old scientific theory that suggested inter-planetary space is filled with the inert gas *ether*. That theory was dismissed long ago, but the adjective *ethereal* is still used in everyday language to mean heavenly.
>
> The term Wi-Fi also has interesting origins. Hi-Fi was used decades ago to refer to new stereo technology that produced sound with Higher Fidelity to how live music would sound. Wi-Fi is a clever adaptation of that old term applied to advances in wireless networking.

Cell Phone Networks are obviously wireless (though the air), but use entirely different protocols than Wi-Fi. The transmitters, often called *cell towers*, are usually placed on top of a tall pole or building, and can transmit signals for a few miles. The different cell networking technologies are grouped into different generations: 3G, 4G, etc. These designations don't refer to specific technologies or protocols, but rather minimum standards (data transfer rates, reliability, etc) that a given technology must meet. For example, two different cell providers like AT&T and Verison might use significantly different technologies that both meet the minimum standards to be called 4G.

Obviously, 4G cell networks were created to replace older 3G networks. The newest 4G networks are called 4G LTE, or simply LTE (Long Term Evolution). LTE networks are basically the fastest (10x faster than 3G) in the evolution of the 4th generation cell networks. At the time this was written, 5G cell networks are in trial phases and likely won't see commercial deployment until the early 2020s. 5G networks purport to be as much as 20x faster than LTE networks, but they also pose logistical problems. For example, the EM waves needed for 5G cover shorter distances and thus require a higher density of cell towers (on every lamp post?), and are higher energy and thus could pose public health risks.

ISP Networks include several very different networking technologies that have one thing in common. They are used to connect your computer or device to an Internet Service Provider. The most basic ISP

connection uses a *phone modem* and the Point-To-Point networking protocol. This literally means a two-computer network link between two points: your computer and one at the ISP. Phone modems transmit over a traditional hardwired telephone line, and phone calls can't be made while the modem is in use. Thus, people would often install a 2nd telephone line (in addition to the one used for telephone) called a Digital Subscriber Line (DSL) which offers a faster connection to the ISP and doesn't interfere with the telephone service. These types of point-to-point connections to ISPs have became increasingly uncommon in the USA.

Now for most in-home use, people generally pay for Internet service as a bundle with a Cable TV or Satellite Dish subscription. The device on your end is usually still called a modem. For example, the purpose of a cable modem is to make a network connection (over a coaxial cable) to the cable TV provider, which also functions as the ISP. Sizeable companies and educational institutions often require much more data capacity, so ISPs provide a wide range of commercial options. For example, most colleges require so much data throughput that they purchase a subscription that features a fiber optic bundle network connection straight to the ISP.

Other than having to buy your own equipment and devices, setting up a network is free. For example, you don't have to pay a subscription service to set up an Ethernet or Wi-Fi network so that several computers in your house can communicate. But if you want to access to the Internet, then you have to pay an ISP for a subscription. Cell providers also function as ISPs for mobile devices but, unlike Cable/Dish TV providers, may charge based on the amount of data you transmit. You may walk into a Wi-Fi hotspot and get a "free" Internet, but somebody (your college or coffee shop) is ultimately paying an ISP for the service.

It wasn't long ago that computers had limited networking capabilities. For example, if you bought a desktop or laptop computer in 2005, it might only ship with Ethernet capabilities. If you wanted Wi-Fi, you could buy and install a Wi-Fi *network card*. It was called a *card* because it was a small electronic circuit board about the size of an ordinary playing card. Over time, the circuitry required for networking got smaller, but it is still a limiting factor. For example, the minimum thickness of a cell phone is in part determined by all the networking hardware that it has to contain.

Now, nearly all devices (desktops, laptops, tablets, phones, etc.) come with several built-in network capabilities. Desktops/Laptops can switch between Ethernet and Wi-Fi. You can configure which type it should always use, or to automatically look for a Wi-Fi network if there is no Ethernet cable plugged in.

Phones also have several different networking capabilities built in. Most people set their phones to automatically choose Wi-Fi if available, but to resort to Cell networking otherwise. That's good because Wi-Fi doesn't incur data use charges like some Cell carriers. Phones will even dynamically switch among types of Cell networking. For example, the author was recently in Yellowstone National Park where cell coverage is spotty (at best) and observed a phone automatically switching among 3G, 4G, and 4G LTE depending upon what type of network signal it could detect in the air at any given time.

Newer Wi-Fi transmitters can run wireless networks in both the 2.4 Ghz and 5 Ghz bands simultaneously. You can configure the two distinct Wi-Fi networks to have different names so that you can choose either one. Or you can let your device decide! If you give both networks the same name, you just choose that single name and most newer devices are smart enough to automatically switch between the two different Wi-Fi networks (the slower one that works through walls, or the faster one that doesn't) depending upon which one is providing the strongest signal at any given time.

Having discussed different network types, you might be curious to see how they compare in terms of speed. Figure 4 contains comparisons of how long it might take to transfer an MP3 music file (let's say Metallica) that's 5 Megabytes, which is actually a bit larger than an average MP3 file. The estimates are very approximate, since the different network speeds are approximate. For example, different 4G Cell carriers might have different bandwidth rates. Note also that these are download

speeds, which are often faster than upload speeds over ISP network connections Finally, Wi-Fi comes in many flavors as you can see from the actual protocol numbers like 802.11ac in Figure 4, which lists several recent Wi-Fi versions that have been widely deployed. When you walk into a Wi-Fi hotspot, you usually don't know what flavor you might get!

Network Link	Approximate Transfer Time	Typical Bandwidth
Gigabit Ethernet, Wi-Fi (802.11ac), 4G LTE Cell	1/25th of a second	1000 Mbps (1 Gbps)
Wi-Fi (802.11n), 4G Cell, Cable Modem	.4 seconds	100 Mbps
Wi-Fi (802.11g), Satellite Dish Modem	1.25 seconds	50 Mbps
3G Cell	8 seconds	5 Mbps
Dialup Phone Modem	13 minutes	.05 Mbps (50 Kbps)

Figure 4 Approximate transfer times for a 5 megabyte MP3 file

A final point to make about networking in general is that even a single transaction, made from a typical device, likely traverses several different network types. For a convenient case study, consider a typical in-home setup where Internet access is purchased through a cable company. The cable company will install a cable modem somewhere near where the coaxial cable enters the house. You might then run an Ethernet cable from the cable modem to a Wi-Fi transmitter placed in a central location that would provide good hotspot coverage for the whole house.

Now suppose your cell phone sends data to the Wi-Fi transmitter (perhaps, because it has automatically switched from the Cell network to Wi-Fi to save data). After the Wi-Fi network transaction, and then the Ethernet network transaction, the data goes from the cable modem to the ISP (cable company) over yet a different networking protocol (however they transmit over coaxial cable). But that's not even the end of it because the ISP then sends the data to an *Internet Backbone* that can send it around the world. Internet backbones are a key topic in the next layer up in the Internet model.

The postal system for physical mail and packages (snail mail) makes a good analogy to demonstrate the division of labor among the Internet layers. The physical layer for a postal service contains the trucks and airplanes that carry the packages. Packages can get lost or damaged in transit. Likewise, the transfer of computer data is not reliable in the physical layer. Electricity dissipates in copper wire and something as common as weather can interfere with data in the air. There needs to be some checking at each endpoint to detect data loss, hence the Network Interface Layer.

All the networking protocols mentioned above have built in error recovery so that network communications are reliable. It makes sense to think of this as the physical mail carrier scanning and weighing the packages before putting them on the truck or plane, and then again when unloading. You basically check the goods on each end.

The Internetwork Layer

It's best to begin discussion of this layer with a key question. You have seen how many very different network types there are. How can we establish communications among all the different types of networks? For example, if you are on a cell phone transmitting over a Cellular network, how can you communicate with a hardwired computer on an Ethernet network?

The short answer is in the boldfaced portion of the name of the **Internet**work layer. That's where the term **Internet** comes from. The Internet is a network of networks that enables computers to communicate, regardless of what type of very different networks they are on. The long answer is to describe how the *Internet Protocols (TCP/IP)* work. The *Internetworking Protocol (IP*) is discussed below, leaving TCP for the discussion of the next layer.

First, every computer needs an address that's independent of the underlying networks. Addressing at the network level is done with hardware addresses. Each piece of networking hardware comes from the factory with a unique address code. But the Internet uses *IP addresses* that are independent of the underlying network connection. Each IP address is of the form 164.68.21.170, where each of the four

numbers can be in the range from 0-255. To communicate on the Internet, a computer must have an IP address. By the multiplication principle, there are 256^4 (approximately 4 billion) possible IP addresses. The *Internet Assigned Numbers Authority* is responsible for giving out IP addresses, which are free based upon need. IP addresses are usually not given out individually, but are given in groups to ISPs, businesses, colleges and the like.

A *class A* license grants control of a block of addresses of the form 164.x.x.x. With 256 possibilities for each of the x's, this license grants control of 256^3 (approximately 16 million) different IP addresses. A class A license might be given to a very large ISP such as Comcast or AT&T. Class B licenses are of the form 164.68.x.x. This yields around 65 thousand different IP addresses by filling in the two x's. The smallest license is *class C,* which is of the form 164.64.21.x, and provides only 256 distinct IP addresses. A Class C block of IP addresses might be granted to a small business or organization.

The next notion is how data is moved from one IP address to another. It is not efficient to move data in big chunks. Data is divided into small chunks called *IP packets*, or simply *packets*. Packets are typically about .5K to 1.5K in size. To gain some perspective, recall the 5Meg MP3 file we used to illustrate different network transfer speeds. If that file is sent over the Internet, it would be "chopped up" into *several thousand* separate data packets during its journey.

The notion of using small packets to send data is a major contributor to the efficiency of the Internet. Think of how a large shipment of books would be handled in the real world. They would be packed into boxes of manageable sizes, each stamped with addresses. If part of the shipment is damaged or lost, only that part of the shipment would need to be resent. Similarly, packets are marked with the origin and destination IP addresses. Any packets associated with a given data transfer that are lost or damaged can easily be re-transmitted. This is far more efficient for the Internet's infrastructure than re-sending entire chunks of data just because one small part of it is corrupted.

So how are all these trillions of packets moved around the planet? Computers that move packets around the Internet are called *IP routers,* or simply *routers*. They literally route packets to where they need to go. A router is typically a dedicated computer whose sole purpose of existence is to receive a packet, look at the destination IP address, and send it to the next router along the way.

Groups of routers are organized into *Internet Backbones*. Basically, ARPANET and NSFNET were the first two backbones. After the US government released the Internet into the marketplace, large telecommunications corporations built vast commercial backbones that now cover most of the habitable portions of the planet. They are basically huge IP routing networks that can move vast amounts of data through fiber optic bundles (or otherwise) all around the world. It's hard to comprehend the size and speed of a major commercial backbone without actually seeing one, so a real backbone segment is shown on the course Web site.

Having introduced the relevant terminology, it's helpful to depict the Internet as a vast network of networks "glued" together by Internet backbones as shown in Figure 5. To access the Internet, you (or your employer or your college) have to buy Internet access from an ISP. If your ISP doesn't have their own backbone infrastructure, they are most likely paying to pass your packets to a larger backbone ISP. If your packets are heading to an IP address on the other side of the world, they may be passed through exchange points to other backbone providers along the way. And the beauty of it all is that the backbone routing infrastructure simply doesn't care what type of networks from which the packets originated. It's literally an **inter-net**work!

IP (Internetworking Protocol, if you recall) has some interesting features that have contributed to the Internet's efficiency. The protocol has been improved over time, but these efficiencies were mostly implemented as far back as ARPANET. First, is the notion of *flow control.* Flow control basically means routers can communicate with each other to try to optimize the flow of packets. For example, if one router in a backbone is overloaded or malfunctioning, IP is smart enough to avoid that router and send packets to a different router. So in IP routing, the optimal path is not always the shortest one in terms of distance, but may be the best one in terms of time and efficiency.

One ramification of flow control is that two packets from the same transaction might not travel the same route across a backbone, or might even be passed through different backbones. For example, consider again the 5Meg MP3 file that would result in thousands of small packets. If that file were sent from New York on the East coast to Los Angeles on the West coast, it's quite possible that some of the packets would route through Chicago, but others from the same transaction wouldn't.

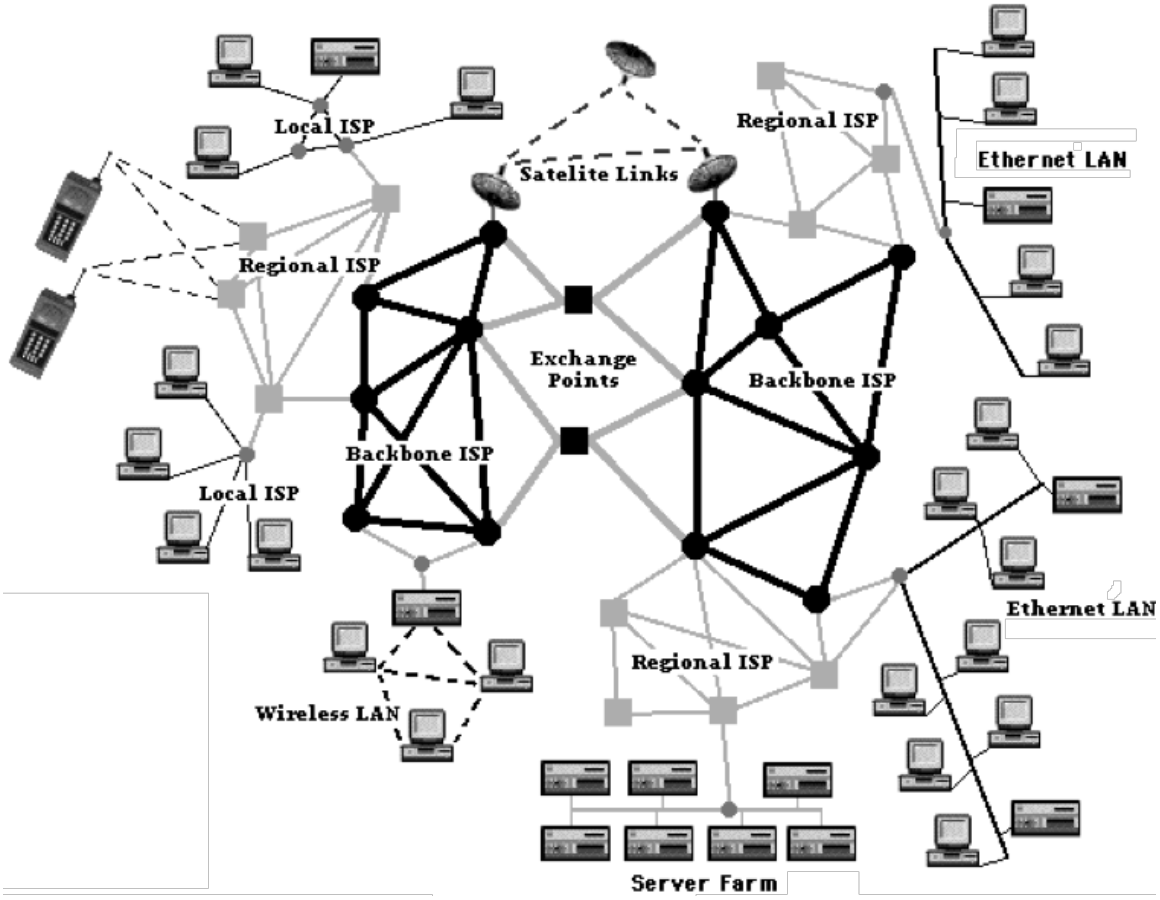

Figure 5 The Internet is a Network of Networks

Time To Live (TTL) is another interesting feature of IP routing. The TTL is a number (positive integer) assigned to each packet that specifies the maximum number of router jumps that the packet is allowed to make. Every time a router handles a packet, 1 is subtracted. If the TTL hits 0 at a given router, the packet is simply deleted. No notification is given to the source or destination computers. IP simply discards the data packet and continues routing other packets.

At first that might seem surprising, but it makes perfect sense. A packet can't be allowed to cruise around the backbone aimlessly or end up bouncing back and forth between two routers (think ping pong) because of some sort of addressing confusion. Rather, if a packet doesn't make it in a timely fashion, IP simply kills it. Generally the maximum TTL that's used is 255, but smaller numbers can be used to increase efficiency.

Aside from TTL, something as simple as a malfunctioning router can cause packet loss. IP routing is simply not reliable and doesn't guarantee delivery of packets. Modern commercial backbones typically deliver packets with at least a 99.99% rate. But even a .01% loss rate (as a fraction that's .0001) is a big number when moving trillions of packets.

To wrap up discussion about IP, it's instructive to return to the typical in-home setup we used as a case study when discussing networks: ISP⇔Cable Modem⇔Ethernet Cable⇔Wi-Fi hotspot. The ISP (Cable Company) assigns the cable modem an IP address. That's the primary, public Internet address for the house. However, there may be many devices in the house that connect to the Wi-Fi hotspot to access the Internet. To the outside world, all those devices appear to be using the same public IP address of the cable modem.

Earlier we called them Wi-Fi transmitters, which is accurate. But they are usually called *Wi-Fi routers* because they also have IP functionality. Since lots of different devices might use the same hotspot, the Wi-Fi router implements a *private network* and uses a *private IP address space*. For example, the class B IP addresses space 192.168.x.x is reserved, and never used for public IP addresses. As devices connect to the Wi-Fi router, it assigns them private addresses in this space: 192.168.0.1, 192.168.0.2, and so forth. The interesting thing is that the house next door likely has its own Wi-Fi router, which also uses the same private address space. That means a device in each house can both have the same private IP address, 192.168.0.1 for example, which seems problematic.

Network Address Translation (NAT) is an IP routing extension that solves this problem. Basically, the private "fake" IP addresses are used for packet routing internally inside the hotspots, but are translated into the actual public IP address when packets leave the house. Routing in the outside world (backbones) only uses the real public IP address assigned by the ISP. Recall from earlier that there are only about 4 billion distinct IP addresses. Use of NAT and private address spaces allows many, many more than 4 billion devices to connect to the Internet. The ISP only needs to assign you one IP address, and equipment you can buy like Wi-Fi and Ethernet routers create "virtual" private address spaces to accommodate all your devices.

> The analogy with physical mail extends naturally to this layer. An IP router is similar in functionality to a post office. A post office gets a letter, reads the address, and then moves the letter on its way, which might be another post office or the final address. The underlying delivery networks (planes, trucks) have protocols to guarantee delivery (endpoint scans) between post offices, but the post offices themselves don't guarantee delivery. Like a router moving packets, a post office just tries to move letters toward their destination.

The Transport Layer

IP routers do not care whether a packet makes it to the destination, or even which transaction a packet is part of. A router might move two packets toward the same IP address, but one packet is part of a streaming video and the other packet is part of an email message. The router has absolutely no idea, and doesn't care. Its job is just to route them, hopefully toward their destination.

The *Transmission Control Protocol (TCP)* adds reliability on top of IP. In fact, the *Internet Protocols* are usually listed together as TCP/IP. They form a very important division of labor since TCP implements the delivery guarantee that IP does not. TCP establishes an end-to-end "conversation" between two computers that ensures a transaction is seen through to completion. Whereas IP handles the routing between computers, TCP operates only on the endpoints, the two computers involved in the transaction.

For purposes of discussion, let's say one endpoint is a *server* (a computer that makes data available) and the other endpoint is a *client* (a computer that consumes the data). For example, the client might be downloading an email from a mail server, or a web page from a web server. In many situations, a computer like your phone is acting as both a client and a server as it both consumes and generates social media posts, for example. But for purposes of discussing a single data transaction, we'll just call the originating endpoint the server, and the other the client.

Let's suppose the client has requested an email message from the server. In this case, the data refers to the entirety of the email message. Here's what TCP does on the server:

- It divides the data into packets and adds a sequence number to each packet's header so they can be reassembled later in the proper order.
- It calculates a *checksum* for each packet and adds that to the packet's header. A checksum is a count of the data in a packet that can be used later to test for data loss.
- It adds the destination IP address (the client) to each packet's header.
- It gives the packets to the IP layer for delivery.

The packets eventually make their way to the ISP, then across the backbone(s), and then (hopefully) find their way to the client that requested the email message. As the IP layer on the client receives packets, it passes them to the TCP layer, which does the following:

- It re-calculates the checksum for each packet. If that doesn't match the original checksum, the packet is considered damaged, so the TCP layer on the client asks the TCP layer on the other end to re-send it.
- It can detect from the sequence numbers if one or more of the packets have not arrived after some period of time (often called a timeout period). The client's TCP layer will simply ask the server to resend any packets that never arrived.
- It basically repeats the two steps above until all the packets have arrived intact.
- The TCP layer re-assembles the original data according to the sequence numbers of the packets. The email message has thus been entirely transferred to the client.

You can see why this "conversation" has to be handled by the endpoints. No way can IP routers attempt to monitor the state of transactions between millions of computers scattered all over the planet. In the example transaction described above, only the client and server should have to keep track of the progress of the data transfer. IP has enough to worry about just routing the packets around the world.

Figure 6 depicts the end-to-end TCP service and introduces a few more concepts. It depicts sending data from a server to a client like in the above discussion. Virtually all modern devices (including phones, etc) have TCP/IP built in. IP doesn't do much on the endpoints other than send packets to/from the "nearest" IP router. The rest of IPs routing work is done in between the endpoints. But TCP does all of its work on the endpoints. There's no function of TCP in the actual routing.

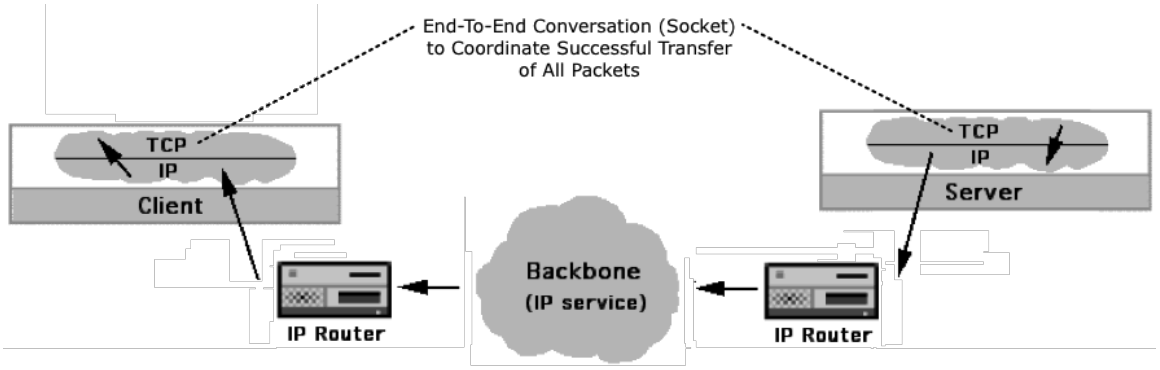

Figure 6 TCP provides the end-to-end service for Internet transactions

The end-to-end "conversation" is called a *socket*. That sounds like something physical or mechanical, but it's just a name for the duration of the conversation -- the sequence of brief messages (perhaps one packet each) sent back and forth between the endpoints to track the progress of the transaction. When all the data is transferred and the conversation is over, TCP closes the socket by sending what are basically "goodbye, mission accomplished" messages.

You are no doubt aware that some transactions are secure, like when a web browser shows you a padlock icon. All this means is that all the packets for the transaction are encrypted (scrambled) by TCP on one end, and then un-encrypted by TCP on the other end. That makes the packets worthless to any 3rd party that might try to read packets during transit. This is called a *secure socket*. When you first visit a web site that's secure, the browser will ask you something like "are you sure you want to trust this site?" If you click yes, then the endpoint TCP layers quickly exchange security certificates in a preliminary transaction. These security certificates are then used to encrypt/de-crypt the packets on each end. The security certificates are only known by the endpoints, so no one in between can de-crypt the packets. Without transaction security, it wouldn't be safe to enter credit cards in web pages, and e-commerce would have never taken off.

Historically, the Internet protocols are usually listed as TPC/IP because is was desirable to ensure complete data delivery for most transactions. However, that's not necessarily true when "streaming" music and video, which has increased dramatically in recent years. Music/Video streaming involves relatively huge amounts of data and happens gradually. In most cases, the end user will not even notice if a very small amount of the data does not arrive with the stream. Thus, for most streaming purposes, the minimum 99.99% delivery rate achieved by most IP routing services is perfectly adequate. Moreover, it's not desirable to pause steam buffering in order to request lost packets. Basically, it's usually not desirable to incur extra work during streaming by requiring TCP to continually monitor the transaction state at the endpoints.

There is another protocol called *UDP (User Datagram Protocol)* that works on the endpoints like TCP, but is a *connectionless protocol*. It still breaks data down into small packets and gives them to IP for delivery, but that's about all UDP does. It does not maintain a socket conversation to correct for packet loss. It just passes the packets to IP for delivery and then basically forgets about them. On the other side, UDP simply receives whatever packets arrive and doesn't concern itself with any missing ones. UDP/IP is an alternate Internet protocol for uses like streaming where TCP/IP incurs unnecessary work.

Back to the post office analogy one last time. TCP is like coordinating with your friend to see if the letter you dropped in the mail has actually arrived. If you drop a standard letter in the mail, you will never know if your friend got it unless you later have a conversation with them. Snail mail does offer certified delivery (at an additional cost) that will confirm delivery, but that is an endpoint service like TCP.

Unlike traditional snail mail, most package delivery services like UPS and FedEx supply tracking numbers that show the progress of a package along shipping nodes. But IP offers no such tracking potential. There is a protocol called *ping* that you can use to test an Internet transaction route. You can ping an IP address to see the sequence of router jumps (and the times involved) a single packet undergoes being sent between your location and that IP address. It's actually remarkable to see a ping trace of multiple router jumps that go halfway around the planet in a fraction of a second. The term has even made its way into everyday language. You are said to "ping somebody" to see if they respond.

The Application Layer

Computer programs, *software applications*, *software*, *applications*, and *apps* are all common terms that basically mean the same thing. They all refer to tools that computer programmers have written to facilitate humans interacting with computers. The computers themselves are the hardware (circuit boards, network components, hard drives, etc), whereas the programs you can install to make computers more useful are not part of the hardware, hence the term software. Of course, *apps* is the cool, trendy term used most often these days.

There's also the notion of a *killer app*, a software application that basically becomes indispensible and you can't imagine living without it. Killer apps generally cause a leap forward in the productivity humans derive from computers. Some of the original killer apps that come to mind are word processors and spreadsheets created in the 1980s.

But these first killer apps were created for personal/business productivity, not to make the Internet more useful for humans. That need arose in 1969 when ARPANET went online. The first "killer apps" written to increase the usefulness of the Internet were for remote login, remote file operations, and remote messaging. Now that computers could communicate across long distances, these types of remote capabilities were of immediate need.

Telnet software was created in 1969 to enable remote login, so that you could give commands and run programs on a distant computer. It is still in widespread use today, but most people use a version called SSH (Secure Shell) that uses secure sockets. FTP (File Transfer Protocol) was created in 1971. Software implementing FTP is often called an *FTP client* and allows for all sorts of file operations (get, put, delete, etc.) to be performed on a remote computer. Mostly SFTP (Secure FTP) is used today, but the software is still widely called just FTP.

The first email was sent on ARPANET in 1972. When software is built for the Internet, there's always an underlying communication protocol – the rules for the back-and-forth messages between the remote computers. Email is the common term, but the Simple Mail Transfer Protocol (SMTP) makes it work. Even from the onset, email addresses used the @ character like they do today.

The remarkable thing is that this software was all created within the first 3 years of the Internet, but no other "noteworthy" software was built specifically for the Internet until around 20 years later. Here noteworthy means software that became highly useful to the general public, not obscure software built by scientists at the time. The Internet would not begin to reach it's full potential until one of the most killer apps of all time was invented around 1990 – the *World Wide Web*.

> The WWW lives in the very top layer of the Internet. It's literally just a software application. The Internet, its protocols, and its infrastructure existed for over 2 decades before the WWW was created. They are radically different things. Yet many people use the terms Internet and WWW interchangeably, as if they are the same thing. Even articles in major news publications routinely make that mistake. Basically, not very many people had even heard of the Internet before the arrival of the WWW motivated the masses to get online. Hence the two terms are often conflated.

Figure 7 depicts how software completes the five-layer Internet model. In all the previous discussions about dividing data into packets and so forth, the data is coming from Software apps. When building apps, software developers can directly program the app to "call for a TCP socket" to transmit data over the Internet. Basically the software passes the data to the TCP socket call, and then TCP/IP and the rest of the Internet's infrastructure do the hard work. It's a division of labor and the software doesn't have to worry at all about how the lower levels of the Internet do their thing.

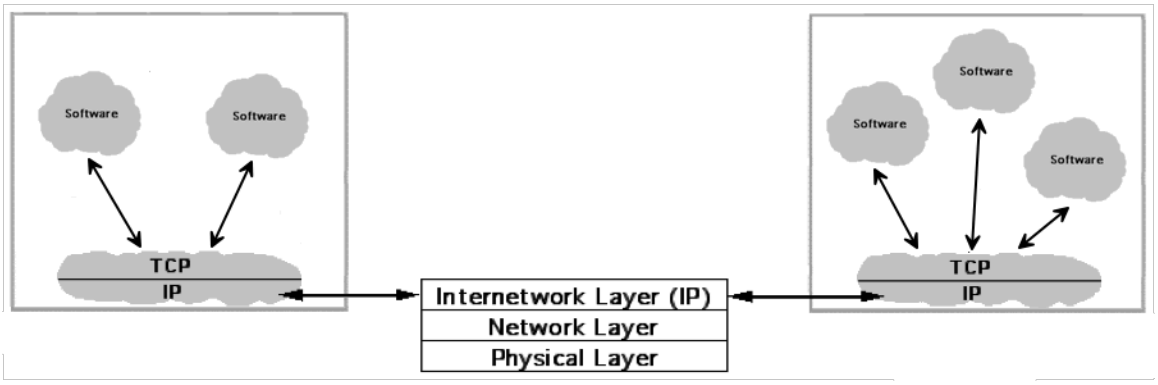

Figure 7 Software Applications call for TCP sockets to Start Internet Transactions

The TCP layer often needs to work with several different types of software at the same time. A given TCP layer might be dealing with email packets, packets from a web page, and packets from a

streaming video in a YouTube app, all at the same time. Obviously, TCP needs some way to keep the data from each app separate. Basically, TCP uses numbers called *ports* to keep track of all the apps it supports. It simply assigns a different port number to each app. For example, a web page is usually sent to TCP on port 80. So if a web browser is programmed to "listen on port 80," then TCP will pass all data that arrives on that port to the browser, thereby keeping that data isolated from other apps.

4. BRIEF HISTORY OF THE WORLD WIDE WEB

Originally from London England, Tim Berners-Lee got a job as a computer scientist in Geneva Switzerland at the *European Organization for Nuclear Research (CERN)*[1], which is still today one of the world's leading physics laboratories and sub-atomic particle smashers. Teams of physicists would come to CERN from all over the world to do research, and then go back home to their labs and universities. Tim Berners-Lee proposed to his bosses that he be allowed to build some computer software that would help the physicists share their research. At the time, there was no software available to publish large-scale document collections and make them readily viewable via Internet transactions.

He had a hard time convincing his bosses to devote money and resources to such a project. At first they said no, but Tim eventually sold them on the idea. He built the software over the period of a few months. He named the software the *World Wide Web*. The software consisted of two parts that worked together: a browser that could be used to view and edit documents, and server software that made documents available via Internet transactions. The documents were called *Web pages*. The name *World Wide Web* literally referred collectively to those two original pieces of software. Tim envisioned that this software could be used to create a web of inter-connected information around the whole world. To give his new software that name from the onset was quite visionary, to say the least.

The first web pages only contained text and links, no graphics or other fancy stuff. The formal term for a link was *hyperlink*, which would transfer to another web page. The simple language he created was called the *HyperText Markup Language (HTML)*, and the formal term for a web page was a *hypertext document*. Hypertext basically just means text that contains links to other text.

In 1991, the world's first Web server went online at CERN. But more importantly, they made this new software (both the browser and server software) available for free so that anyone else could also start a web server. The world's second web server went online about a year later at a physics lab in California. After that web servers slowly proliferated, and the first web site at CERN was used to list new web servers that came online. In a very loose sense, that was the first "search engine," although it was just a list of all web servers known to exist online at the time.

The WWW was growing, but was mostly being used by physicists and other scientific communities. The next major advance happened at the University of Illinois at the *National Center for Supercomputing Applications (NCSA)*, which had been created through funding from the NSF back around the time NSFNET had gone online. In 1993, a graduate student at NCSA named Marc Andreessen led a team of researchers that created a new web browser named *Mosaic*. The Mosaic browser introduced support for graphics in web pages, hence the name Mosaic. Like the original WWW software, they made the Mosaic browser available for free.

Mosaic was such a huge success that Andreesen formed a company called *Netscape Communications Corporation* and renamed the browser *Netscape Navigator (NN)*. Released about 1995, NN version 2 was one of the most influential browsers in history. It introduced JavaScript to make pages more interactive, cookies to enable the browser to store data between transactions (and track you), and fill-in forms so web pages could collect data from users. Netscape Communications even introduced the secure sockets extension to TCP so that packets could be encrypted during transit,

[1] The acronym comes from the organization's original French title, "Conseil Européen pour la Recherche Nucléaire."

keeping data entered into web pages secure. New startup companies like Amazon (originally just an online bookstore) took advantage of this to securely collect credit card payments in Web pages.

Given the quick success of Netscape, Andreesen took the company public on the stock market in 1995 through an IPO (Initial Public Offering). This was very unusual since they were giving the browser away for free, and the company was not yet profitable. In fact, they didn't even have a very good plan for increasing revenue other than through selling advertisements. But their stock nearly broke a first-day trading record, when it increased from $28 per share to $75. It finally ended the day at $58 per share.

> The Netscape IPO was basically the moment that the world became aware of the WWW. Newspapers and TV reporters everywhere trumpeted the remarkable story that a non-profitable company had raised nearly *3 billion* dollars in one day through an IPO. Unless you were hiding under a rock, you probably took notice. At the time, most people had not even heard of the Internet, and that's one of the main reasons people still equate these two very different things.

Netscape dominated the web around this time. If you were surfing the web, you were almost certainly using Netscape Navigator. Then Microsoft released a new browser named *Internet Explorer (IE)* in 1996. For all practical purposes, there were no IE versions 1 and 2. Since NN3 was already released, IE also needed to be version 3 so it would appear to be on equal footing with Netscape. But Microsoft didn't play nice, and the next few years would come to be known as the *browser wars*.

To make a long story short, Microsoft tried to kill the Netscape browser by integrating IE with their dominant Windows operating system. By 1997, IE4 was so tightly integrated with Windows that you effectively couldn't use NN4 on a PC running Windows, which dominated Macintosh at the time in terms of operating system market share. Microsoft was actually on the verge of killing the entire Macintosh brand by not releasing Microsoft Office software for Macs. Their Office software suite had become so indispensible that it wasn't practical to buy Macs if office wasn't available.

Because of all this anti-competitive behavior, Microsoft was hit with anti-trust lawsuits aimed at preventing it from becoming a monopoly. They eventually played nice, allowing NN to be easily installed on Windows and releasing a version of their Office software for Macs. Had Microsoft succeeded in effectively taking over the WWW by locking it into a proprietary system that they effectively owned, who knows what it would be like today. But it's safe to say that the WWW technologies would not have remained free and open, as they are today. That was a dangerous time for the WWW. It's also interesting that releasing Office for Macs effectively saved the Apple company, whose stock was trading at under 1$ per share during the browser wars. Apple would later introduce the iPhone, and their stock is trading at over $200 at the time this was written.

The competition between NN and IE during the browser wars was also having an adverse effect on the HTML language used to create web pages. Basically, each browser was introducing new HTML and JavaScript features trying to make web pages fancier. The HTML language was literally diverging as the competing browsers pushed the technologies in different directions. By the time the version 5 browsers came out in the late 1990's, web developers were increasingly creating different versions of the same page so they could take advantage of the most advanced features of each of NN and IE. Some web pages would actually say something like "best viewed in Netscape." Obviously, this was not a sustainable model.

The browser wars faded away largely because of the *World Wide Web Consortium (W3C),* a non-profit organization that had been formed a few years earlier by Tim Berners-Lee and some of the other people who had been instrumental in creating the web technologies. So they could get as broad of input as possible, they invited Microsoft and Netscape to the table as well as other companies that had a stake in the web. One of their first major tasks was to make a formal recommendation for the HTML language. Still to this day, the W3C maintains international standards for several important

web-related languages, including HTML. They also run an excellent web site named *W3Schools* that offers free tutorials and reference materials for the web.

The browser makers slowly conformed to the W3C standards and became more uniform. By the time the version 6 browsers were out in the early 2000s, both NN and IE were mostly compatible because of adhering to the W3C standards. But IE had pretty much won the browser war. The Netscape brand was sold to a larger company called America Online (AOL) and faded into obscurity. But the Netscape Navigator code base was given to a non-profit foundation called the *Mozilla Foundation*, which changed the name of the browser to *Firefox*. At the time, Firefox was considered to be the most advanced browser around and took quite a bit of market share from IE. There are now several popular browsers that all mostly compatible: IE (now called Edge), Safari, Chrome, Firefox, and Opera. Thanks in large part to the W3C standardizations, the web technologies have remained free and open to the benefit of us all.

> There was actually a competing technology called Gopher that appeared around the same time as the WWW. The software got its name from the mascot of the University of Minnesota where it was created. At first they gave it away like the WWW. But as it was gaining in popularity, they made the horrific mistake of trying to charge a license fee to install a Gopher server. Most people believe that the WWW was a superior technology and would have eventually won on its own anyway. But Gopher's mistake was an early indicator that the era of free, *open source* software was at hand.

The WWW started out serving up pre-constructed web pages interconnected via hyperlinks. But in short order, the technology advanced to allow for database-driven web applications. That means data is stored in a relational database from which web pages can be constructed on-the-fly. That means you are actually requesting a computer program (rather than a pre-built page) on the server that queries the database for information and builds a custom page based upon that. Think about a web page full of sortable sports statistics. Given how often sports stats change, you obviously can't pre-build web pages for each stats view. Rather, you build a database-driven web application that pulls sports stats from a database and lets you sort them however you want.

There are still lots (certainly billions) of pre-built web pages, but much of the data you consume on the Web comes from databases. This is sometimes called the *Deep Web* – the vast amounts or raw data, and the server-side computer programs that make web pages on-the-fly from it. That's not to be confused with the *Dark Web*, which is basically a second web with parallel protocols designed to hide your identity (think black market transactions). If you know much about the Dark Web, you are likely up to no good.

The "information at your fingertips" era enabled by the WWW was already amazing. Then Facebook, at the time a fairly simple database-driven web app, started the social media era. Now historians and social scientists are studying the impact that the wide variety of social media Web apps and mobile apps is transforming the way humans interact socially, for better or worse. Aside from the social media revolution, Modern browsers have gotten sophisticated to the point that much of the software delivered to you in web browsers resembles traditional desktop software. For example, word processors and spreadsheets are now delivered "from the cloud" to your browser as web pages that work like fully functional software. When working on a web-based spreadsheet, updating one cell actually incurs a client-server web transaction. Increasingly, cloud-based software delivered in sophisticated web pages is replacing the old model of installing stand-alone software packages on your PC.

By the end of 1993, around the time the Mosaic browser appeared, there were a few hundred known Web servers in existence and data generated by WWW software constituted around 1% of the packets traveling around the Internet. By the late 1990s during the browser wars, WWW-generated data finally surpassed data generated from email software, which had dominated Internet traffic until then. The vast majority of Internet packets are still generated by WWW software, but non-web

software apps like Snapchat are also sending huge amounts of data over the Internet. Current estimates are that almost half of all Internet packets are generated by *bots* (web robots). Bots are automated computer programs that create things like spam (around 50% of all email traffic) and bogus social-media postings. Shockingly, automated bots are competing with humans using software in terms of generating the most Internet packets!

5. WEB-RELATED TOPICS AND TERMINOLOGY

Most of the topics and terminology in this section also apply beyond the WWW. Remember, the WWW is just one type of software that uses the Internet's infrastructure to make remote transactions. For example, domain names like *lakeforest.edu* are used for remote login, FTP connections, and email transactions just to name a few Internet-capable software apps. But to most people, the concepts below are most relevant in the context of the WWW, so this is the perfect spot to introduce them.

Virtual Domains

For example purposes, consider a fictitious university named UWeb that owns *uweb.edu,* which is called a *domain name*. You are familiar with a wide variety of domain names like *google.com*, *lakeforest.edu*, and *w3schools.org*, just to name a few. But what exactly are they? The term *virtual domain* is an alternate term that more closely hints at their nature. We think of virtual as something that's not quite real, and that loosely applies here. A virtual domain is real in the sense that you can buy it and own it, but it's not real in the sense that it does anything useful by itself. If you buy one, you simply own and control it. It's literally like buying the intellectual property rights to own a name, *uweb.edu* in this case.

There are many places you can go to buy a virtual domain including Godaddy and Network Solutions. Go there for yourself and you can see how much buying a virtual domain costs, assuming of course that you can find one that's available. Most dictionary words have already been purchased. For example, type in *aardvark* and you will see that most domains based on that term (.com, .net, etc) are already owned by someone. If you find one that's still available, it's relatively cheap and easy to buy ownership of it for a period of time, usually 1-10 years.

Fully Qualified Domain Names

So how do you turn virtual property like *uweb.edu* into a viable Internet address? The short answer is that you give it a prefix like *www*, and then associate *www.uweb.edu* with an IP address. Remember all addressing on the Internet uses IP addresses. You can actually surf to Web sites using IP addresses. Simply type *http://74.125.28.147* into the address field of a browser and you will see that it accomplishes the same thing as surfing to *http://www.google.com.*

To make *uweb.edu* useful, we first choose a prefix, and *www* is the most natural choice. When we add the prefix and associate *www.uweb.edu* with an IP address, the result is a *Fully Qualified Domain Name (FQDN)*.

www.uweb.edu <---------- DNS Mapping ----------> 164.68.21.170

We will discuss DNS (Domain Name Service) in the next subsection, but the net effect of this is that when you type *www.uweb.edu* into a Web browser, it gets mapped onto the IP address. The browser then uses the IP address to find the server. The FQDN is strictly just for humans. But Internet transactions require IP addresses.

You can form multiple FQDNs from one domain name by adding different prefixes. For example, we could create two FQDN's*, www.uweb.edu* and *home.uweb.ed*u, and map them onto the same IP

address. We could use a third prefix and map *computerscience.uweb.edu* onto a completely different IP address. Basically, if you own a virtual domain, you can construct lmany different FQDNs from it by adding prefixes and mapping to IP addresses. That is, one virtual domain can be used to create multiple human-friendly addresses that point to one or more actual IP addresses where resources are located on the Internet.

> You have probably noticed that sometimes you can simply type in *google.com* and it will take you to the same location as *www.google.com*. From the early days of the web, *www* has been used as the natural, default prefix to use with a virtual domain. So for example if we buy the *uweb.edu* virtual domain, we would most likely set up *www.uweb.edu* as main FQDN that points to the main web site at UWeb. But other prefixes could be used to point to different web sites under with the *uweb.edu* domain.

Domain Name Service

Domain Name Service (DNS) is an application layer service that translates named addresses into IP addresses as depicted in Figure 8. The software that provides the service resides on a computer called a *domain name server* (also frequently referred to as DNS, but the S technically stands for Service). If you purchase residential Internet access from an ISP, they maintain your primary DNS server. If you access the Internet from an educational institution or large corporation, they likely maintain their own DNS server.

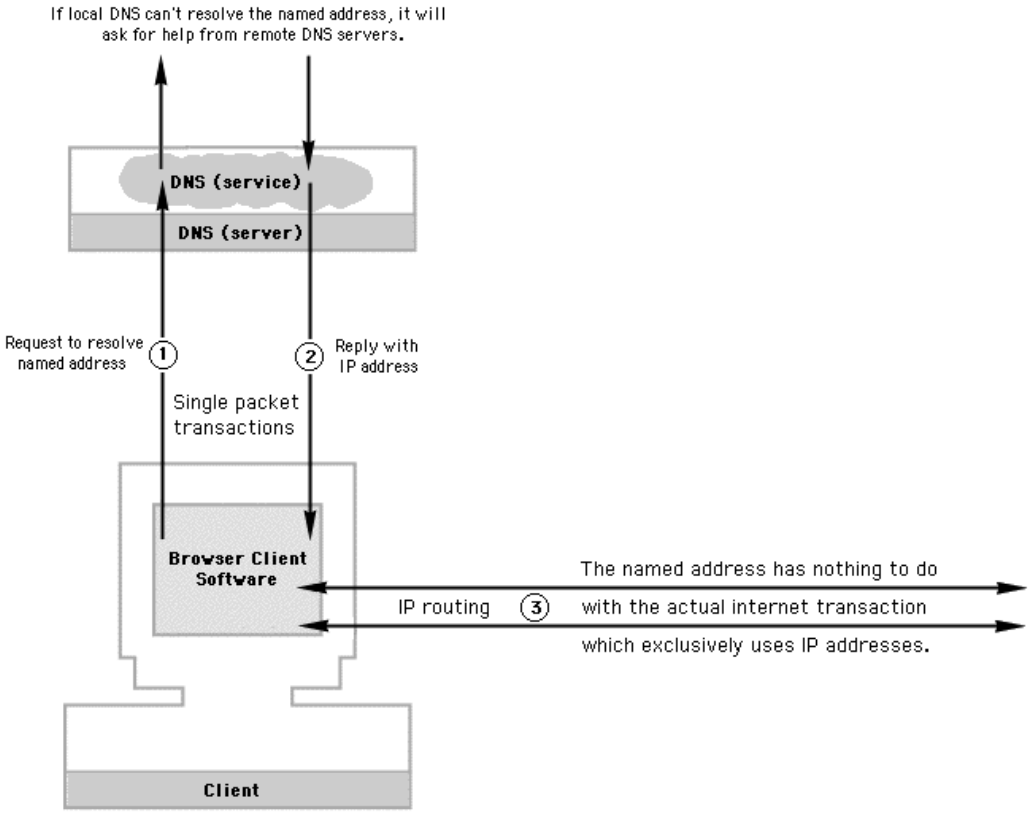

Figure 8 Domain Name Service Resolves an FQDN into an IP Address

When you type something like *http://www.uweb.edu* into your browser (or click a link pointing to the site) your Web browser first makes a request to DNS to *resolve* the named address to an IP address. This transaction is very simple in concept. "Can you find an IP address for this named

address?" "Yes, here it is." Resolution is very fast, usually only requiring a fraction of a second. Most people don't even realize this is constantly going on behind the scenes while you are surfing the web.

If you type in an address incorrectly or click a link with a non-resolvable domain name, your browser pops up a little alert window saying something like it is "unable to locate the server." This has no doubt happened to you. But it is not the Web browser itself that has failed. Rather the domain name service was unable to resolve the domain to an IP address.

Domain Name Resolution

Figure 9 illustrates the nature of a *top-level domain*. By far, the most common top-level domain worldwide is *.com* for *com*mercial organizations. Historically, common domains in the United States were for educational institutions (*.edu*), non-profit organizations (*.org*), organizations involved with Internet infrastructure (*.net*), governmental entities (*.gov*), and military institutions (*.mil*). These distinctions still hold today in many cases, but top-level domains like *.net* came to be used for all general purposes as did *.com*.

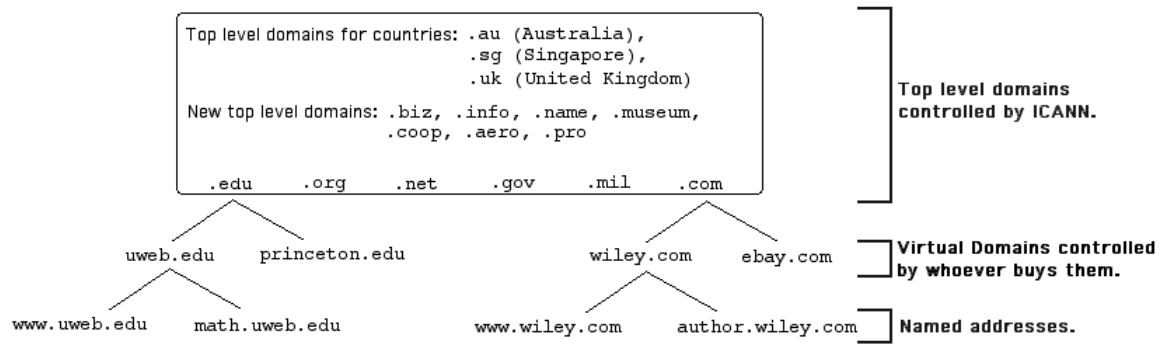

Figure 9 Hierarchical Structure of the Domain Name System

All countries in the world use a two-character top-level domain such as those shown in Figure 9. Complete lists of top-level country domains easily can be found through a quick search. The non-profit organization *Internet Corporation for Assigned Names and Numbers (ICANN)* controls the top-level domains. Over the years they have made lots of new top level domains available like *.biz*, *.info*, and *.name* just to list a few.

Notice the hierarchical structure shown in Figure 9. It's organized like that because one DNS server can't possibly know all of the world's DNS mappings. For example, UWeb's DNS can't possibly know all the IP addresses of all the obscure domain names to which students on campus might surf. When UWeb's DNS can't resolve a named address on its own, it asks other DNS servers for help. Suppose the request is for *princeton.edu.* UWeb's DNS first goes one step up in the hierarchy, asking the top-level *.edu* DNS server for help. The top level DNS then forwards the request to the *princeton.edu* DNS, whose job is to know all of the IP mappings for prefixes under the *princeton.edu* virtual domain.

If the request had instead been for *wiley.com*, it first would forward up to the top-level *.com* DNS server, which should know what DNS server back down the hierarchy should ultimately be able to resolve the domain name. In this way, the DNS service is distributed all over the world, and DNS lookups are routed up and down the hierarchy to different DNS services, each with separate areas of "expertise." In practice, that's more efficient than trying to have a central repository for all DNS mappings. It's pretty amazing that such lookups might bounce around the world in a fraction of a second, and most people don't even realize that's happening when they surf the web.

For increased efficiency, DNS implements a feature called *caching*. In computer terminology, *cache* (pronounced "cash") refers to a temporary storage location. When a cache gets full and new

data needs to be added, the oldest data is deleted to make room for the new. In this way, a cache will always contain the most recent data. In the case of DNS, recent IP lookups are cached on the local DNS server. So for example, when someone at UWeb first surfs to *www.google.com*, UWeb's DNS server will cache the IP address it obtains through the hierarchical lookup process described above. So next time someone surfs to *www.google.com* at UWeb, the local DNS server can grab the IP from its cache rather than having to ask other DNS servers to look it up again.

In the early days of the WWW, it was proposed to create a top-level *.xxx* domain for pornography. People argued that would make it very easy to regulate online pornography, and especially easier to program filters to keep children away from it. Basically, it would have to be published in the *.xxx* domain, or it would be illegal. On the other side, people argued that creating an *.xxx* domain would increase the visibility of pornography, cause more of it to be published, and turn what wasn't currently that big of a problem into a huge online industry. Back in the day, the latter argument won out and an *.xxx* domain was never created. Who do you think was right?

Virtual Hosting

DNS allows for different named addresses to be mapped onto the same IP address. Thus, multiple web sites can be *hosted* on the same server. Using one server to host multiple web sites is often called *virtual hosting*, as depicted in Figure 10. All 4 named addresses you see in Figure 10 get resolved to the same IP address. Even though all 4 FQDNs map to the same IP address, the server is configured to send each one to a different folder, the *root directory* for each distinct Web site.

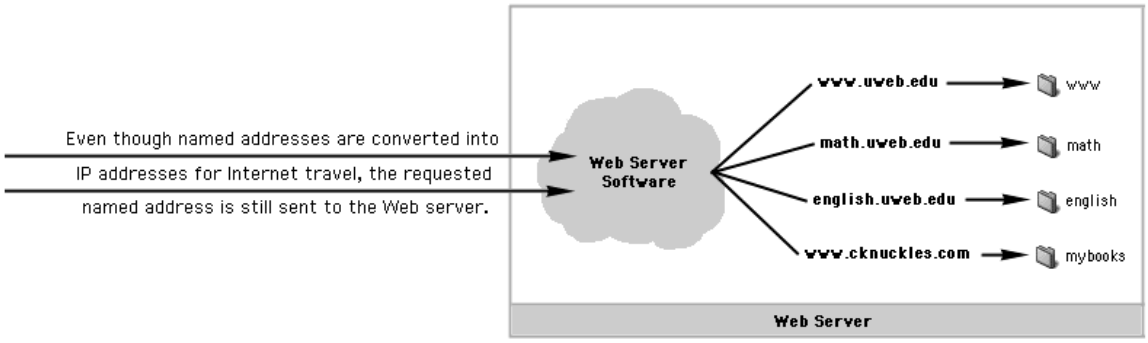

Figure 10 Different Web Sites can be Virtual Hosted on the Same Server

Student web server accounts are also a form of virtual hosting, sometimes called *account based hosting*. It's not practical to create an FQDN like *jsmith.uweb.edu* for each student in a web programming class, for example. So generally students are given a standard login account on a web server. The public address for *jsmith's* public web site is then something like *www.uweb.edu/~jsmith.* All student accounts use the same FQDN, and the server takes care of the rest. In this case, all the different students' web sites are simply contained in different root directories on the same server.

The Uniform Resource Locator

Current browsers are pretty nice. If you type *www.uweb.edu* into the address field, the browser will perform the courtesy of changing your request into the form *http://www.uweb.edu*. The browser changes the named address into a *Uniform Resource Locator (URL)*. A URL provides the *how*, *where*, and *what* parts of a request as shown in Figure 11.

Figure 11 The Three Parts of a URL

The *how* part tells the server what type of transaction to initiate. In *http://www.uweb.edu*, the http part stands for *hypertext transfer protocol* – basically give me a web page. When you type *www.uweb.edu* into a browser, it assumes you want a web page, so that's why it automatically forms the *http* URL for you. When your browser is showing a lock, it's using the secure *https* protocol, which works exactly like *http* except that a secure socket is used to encrypt the packets during transit. There are other protocols used in URLs. For example, a URL like *ftp://www.uweb.edu/path/to/file.mp3*, specifies the file transfer protocol to initiate download of a file to your downloads folder. You typically don't directly type an ftp URL into your browser. When you click a download link, the browser grabs the file in the background using an ftp URL.

The *where* part of a URL is simply the address of a server or other resource. The *where* part is usually a named address, but it can actually be a raw IP address as well. As for the *what* part of a URL, you first have to understand the notion of *directory paths* and the hierarchical organization of directories (folders). Figure 12 shows two different graphical representations of the same directory structure.

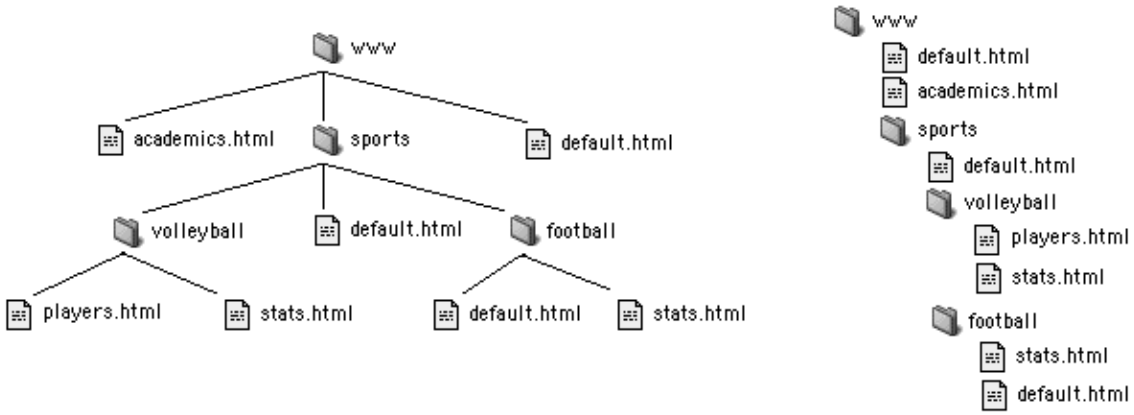

Figure 12 Two Visualizations of the Same Directory Structure

In Figure 12, the *www* directory is the root directory for the web site of the fictitious UWeb. As they developed their Web site, they created more and more folders to organize all of their web pages. The natural way to view the directory structure is a hierarchical "tree" as shown on the left of Figure 12. In the tree representation, it is easy to see the containment relationships among the folders, and which files are in which folders. The compact representation on the right is more like how your operating system shows you directory structures.

The 1st URL shown in Figure 13 basically says give me a web page in the root directory of the *www.uweb.edu* web site. But there is more than one page in that directory, so the server would automatically serve up the default file. That is, because of the default page, the following two URLs are equivalent.

http://www.uweb.edu
http://www.uweb.edu/default.html

That's convenient so that the 3rd URL in Figure 13 can pull up a sports page without specifying the name of a specific file.

URLs like the 2nd and 5th ones in Figure 13 point to specific non-default files. Usually, you would surf to the default page in a given directory, and then links in that page would transfer you to the other files so that you don't have to actually know the file names on the end of the URLs. Notice the forward slash (/) in the directory paths. Each / goes one level deeper in the hierarchical directory structure. So the long URLs are requesting resources several folders deep within the site.

Finally, consider the 4th URL in Figure 13. Since it ends in /, the URL is requesting a directory, rather than a specific file. If that directory happened to have a default file, it would be served up automatically to represent the directory. But it doesn't. Some web servers are configured to serve a generic web page that lists all files in that directory as links. Other servers are configured to give a message something like "permission denied" when a directory is requested that doesn't contain a default page.

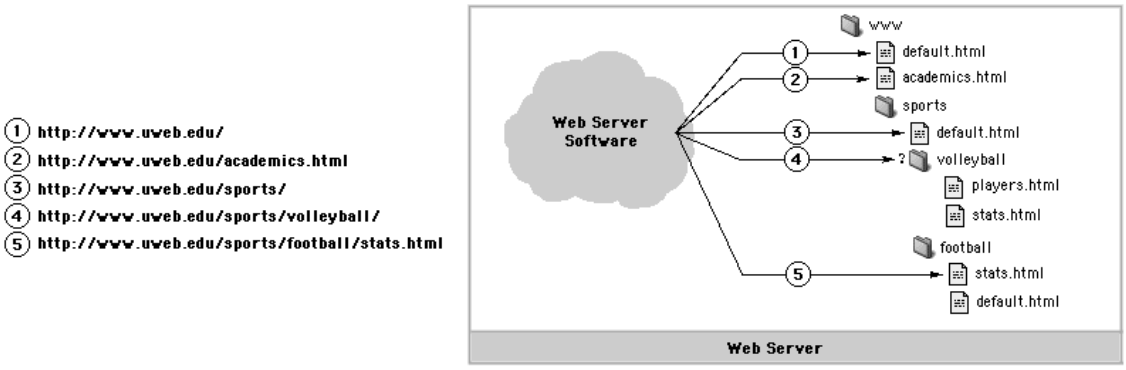

Figure 13 The *what* part of a URL is the Path to a Specific Resource within the Web Site.

A final point about URLs is that you have probably noticed that they may even contain more information than the how, where, and what parts discussed above. If you aks the Google search engine to search for *wildebeest*, a URL like the following (probably with even more stuff after the ?) results in the browser's address field:

*http://www.google.com/search**?**q=wildebeest*

The data after the question mark is called a *query string*. Recall the earlier discussion of the Deep Web and that that many "web pages" are actually computer programs that search databases and create web pages on-the-fly to send to the browser. A query string contains input data passed to such a program. In the case of a search engine, that data is used to *query* the database for matching information. Hence the term *query string*.