Chapter 2.3 Internet formats and protocols

Preview

The last chapter described the structure and operation of a computer. This chapter describes how computers are used to format messages and route them over the Internet.

Only one Internet

The Internet is important because so many computers are connected to it. If it were not possible to transmit a message from a computer on one network to a computer on almost any other network around the world, the "I" in Internet would not be capitalized. There wouldn't be just one Internet, and it wouldn't be so important to be on that one Internet.

If millions of people were not using the Internet, the cost of sending an e-mail message from San Francisco to Paris would be high. The absolute cost of the transmission link is huge, but because it is shared by millions of people the cost of one e-mail message is very cheap. The cost to send one message is so low that it isn't cost effective to bill the sender by size and distance traveled; there isn't even the infrastructure to bill at that level. Internet users that connect using a telephone modem are charged by the hour because the telephone line to the Internet Service Provider is tied up for this time, not because the Internet itself has an hourly charge. Of course some one eventually does pay for the networks that carry Internet traffic. Internet Service Providers, large businesses, universities, and governments pay fees, based on average traffic flow, to be connected to an Internet link.

Growth of the Internet

The Internet grew proportionally more in a few years than the telephone system did in as many decades. This was possible because the US Government funded its initial development and later provided financial incentives for commercial software companies to include Internet software with their products, and financial disincentives if they did not. The motive of early work on packet systems was to provide a communication network that could function after a massive atomic bomb attack on the country. However, ARPA net, the direct predecessor to the Internet, was built to allow researchers to share computer systems that had been funded by the US Defense Department. No one had to write a business plan, and the system wasn't supposed to pay for itself anyway.

The Internet uses formats and protocols which allow Internet data to move across networks built for other purposes. Most of Internet traffic, at least in its early years, moved over networks built by the telephone companies for voice communication, with Internet data typically mixed in with digital voice data. However, the volume of Internet traffic has become so large that it is now the driving motivation for expansion. The tail has become the dog. In addition, the distinction between voice and Internet data has become very fuzzy indeed, as an increasing fraction of voice data uses Internet protocols, and computers on the Internet are used to transmit voice messages.

Thus the Internet is not so much a network of computers as a network of devices. However, as mentioned in the previous chapter, it is not easy to distinguish computers from other digital devices anyway.

Internet over lower formats

Exchanging messages between millions of people around the world might seem to require a single format. However, the sender, receiver and intervening networks often use technologies that are so varied that a common format would neither be technically possible nor desirable. Even if technically possible the history and politics of different networks and their associated formats often make change difficult. The problem is solved by layering higher level Internet protocols over the native formats of the individual networks. Only the lower level layers are changed as the message moves toward its destination.

Topology of the Internet

There are several basic topologies that can be used to link a group of computers together.

Figure netTopo

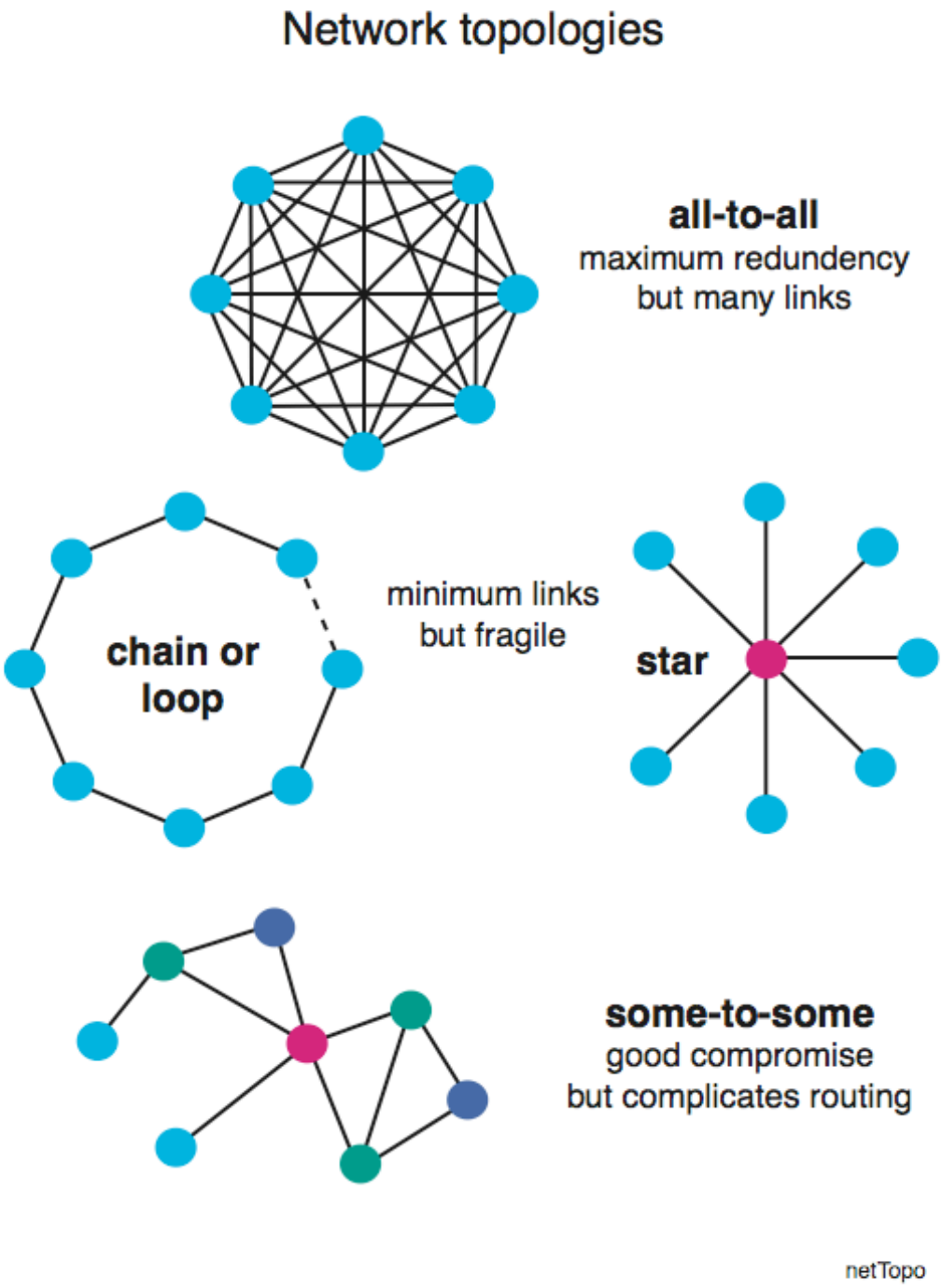


Figure netTopo. The three basic ways computers (or anything) can be linked together are all-to-all, chain, and star. The all-to-all topology requires too many links if the number of nodes is even modest. The chain and star topologies are fragile in that a break in a few loops can cause nodes to be isolated. The some-to-some pattern, while rather vaguely defined, has a good deal of redundancy, and the many alternative paths from one node to another make it possible to shift traffic from one path to another to avoid bottle necks.

All units linked to each other

This topology is least sensitive to breaks in links and has the greatest routing flexibility. In addition, there is no need for a node to forward a message, since there is always a direct route from one node to another. However, the number of links becomes impractical when the number of computers becomes even moderately large. The number of links increases as the square of the number of computers, with one hundred computers requiring approximately 5,000 links, one thousand computers requiring 500,000 links, etc.

Star: units linked to a master computer

The master computer becomes heavily loaded if the number of satellite computers becomes large, i.e. the topology does not scale well. In addition, if the master computer fails all is lost.

Chain: each unit linked to another

If any single link is broken the network is split into two groups that can not communicate with each other. If one more link is added the chain becomes a loop, and now it takes two breaks to isolate any group of computers. However, on average half of the total data exchange flows through any given node, thus, all the links and nodes must be able to handle this amount of traffic.

Composite

The composite topology has the greatest flexibility and is the one used by the Internet. Several high capacity links can be created between computers that exchange a large amount of traffic and have essential functions, while a computer that is of minor importance may be connected to the others by one low capacity link. However, the price for this flexibility is the need for a sophisticated data routing scheme.

On the Internet routing decisions are made as a message makes its way from sender to recipient. In fact, the flow of traffic itself modifies the routing pattern in order to minimize the transit times of future messages. The autonomous nature of message routing makes the network fairly resistant to failures of specific links and nodes. The Internet routing scheme allows new users and nets to be easily added. As we will see, the only obligation of a new member is to use Internet formats, protocols, and an address that is not already in use. Even authority for assignment of new addresses is decentralized and expands along with the growth of the Internet itself.

Protocol layers

The tasks needed to transmit a message across the Internet are specified by multiple, nested protocols with their associated formats. Protocols are grouped into

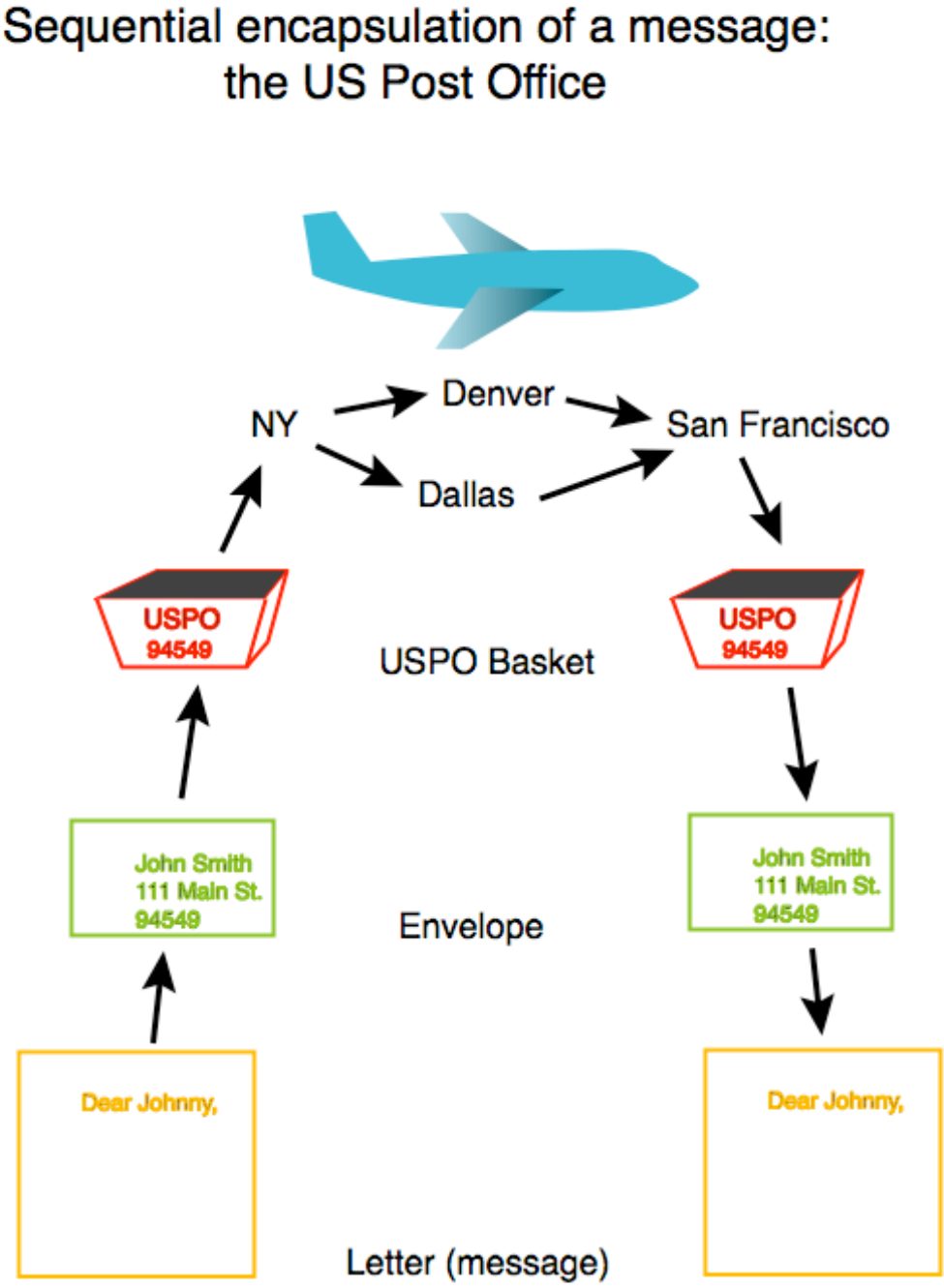
levels according to the type of function they provide. Thus, all protocols at one level generally accomplish a similar general task, but use a different method or provide different features. Even on the same network different messages can use different protocols, and a single message may use different protocols as it moves through different segments of the Internet. The number of format levels for a message is not absolutely fixed, and there are different systems for defining and naming levels. Software engineers and computer scientists occasionally disagree and often have heated arguments about the level that a particular protocol or function belongs in. We will describe the levels most appropriate to the Internet. Perhaps the easiest way to explain nested protocols is to use an analogy, the postal system.

The post office uses protocol layers

We use nested protocols or structures analogous to then, in everyday life.[1] Let's follow the delivery of a letter by the Post Office to illustrate protocol layers.

[1] By everyday life I mean the non-computer world. This phrase is somewhat of an oxymoron, since it would be difficult to avoid the implicit use of a computer in a day, since it would mean not using a telephone, automobile, bus, a retail store, or the Post Office.

Figure uspo



uspo

Figure uspo. As a letter moves from source to destination it is placed in containers with increasing general addresses and then removed in the inverse order to reveal increasingly specific addresses.

At the bottom left of Fig. uspo, we start with the message, or letter, which is addressed to Johnny. The letter contains all the information Johnny probably cares about, but it can not be delivered to him by the Post Office because his address is not given. The writer could just add the address at the top, but the private content of the letter would be there for all to see, and the postal sorter would also have to hunt around for the address. The better solution is to encapsulate the letter in an envelope, which only reveals John's official name and physical address. To make a better analogy with the Internet let's assume (you will see why later) that the letter writer does not know the ZIP code of the destination, and is too lazy to look it up. Not to worry, it will be added at the Post Office, because the number is so more convenient for their internal use that it's worth the trouble for them to look it up. We also add the sender's name and address. In case something should go wrong with the delivery the letter can then be returned stamped “Not at this Address”.

Now, the letter can be delivered, but it is very convenient for the Post Office to first gather together all the letters for the same general location into one container. In this example I have ignored the sophisticated distribution system the Post Office actually uses, and have just put all letters that will go to the same local Post Office, which is specified by the ZIP code, into one container. In this mail container the addresses of the individual letters are hidden, but that is all for the best at this stage of mail delivery, since it would just confuse the next link in the transport chain.

This container is then carried to the airport, where the air-freight staff decide which airplane to use to carry it to the destination Post Office. At different times of the day, or days of the week, a container addressed to the same ZIP code may take different routes, perhaps stopping at one or more intermediate cities to be transferred to another airplane. At the end of the plane trip, the whole process of encapsulation is reversed step by step. Finally, Johnny gets his letter.

Nested data protocol layers used to carry a message through the Internet are analogous to this example. Each layer is added to accomplish a certain function, and later removed in the process of getting the message through a section of the journey to its final destination.

The voyage of a typical message

The transmission of a message is typically a complicated process, even at the level of protocols, without any consideration of implementation by hardware. As the message proceeds along its route, it is processed repeatedly to encapsulate it into a new protocol or remove an existing protocol layer. Our goal will be to get a flavor of how the Internet works by following the voyage of a typical message.

The physical layer

The physical layer is the lowest protocol level, the one most tightly associated with specific hardware. The low-level protocol is analogous to the airplane in Fig. uspo; it encloses and carries everything else. Thus our discussion of Internet protocol layers

will start at the equivalent to the top of Fig. uspo and will work down to the actual message.

The physical layer is responsible for the process of actually delivering a stream of bits; all higher protocols assume a stream of bits as input. This layer is always active, even if there is no message being sent at the moment. It is not concerned with addressing, it runs on a real physical link with one input and one output. The physical layer provides a time frame, or clock, that enables sequential voltage pulses and gaps to be interpreted correctly, and it typically also provides error correction.

To detect individual pulses the receiver must associate the level at a given time with a specific pulse in the stream of data. One might propose using highly accurate clocks at both transmitter and receiver to identify individual pulses by absolute time of arrival. However, extremely accurate clocks are expensive, and more importantly, the transmission times for the pulses are unknown and vary even within a message.

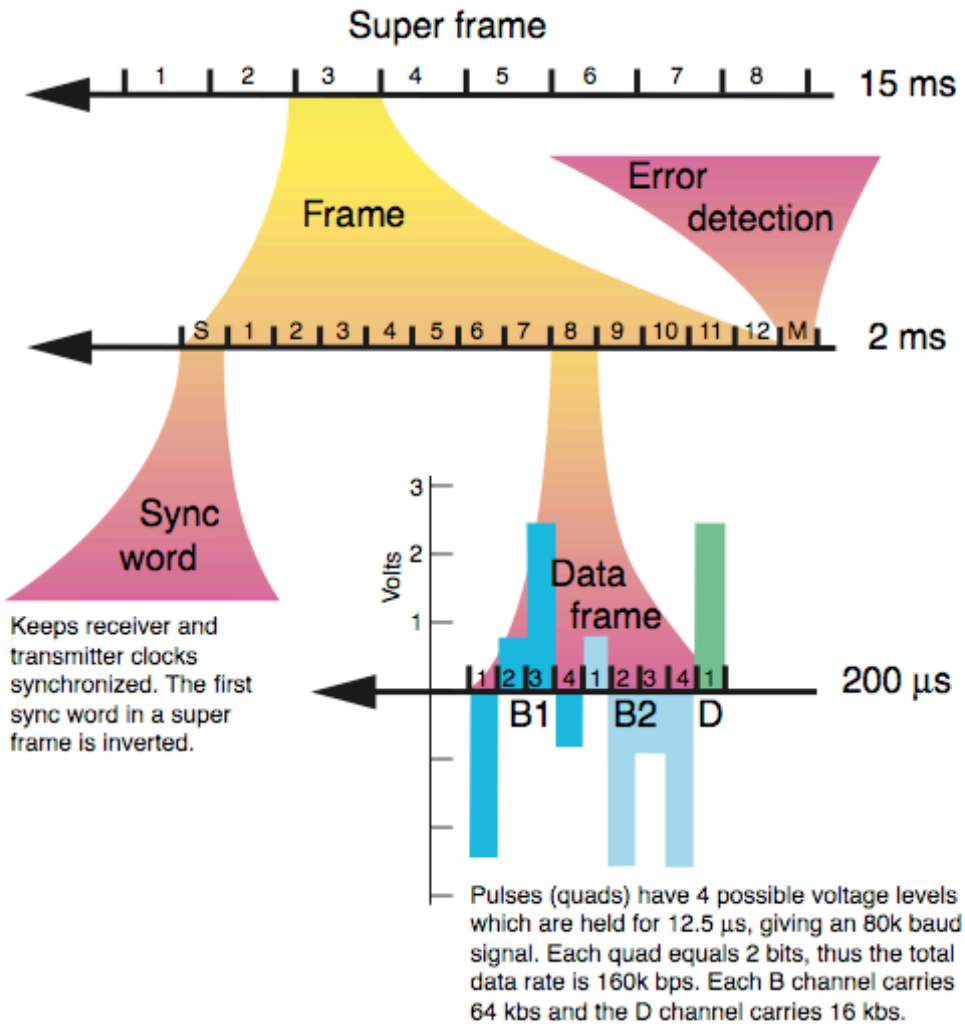
The solution to the timing problem is for the receiver to use a local clock which runs at a predetermined standard rate but is frequently reset to the "correct" time by the incoming bit stream. A specific bit sequence signals the receiver that a block of data pulses follows, a second start sequence marks the start of the second block of data, and so on. Thus the local clock is needed only to define the pulses in between start sequences.

Errors are made during detection of the pulses. Error rates during digital transmissions are highly variable and depend on the specific technology used and the length and quality of the transmission medium. High quality optical links may loose only a few out of many billion bits a day, while transmission over rural telephone lines may be very poor. The error problem is so ubiquitous that most protocols have some error detection mechanism. If an error has occurred, the group of pulses is usually just retransmitted. In some cases enough redundancy is provided by the protocol that simple errors can be corrected without repeating the transmission.

ISDN BRI as an example

The Integrated Service Digital Network (ISDN) is a collection of protocols originally developed and used by telephone companies for voice channels. One common version of ISDN is the Basic Rate Interface (BRI). This protocol can be carried by a standard twisted wire pair, as long as you are within about two miles of the local exchange. BRI consists of three channels, two bearer channels that can each carry 64 kbps, and one data channel with a capacity of 16 kbps. Each B channel can be used to carry voice communication, while the D channel is typically used for call initiation and termination. However, the D channel can also be used to carry the data from a credit card reader or other device that doesn't handle a large amount of data.

Figure frame_isdn



frame_isdn

Figure frame_isdn. A BRI ISDN signal consists of a series of voltage pulses with four possible levels and a length of 12.5 μ seconds. A Super Frame contains 8 Frames. Each Frame starts with a word containing a unique pattern of pulses that resets the internal clock of the receiver so it is in exact phase with the following data. The data is divided into words containing 4 pulses from channel B1, 4 pulses from channel B2, and one pulse from channel D. A group of pulses at the end of the frame provides error detection.

The implementation of the BRI in the United States is illustrated in Fig frame_isdn. The information is encoded as a series of pulses that are each 12.5 (s long; thus there are 80,000 pulses per second. Each pulse can have one of four voltage levels, and thus transmits two bits of information. Nine pulses constitute the basic group, with the first four belonging to one B channel (B1), the next four to the other B channel (B2), and the last pulse assigned to the D channel. Twelve of these groups are transmitted sequentially to form a frame. However, each frame starts with a special group of nine pulses to synchronize the receiver clock, and each frame ends with a special group of three pulses that provide error detection. Eight frames form a super frame, with pulses of the initial clock synchronization group inverted to identify the start of the super frame.

Each B channel carries 32,000 pulses per second, or 64 kbps, since each pulse carries two bits of information. When carrying a voice signal, the 64 kbps is generated by sampling the analog signal 8,000 times a second, and representing the value at each sample time with 8 bits or 256 levels. This is sufficient to represent frequencies up to about 4,000 Hz. The result is a far cry from high fidelity, but you can understand the conversation. Of course, if a B channel is carrying computer data, the bit stream usually represents something other than a voice.

Non-computer uses of a ISDN BRI

In a typical home or small office the BRI could be used by two phone sets or a phone and a fax. The two B channels are associated with two distinct telephone dial numbers, since they can be used individually and simultaneously as voice channels. A "terminal device" must select one of the two sets of four pulses, and the last pulse from each basic group of nine, and route them to the appropriate device. In our example, the devices are analog telephone and fax units, so the information in the pulses must be converted into a conventional analog signal. A typical terminal device will also perform the digital to analog conversion. There are digital telephone handsets, but they tend to be more expensive than the analog variety. In any case, the digital pulses must be converted somewhere to an analog signal to drive a headphone or speaker.

There is some irony in the conversions that take place in the use of a conventional (analog) fax machine on a BRI. The paper containing the image to be transmitted is scanned along parallel horizontal lines and the reflectivity of the paper at each successive spot is represented as a single bit (at least in the most commonly used black-white mode). This digital signal is then compressed with a simple algorithm, and converted to a modulated tone, which is normally sent out on the analog phone line. Using our ISDN system however, the modulated tone is converted at the terminal device back to a digital signal to be sent out over the ISDN line. If the destination FAX machine is also analog, the digital ISDN bit stream will be converted to an analog signal, the back to a digital signal inside the FAX machine so the black-white image can be generated on the paper. Wouldn’t it be easier if everything was digital?

Attaching a computer and channel bonding

When a computer is connected to the BRI the stream of binary pulses from the computer is fed directly to the terminal device. It is then converted to quad pulses and sent out to the ISDN line on one of the B channels. Now we have a simple digital to digital format conversion.

The computer can also make use of a powerful feature of the ISDN protocol: channel bonding. Most PCs are capable of transmitting data at speeds greater than 64 kbps. We have a capacity of 128 kbps using both B channels of our terminal device; with the right software and the support of ISDN protocol we can "bond" channels B1 and B2, and use the entire eight quads of each group to connect our computer to the Internet. Bonding can be done on other ISDN interfaces, such as the Primary Rate Interface (PRI), sometimes called a T1 line. A PRI contains 23 B channels and one D channel. When all 23 B channels are bonded, a channel with a capacity of 1.472 Mbps is thus created. Unfortunately, the rate charged for a PRI are also significantly higher than the rate for a BRI.

Digital Subscriber Line (DSL)

ISDN is only one method for digital communication on telephone lines. An alternative, Digital Subscriber Line (DSL) service is fundamentally different. A PC attached to a DSL line transmits digital pulses to a different type of "terminal device", which transmits binary digital pulses over the phone line at a high rate. The pulse frequency is significantly higher than frequencies used in normal analog voice or fax communication, and frequency filters at both ends of the local phone line isolate the normal analog devices from the DSL pulses. Thus, the frequency difference between DSL pulses and conventional analog communication means that one pair of wires can carry DSL and voice data at the same time.

However, the more important feature of the DSL digital stream is that it is taken off the line at the local switching office, before the switches, and fed into a packet based digital line. A packet line transmits data from many sources, effectively simultaneously (the Internet is a packet system). Thus, when you use a DSL to connect to the Internet, there is no telephone number to dial, because you are not really using the telephone system, you are just sharing the twisted wire pair used by the telephone system. The net effect is that you are "always connected" to the Internet when your computer is on.

The data link layer

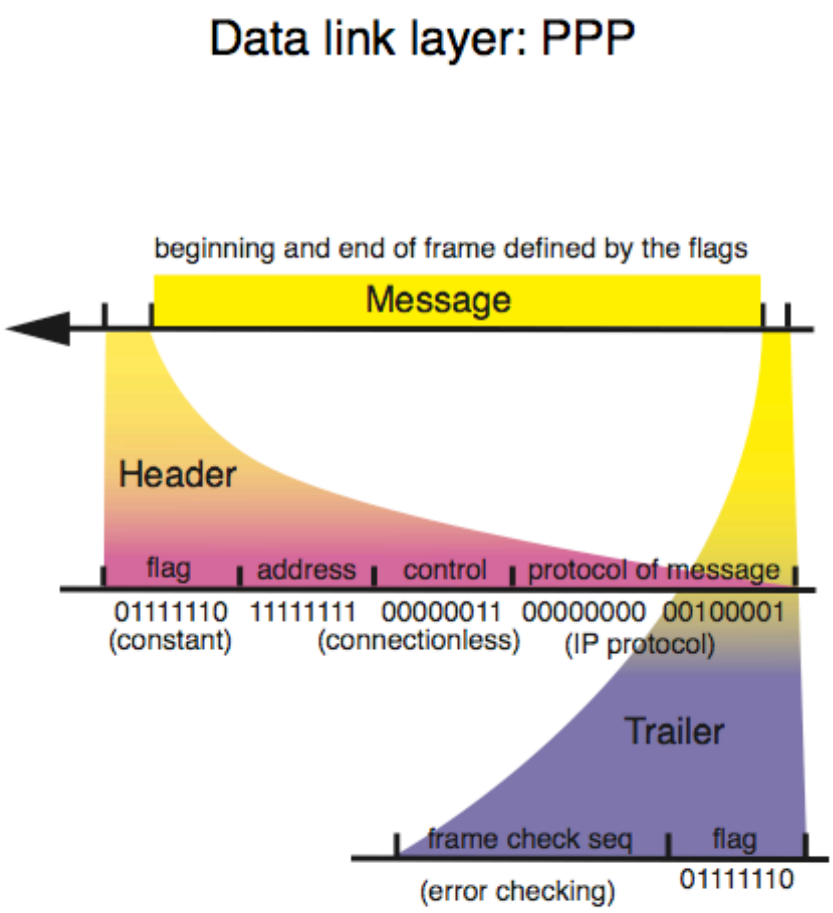
The physical layer carries digital information, any information. If you have a telephone connected to the line the digital information is just a digital form of the analog voice signal. The telephone system has no knowledge of the content of a voice message, and so it just transmits everything, including the quiet sounds that are really noise and not speech. If you make a telephone call you pay for every minute, the silent seconds count as much as the loud ones. The data sent by a computer is very different than a voice message. There is nothing analogous to the volume of an audio signal in a computer message, the computer is either sending data or it isn't, and you know when it is sending data. Very often the computer isn't sending any data; the data is in the form of short bursts. The data link layer, which is encapsulated in the physical layer, is

only active when data is being sent. This feature allows communication hardware to handle many more channels than would be possible if all incoming channels were considered to be always active.

PPP as an example

The Point-to-Point Protocol (PPP) is commonly used over telephone lines. In analogy to ISDN, PPP transmits data in frames, however the PPP frame is only transmitted when there is data to be sent, not continuously as the ISDN quads are. As seen in Fig. Frame_PPP the PPP packet is not a fixed length, but is rather defined by the unique start and stop bit pattern: 01111110. However, any protocol that uses a variable length frame and a fixed bit pattern for the frame delimiter must have a method for handling a message that just happens by chance to contain the delimiter bit pattern. The basic solution is to precede an internal data 0111 1110 with a special pattern that means it should be processed as data and not used as a delimiter.

Figure frame_ppp



frame_ppp

Figure frame_ppp. In this format the beginning and end of a frame is defined by a special sequence, and the frame can thus have a variable length. The format provides space for an address, but this is not used for Internet connections. The fact that the message is in an Internet protocol is indicated by the protocol sequence.

The PPP frame contains address, control, and protocol fields before the message. The address field does not actually contain an address in this example, but is just filled with 1s. The control field is 0000 0011, which indicates that this is a "connectionless" protocol; each frame is independent of all others. There is no assumption that the sender and receiver have a constant connection over which successive portions of the message are transmitted. This concept is fundamental to the structure of the Internet, but it is more fully implemented in higher layers, so we will not discuss it here. The protocol field contains the bit pattern 0000 00000010 0001, which indicates that the next higher protocol is the Internet protocol. The Frame Control Sequence (FCS) field at the end of the message contains information that allows frames with errors to be detected and discarded.

However, PPP does much more than just carry the message. Before any message data is transmitted, special PPP frames test the performance of the physical layer, i.e. determine the error rate, and then communicate with the receiving software to negotiate options for establishing the link. Then PPP frames transmit a password to prove that the sender has the privilege to use the link. The Password Authentication Protocol (PAP) sends the password as text, while the alternate Challenge-Handshake Authentication Protocol (CHAP) uses a one-time code for the password. If the CHAP coded password is intercepted by an intruder and used again, it will not be accepted. PPP can also set up a Multilink Protocol (MP) which allows data to be sent over multiple channels. Long packets will be fragmented and reassembled at the receiving end of the link. PPP is thus the glue that allows multiple ISDN B channels to be used to transmit data at the sum of the speeds of the channels. Finally, PPP must be able to terminate the connection to free up the receiver for other users.

The network layer

This is the layer that defines the Internet. Messages carried by the Internet are in the form of packets, and each packet contains the address of its destination. The alternative to packets is a circuit, conceptually simple and simple to implement.

Circuits: the telephone system

The original method for routing on the telephone system was to create an actual wire circuit between the two parties. The first step in making a call was to take the receiver off the hook and turn a crank on the side of the phone box. The crank generated sufficient current to ring a bell at the operator's switch board, and she (it was always she) could identify the calling line because the up position of the hook on your phone had turned a light on next to your jack. She then plugged her head set into your jack and asked whom you wanted to call. When you told her she connected a generator line to the destination jack and rang their phone. When someone answered the phone she connected the jacks of the two parties with a bus line (sometimes forgetting to remove her plug from the circuit). At the end of the call, when both parties hung up their receivers on the hooks, the lights would go off at the operator's

panel. All plugs would then be removed to free up the lines for the next call. The start and end of a call are called call setup and teardown.

If the call was not local, i.e. the sender's and receiver's jacks were not both on the operator's switch board, the operator would have to call at least one other operator and ask that a circuit be set up to the destination phone. A long distance call could require a sequence of these links to be established, each link requiring an operator at the end of the chain to call the next operator and request the next link. As might be imagined, this process could take some time. Thus to place a call from the west coast of the United States to the east coast required calling the operator and placing the call. You would then hang up and she would call you back 5, 10, or 20 minutes later, when the complete circuit had been set up and inform you that your party is on the line.

The most recent incarnation of the circuit method of routing replaces literal with virtual circuits. A virtual circuit is established on a sequence of links that carry many message streams simultaneously in the form of data packets, with each packet containing a portion of one of the message streams. The call set up now consists of sending a setup request to the router computer that connects to the next level of links. The request specifies the final destination and a data transmission rate. If the router has sufficient unused capacity to carry this transmission, it reserves that capacity, and sends the request on a router further along a path to the final destination. This process is repeated until a complete route has been reserved, and then the sending unit is notified that a circuit has been set up. Even though the circuit is “virtual” it still exists as a real collection of committed resources through out the duration of the call.

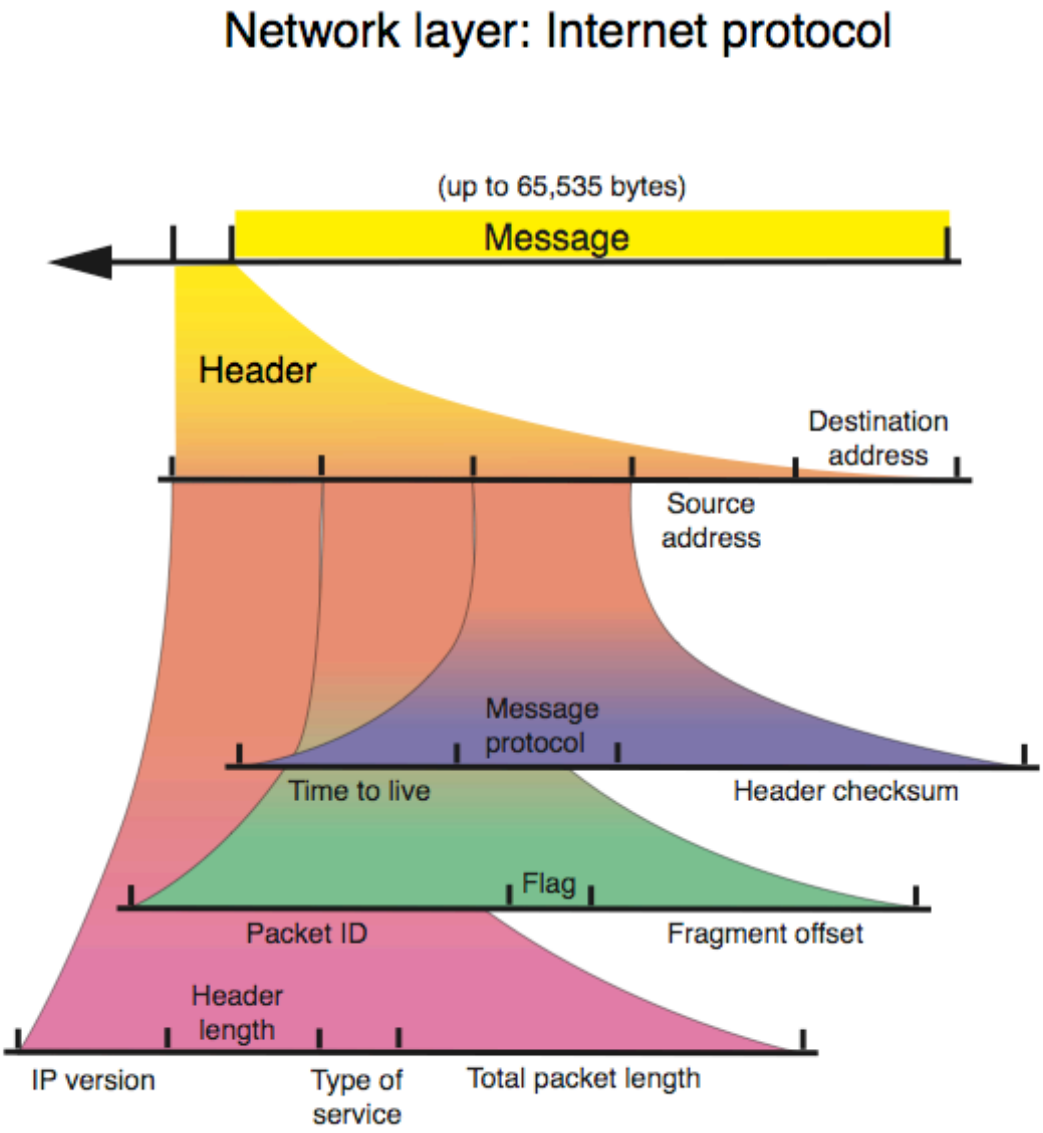
Packets: the Internet Protocol, IP

The essence of the Internet is its connectionless packet routing scheme, which is implemented with the IP protocol. It is connectionless, because, in contrast to the circuit method of transmission, no specific physical or logical path is set up before the transmission starts. Packets containing the message and address are just transmitted, and the routers along the way attempt to send them to the destination. The Internet only guarantees an attempt at delivery. Since no specific circuit is set up, packets containing parts of the same message stream, but sent at different times, may travel to their destination by different routes, may arrive out of sequence, or may never get there at all. The power of IP is that the sender need not know anything about the route topology of the Internet in order to send a message. Even the routers that pass along the packets do not need to know the entire topology, they only need information about nodes they are connected to and a little guidance about which classes of addresses need to be sent to which nodes. As we will see later, most of this information is actually collected and periodically updated by the router from its neighbor routers after it is connected. The first time user of the Internet is often bewildered and dismayed by the fact that there is no central authoritative directory of everyone and everything on the Internet. While it might not be much comfort to the novice, no other person or machine has a complete directory of the Internet either.

The possibility that a packet may not be delivered to its destination may seem so inimical to a useful communication system that it would prevent IP from being used, much less being the foundation of the Internet. However, we will learn in the next section that it is possible to insure that almost always, almost every packet is delivered. That guarantee is enough to make the Internet as useful as it is, but

completely reliable communication is the responsibility of the next higher protocol layer.

Figure frame_ip



frame_ip

Figure frame_ip. The most important contents of the header are the length of the packet and the address of the source and destination. However, it also contains many other parameters that enable it to be delivered over the Internet.

The IP header

Since it is so central to the Internet, the structure of an IP packet will be described in more detail than the previous, lower level protocol examples. The IP packet header consists of five or six 32 bit words, as illustrated in Figure frame_ip. The fields are:

Version: Presently it is 4 or 6. Since the version number is in the header, properly programmed routers can carry a mixture of both versions, which means that the switch over from 4 to 6 can occur over a considerable length of time. Since even the length of header is variable (specified in the header) it should be easy to transition to even more elaborate IP headers. The following refers to IP version 4 only.

Header length (32 bit words): Typically 5, but can be 6.

Type of service: Request to router for a connection with low delay, high data rate, high reliability, or low cost. These requests are ignored by most of today's routers!

Total packet length (in bytes): Since the length is specified in the header, no trailer is needed. This field is 16 bits long and thus the length can be up to 65,535 bytes. Few networks can handle contiguous packets this long, so the following 32 bit word assists fragmentation of the IP packet.

Packet ID: Unique number for this packet; often a sequence number.

Flg: 001 = not last fragment; 000 = last fragment; 010 = do not fragment.

Fragment offset: Position of this fragment in the original packet.

Time to live: The sender typically sets this to a value of from 40 to 64, and it is decremented by each router. When it equals zero the packet is discarded, thus keeping packets from being trapped in endless loops.

Msg protocol: Encapsulated message transmission protocol, e.g. TCP, UDP.

Header checksum: Verifies header has not been corrupted.

Source address: The IP address of the sender, e.g. 199.108.14.219. The structure of an IP address is discussed in the next section.

Destination address: The IP address of the destination.

Options: Used for diagnostic purposes, e.g. PING.

Padding: To make header length a multiple of 32 bits.

The IP address: how to write it

As we saw in the previous section, the version 4 IP address is a 32 bit word. To make the address more friendly to humans it is usually written as four 8 bit bytes separated by dots (the dots are thus not decimal points, just delimiters). Each byte is written as a decimal number, i.e. to the base ten, and thus has a range of 0 to 255. A typical IP address is: 199.108.14.219. An address of 199.666.14.219 is not possible because 666 (decimal) is greater than 255, and thus can not be represented by one byte. If one of the bytes is zero, a zero is indicated so that there are always four fields in the address.

The IP address: how do you get one?

A block of sequential IP addresses is assigned by the Network Information Center to an organization, corporation, or government that requests them. These groups then distribute IP addresses in smaller blocks to their divisions that distribute them to smaller divisions, and so on down to the individual host. Thus, assignment of IP addresses starts with a central authority but is then delegated to other groups down a chain. As we will see, the groups on this chain of distribution are responsible for establishing and maintaining the routers that deliver messages to hosts on this chain. They are also responsible for the domain name servers that translate domain names into IP addresses.

The IP address: what does it mean?

As described in the previous section, blocks of IP addresses are given to organizations which then distribute blocks of IP addresses to smaller groups, and so on until a user finally gets an address. However, the number of IP addresses given to these initial groups are not at all equal, since the number of users in the groups cover a wide range. One organization might be a country, say France, while another might be a United States university. Thus, you can't say that the first N digits of the IP address represent one of the initial groups, it's more complicated than that. Class A addresses are assigned to the largest groups, class B to the next largest, and class C to the smallest. However, unless you are assigning IP addresses, you really don't want to know the details.

IP addresses for dial up users

Many users of the Internet connect only a few times a day by connecting to an ISP using a telephone, i.e. dialing up the ISP. It would be wasteful to assign a permanent IP address to each of these computers; instead they are assigned a new IP address each time they log in to the ISP. The ISP then needs only enough IP addresses to handle the maximum number of users that can connect at the same time. These users do however need a unique address for e-mail but that takes the form of a "user name" concatenated with the URL of the ISP. All e-mail is sent to the IP address of the ISP and then sorted out to users.

When you do need your own IP address?

If you want other computers on the Internet to be able to send packets to your machine 24 hours a day 7 days a week your machine must always be running, you need a permanent connection to the Internet, and you need your own IP address. Otherwise it would be like starting a restaurant in a trailer that moved around to unpredictable locations and was only open at random times. However, establishing a full time presence on the Internet can be done in several ways, and some of these requirements can be satisfied indirectly.

The most straightforward but expensive alternative is to establish a 24 hour a day connection to the Internet, e.g. using a DSL connection via your telephone company, and using a machine at your location as the server. You can then communicate with visitors to your web site as they connect and send information to you, and you can change the computer hardware whenever you want. The ISP is essentially acting as a router for you, it just sends data your way and sends your data to the Internet.

Another way to maintain a web site is to take your computer to an ISP (or rent a computer from them). They will link it to the Internet, keep it cool, dry and make sure it gets electric power. You can then periodically send (upload) material for the web site from your office or home.

A less expensive method to maintain a web site is to host the site on your ISP's computer. You can still have your own domain name and a visitor won't know that your site is physically on the ISP computer.

The least expensive is to have your web site on the ISP machine and use their domain name, with a mock domain name as a subdirectory. Now a visitor will know you don't have your own machine or domain name, but you may not care. Some ISPs will not even charge an extra fee for this service, although they may place advertisements on your site; very tacky!

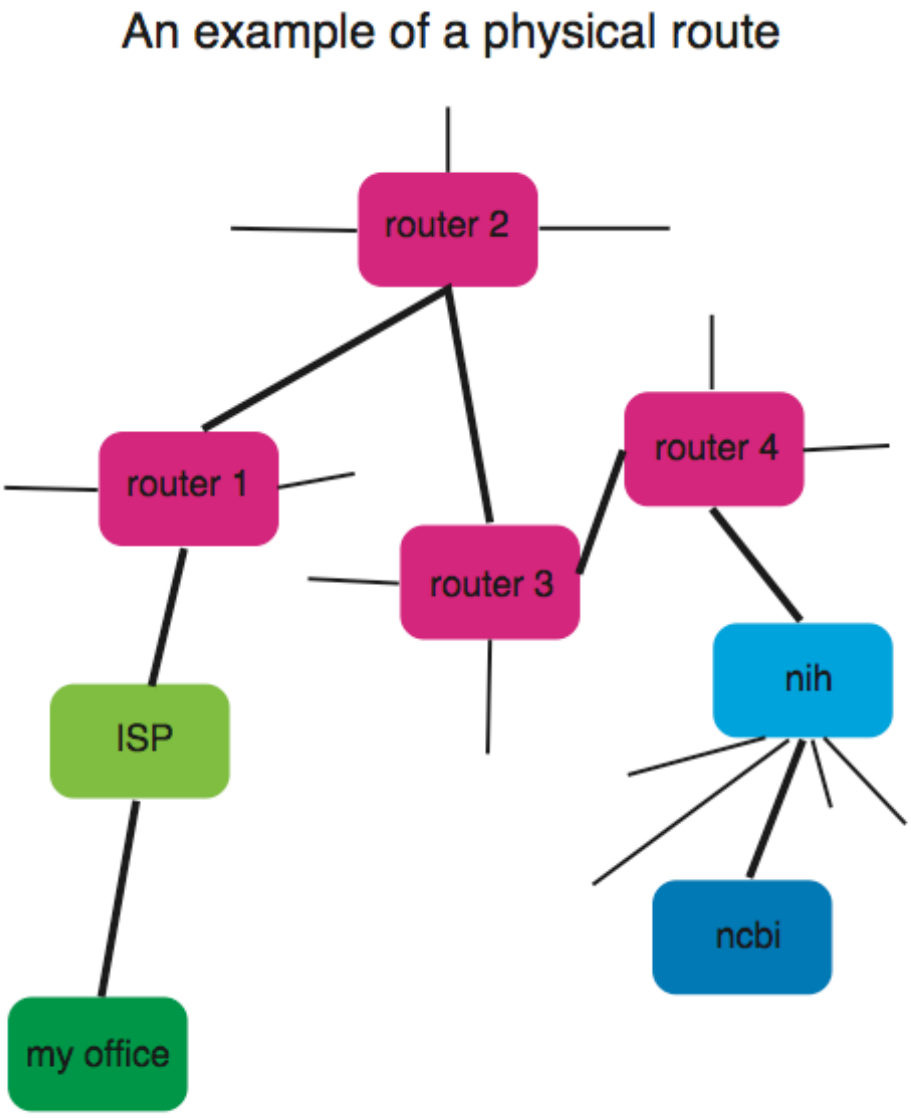
Up to now we have assumed an IP connection to your ISP. Using this protocol your machine is actually on the Internet when you are connected, and you can theoretically do anything that the big boys and girls do. However, you can instead dial up a company, e.g. America On Line, using special client software to communicate with their machine. Now you are not a node on the Internet, but rather a user of their computer system. You can send e-mail, but you will be using their e-mail software. Your can access web sites, but you will be using their web browser. You have given up the independence of directly exchanging data on the Internet for freedom from the need to provide the software to do it. Often the company will have special software that allows users to send messages to each other, e.g. chat rooms. In fact, the dial up computer industry started as a collection of "Electronic Bulletin Boards" which allowed users to send messages to each other.

Routing IP packets

The details of message routing depend on the structure of the specific networks, but here we describe the process in general terms. The fundamental principal is that each router only needs to determine which router or computer, of the ones it is directly connected to, an incoming packet should be sent to. Thus, the path a packet takes from

source to destination is the result of many routers sequentially making local decisions; there is no grand plan.

Figure router



router

Figure router. A packet is sent from a computer in my office to my ISP and then to the first router. Router 1 has in memory a table that defines which IP addresses are to be sent to the routers to which it is connected. The table indicates that my packets should be sent to router 2, and there the process is repeated. The packets finally reach a router that has the actual destination computer in its routing table, and the packets are sent there.

After the router has received a message packet on one port, it must decide which of its other ports to send it out on. This decision is made using its routing table, which is a list of nets, or groups of nets, that can be accessed by each port. If the router carries only a limited amount of traffic the routing table may be fixed, and only updated infrequently by direct human intervention. It is more common to use dynamic routing, in which the routing table is frequently updated by a routing protocol. The routing protocol uses information sent to it by other routers that report that they have been able to relay messages to other groups of nets. The routing protocol can also use information relayed to it concerning the fate of messages it sent out to other routers. As you can guess, the construction of efficient routing protocols is a challenging task.

The traffic on nets changes from minute to minute and the average traffic level tends to be high at certain times of the week and day and low at night and on the weekend. A router might be connected to several nets and other routers, and there are going to be times when a burst of traffic occurs on several of these nets at the same time. Routers have memory, and thus can store a considerable number of packets that arrive almost simultaneously, but what happens if a router is overwhelmed? The dirty little secret we reveal now about the IP level is that there is no guarantee that a message will be transmitted to the intended destination. The IP level only provides a best effort. This confession suggests what a router does if it receives more data packets than it can possibly handle; they are trashed.

More work for the router

We have focused on the address fields first, since the primary function of the IP layer is to provide addressing. However, the router must do much more than just read the address and forward the packet. The first field of the header specifies the IP protocol version, which must be known in order to decode the rest of the header. Most messages now use version 4, although version 6, with 128 bit address fields to provide a greatly expanded number of addresses is being slowly phased in. Reading the second field gives the header length, which enables the check sum to be calculated. If the actual checksum does not equal the number read from the checksum field, the header must be corrupted and there is no use in proceeding; the whole packet is discarded. Note that the IP layer plays no role in detecting or correction errors in the message it is carrying. The Time To Live field is read, and if it is zero the packet is discarded because it is assumed that the packet is in a pathological loop. If not zero the value is decremented by one, and written back in the new header.

The IP layer allows variable length packets of up to a 65,535 byte maximum. Each IP packet has a great deal of autonomy, which helps make communication on the Internet very robust. However, this very autonomy generates a high overhead for each packet the router handles. If the underlying net can support long packets, long packets result in high efficiency. However, many types of links can not carry long packets, and thus it must be broken into fragments which will be reassembled later. This is why the second word of the IP header is devoted to a flag to indicate that fragmentation has occurred, a packet ID number, and the fragment offset number which specifies where

in the original packet this fragment starts. The router must know the maximum packet length supported by the net it decides is the best route to the destination. If the packet is longer than this maximum, the router must fragment the message and set the fragmentation field so the packet can later be reconstructed. The router thus plays an essential part in permitting nets using different lower layer protocols to exchange messages.

The IP layer can support several higher level protocols, and to accomplish this the router must read and follow the Protocol field in the third word of the header. Finally, in the Options field the IP header can carry a request for quality of service, e.g. low cost, short delay, high reliability. If the router has collected information on the values of these parameters (routing metrics) for alternate routes to the destination, it can match a route with the request. Routing metrics are seldom used at present, but the protocol has provision for their implementation.

As we have seen, the router not only receives, stores, and (usually) retransmits packets. It also alters the packet header to decrement the "time to live" and may reformat the packet to send it out on the next link. If the packet has already passed through too many routers, or if this router is overloaded with other messages it discards the packet. A router is much more than a switch

Helping humans: the Domain Name System (DNS)

The Domain Name System was created to make Internet addresses easier for humans to use. It does this by allowing IP address space to be organized as a flexible hierarchical tree and using names in place of numbers to label the branches of the tree. The domain name of the destination host machine is converted into a numerical IP address by a resolver, which obtains the address from a system of Domain Name Servers (DNS).

A domain name embedded in an Universal Resource Locator (URL)

Let's jump ahead a little and look at a more powerful type of address: the Universal Resource Locator (URL). The URL is familiar to every Internet user and contains a domain name at its core. A good example URL, and one you might actually use to explore some of the biology covered in the second half of this book is:

http://www.ncbi.nlm.nih.gov/PubMed/

Let's peel off components from beginning to end of this URL to understand its structure.

http: Hyper Text Transfer Protocol (which is implemented using a web browser): the application protocol which will use the connection. The domain name server will make sure this protocol is supported on the destination machine and give you the complete IP address.

// The delimiter at the start of the domain field of the URL.

www. World Wide Web: the leaf of this domain name tree. This is a directory on a disk that contains a collection of files that constitute a World Wide Web site. A WWW site directory doesn't have to have this name, but as a convention most do. The . is a delimiter between the sub-domains of the complete domain name, and is pronounced "dot".

ncbi. National Center for Biotechnology Information: This could be a group of machines, a specific machine, a specific disk drive on a machine, or a directory on a drive. You don't know or care. However, it must contain only one directory named "www", or the URL wouldn't be specific enough to allow you to connect. It could contain other web sites, but they just can't be named "www". If this domain is omitted from the URL you will usually get the site of the next highest domain, which in this example is the National Library of Medicine.

nlm. National Library of Medicine: you can be sure this library has many computers, but exactly how they are linked together is again of no concern to you. Several other domains are accessed from the next higher domain, e.g. nci (National Cancer Institute).

nih. National Institutes of Health: the main campus containing many buildings is at Bethesda MD, but there are other buildings and agencies scattered over the United States. An alternative domain could be the Internal Revenue Service (irs).

gov Government: the top level domain (the final "." representing the root is suppressed). A domain name without an explicit country name is assumed to be in the US (the Internet began as a US Defense Department network). An alternative might be educational organizations (edu).

/ A delimiter at the end of the domain name.

PubMed The name of a file which links to an application. It appears as a page in the web browser and allows access to citations from several groups of medical publications via a database search engine.

We see in this example how the domain name system eliminates the need to handle or even see the numerical IP address, and, just as importantly, allows addresses to be grouped in a way that has meaning to the human user. The domain name system is organized as a hierarchical tree, and is often visualized with the root at the top and leaves at the bottom (thus a top level domain is at the root, not the top of the tree). The

part of the domain name tree that corresponds to the example URL is seen in Fig. dns_tree.

Figure dnsTree

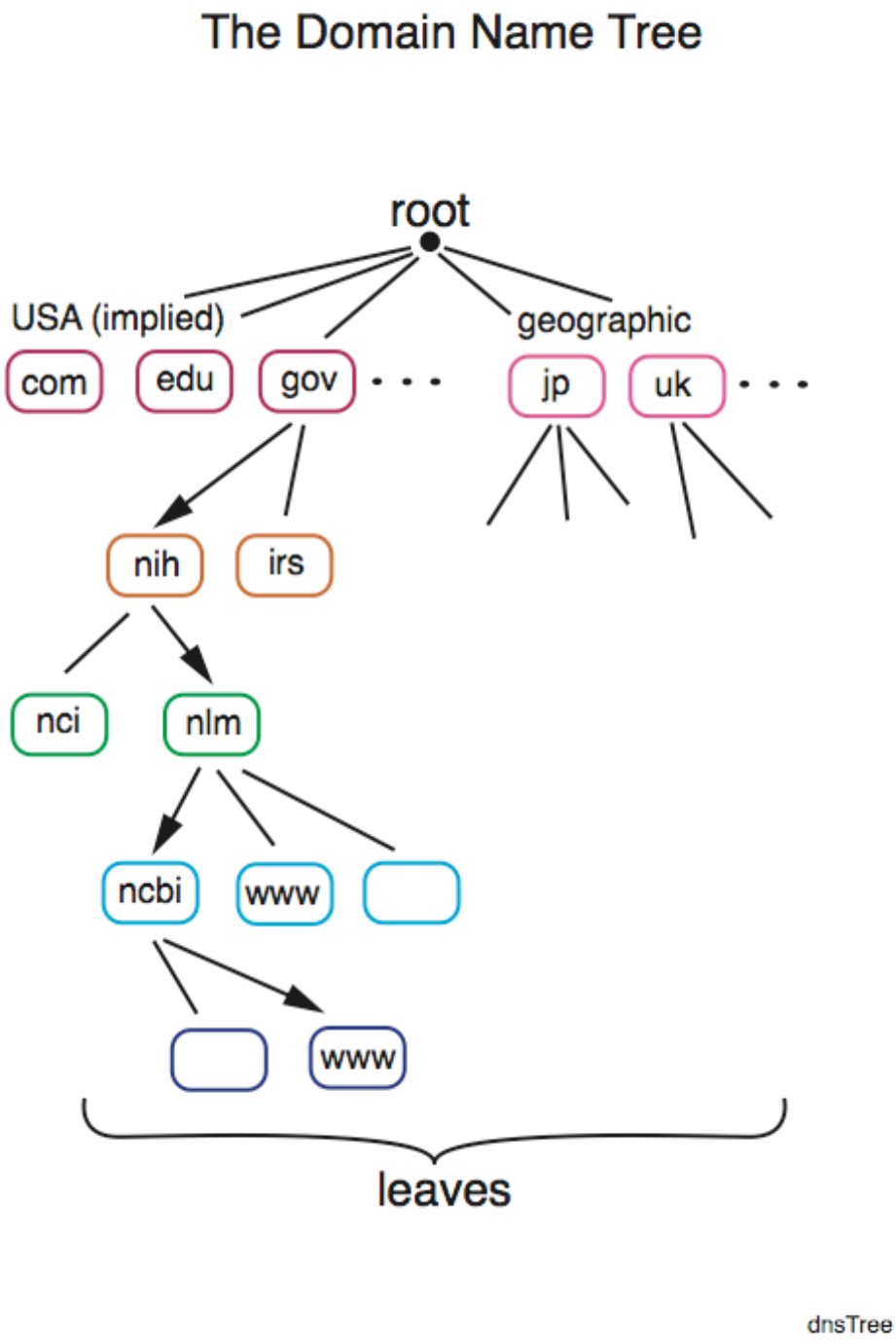


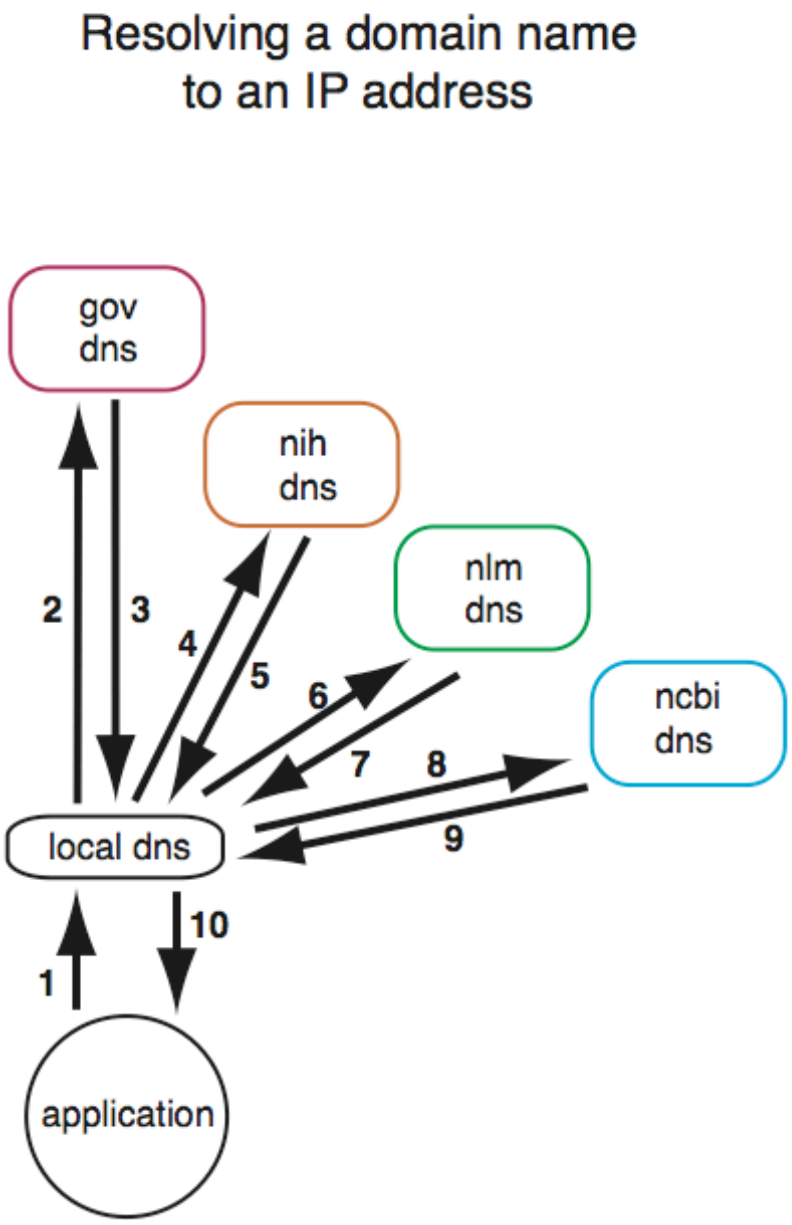
Figure dnsTree. The URL described in the previous table presented as a domain tree.

Top level domains with no geographical reference are usually in the United States: com (commercial), edu (educational, usually universities), gov (government), net (networks, typically ISPs), mil (military), org (non-profit organizations). However, sites in the United States can also use geographical top level domains, e.g. ca.us for California, United States. Domains outside the US usually have a geographical top domain, e.g. a commercial domain in the United Kingdom would use co.uk (**co**untry.**u**nited **k**ingdom).

Getting the IP address

The application that is being used to send and receive messages over the Internet is the program that is going use the IP address, and thus it initiates the search. In the above example, the application would be a web browser, and you would have typed in the URL into a text box and hit the return key. A resolver routine in the browser then sends a request for the IP address to the local Domain Name Server (DNS) on your machine. If you have sent a message to this domain recently, the IP address is likely to have been saved and will be returned quickly to the browser, otherwise the request is relayed to the DNS run by your ISP. Unless this DNS knows the IP address, the request will be sent to a root DNS, in this case the gov DNS. While this top level DNS may not know the IP address either, it certainly does know the IP address of the DNS that has authority for the entire nih domain. This certainty is the foundation of the domain name system: a DNS must know at least the IP addresses of all the DNS's responsible for the domains it is attached to. Thus the top level DNS gives the requesting DNS the IP of a DNS that is closer (more specific) to the desired domain. This starts a chain of requests and replies that is assured eventually to find a DNS that knows the IP address of www.ncbi.nlm.nih.gov. The IP address is then sent to your browser, and it can use it to send a message requesting the page PubMed.

Figure dnsResolve



dnsResolve

The local DNS sends requests to other servers, starting at the top level domain. Each server need only know the address of all DNS machines that are directly below it in the domain tree.

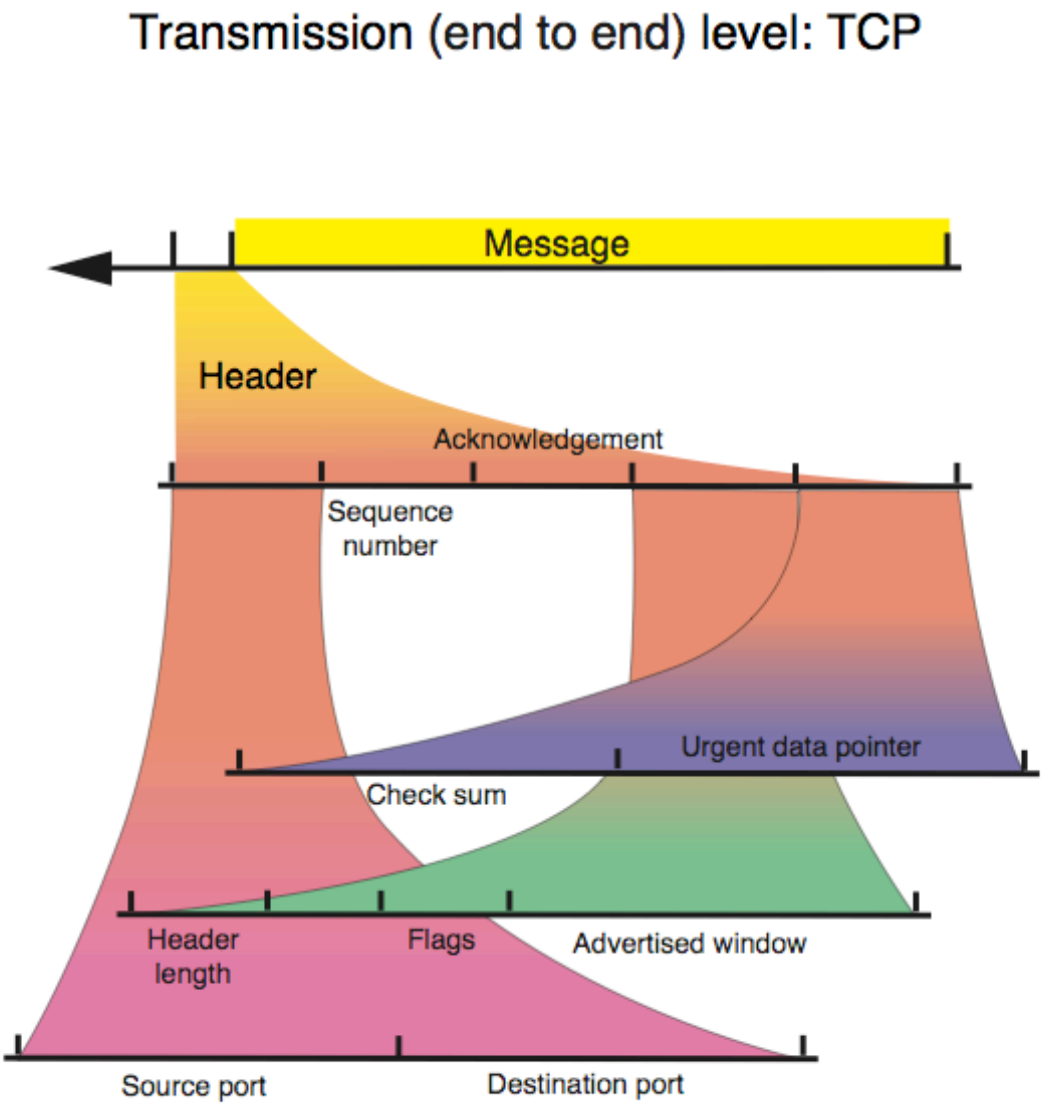
Finding the machine if only the IP address is known

You sometimes have the IP address and want to know the domain name. This can be accomplished by sending a request to the special, inverse address domain: IN_ADDR. The request is constructed from the IP address in inverse order, IN_ADDR, and finally the top level domain. Thus you find the domain name for the IP address, 130.14.22.107 by asking for the address of 107.22.14.130.IN_ADDR.gov (thus you have to already know it is in the gov domain).

The transmission layer: TCP as an example

The level above IP is called the transmission level, and TCP (Transmission Control Protocol) is often used on the Internet. Thus the combination, TCP/IP is sometimes used to define the Internet, although in fact other transmission protocols are also used. The transmission level is also called the end-to-end level because it is the level seen by the applications on the two host machines at the ends of the path as they exchange messages with each other during a session. The virtual link between the two host machines is assumed by TCP to have been established, so the address is replaced by an identifier called a "port", which enables simultaneous communication between different applications running on each host machine. A port is specified by a 16 bit integer, and thus there are 65,536 possible ports. Many of the lower port ID numbers are "well known", i.e. they are known by the Internet community to be associated with specific applications. HTTP sessions are usually conducted on port 80, and thus a web browser application would use this port by default.

Figure frameTCP



frame_tcp

Figure frameTCP. The IP layer has delivered the message, so the header is concerned with machine port numbers, and a sequence ID.

Applications are not concerned with the collection of bytes into packets and the necessary changes in the sizes of the packets as they are transmitted across the Internet. TCP informs the application that data has arrived after packets actually have arrived, any duplicate packets have been discarded, and the bytes been arranged in proper sequential order with no gaps due to undelivered packets. Thus, at the TCP level, data is available as a sequential stream of bytes. This allows data of arbitrary length to be transmitted; delimiters that group the data are the responsibility of the applications.

The greatest challenge for TCP is to ensure that all packets are delivered. Of course a mere protocol can't ensure this, but TCP at the receiver does inform TCP at the sender when specific packets have arrived. If the sender TCP does not receive an acknowledgement for a transmitted packet after a length of time it transmits the packet again. The transmission of the missing packet continues at intervals until an acknowledgement of receipt is received. That is the best TCP can do.

Figure tcpAction

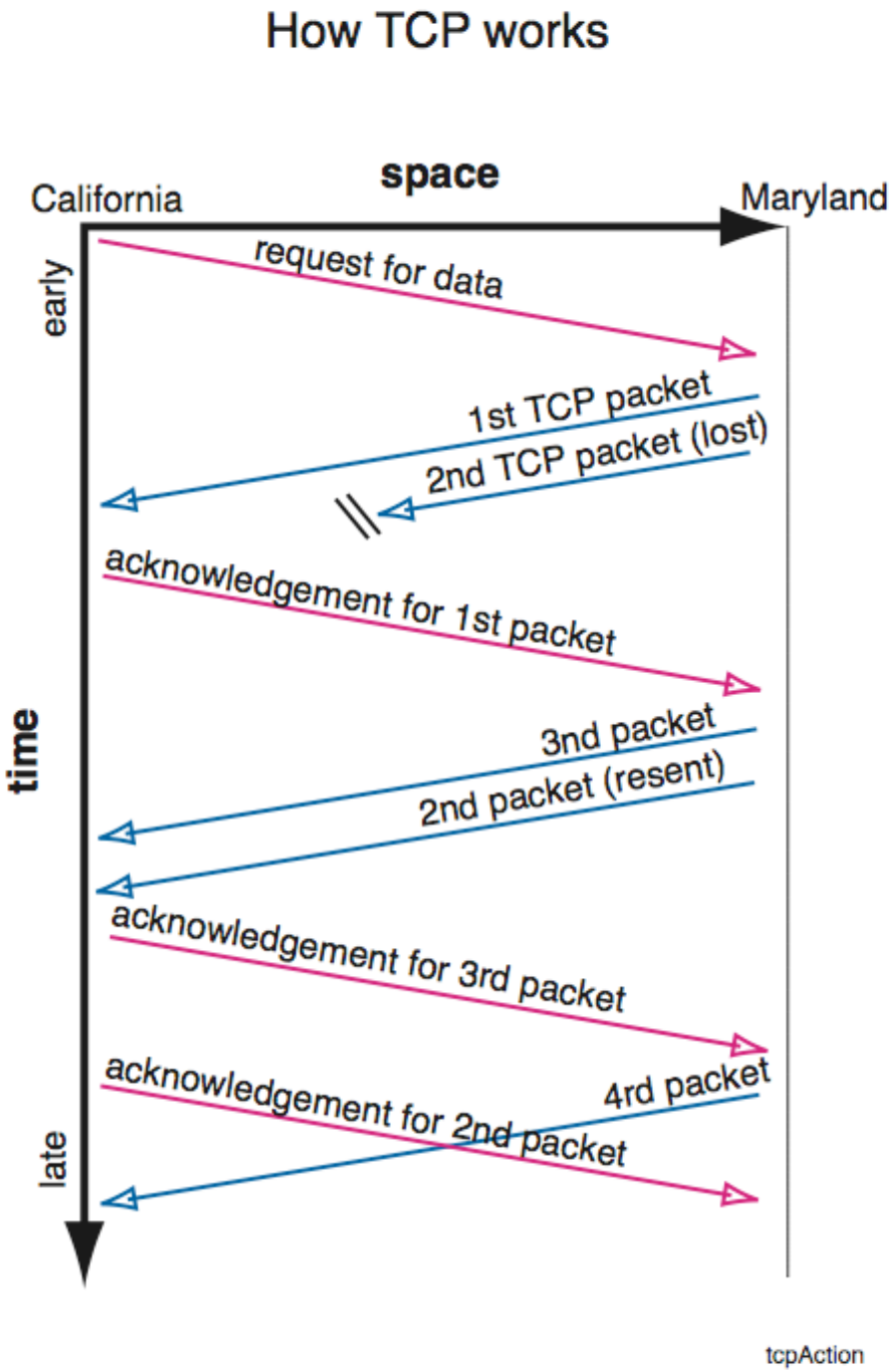


Figure tcpAction. In this example a client machine in California requests data from a machine in Maryland. As packets are received acknowledgements are sent back to Maryland. If a packet is not acknowledged in a set time, it is resent. Geographical space in indicated by the horizontal axis while time is indicated on the vertical axis.

Flow control

However, the TCP send-acknowledge protocol is not just a mechanism for replacing lost packets, it also allows the rate of data transmission to be adjusted to the capacity of the receiving host, a process known as flow control. When the TCP link is set up (we skipped over that step) the first acknowledgement sent by the receiver to the sender contains a field in the TCP header that states the number of bytes it can hold in its receiving buffer, the Flow Control Window. As data packets arrive they are processed by the application if there are no missing packets and the application is able to handle the data. If the buffer at the receiver begins to fill up, the acknowledgements sent back to the sender contain a smaller and smaller value for the Flow Control Window. This causes the sending TCP to send shorter packets.

In the limit the receiving TCP may not be able to receive any additional data, and it returns a zero for the Flow Control Window. However, the only way the sender can communicate with the receiver is to send data and receive or fail to receive an acknowledgement. Thus the sender periodically sends a packet containing one byte of data, and when it receives an acknowledgement it can start to again send packets with the advertised size.

Congestion control

Even if the sender has data and receiver has space for data, the Internet may be congested. Congestion means that at least one of the routers transmitting the data can't keep up with the traffic flow, and has to discard packets. While these packets don't get to the receiver, they do contribute to the congestion, because they fill links before hitting the congested router, and the congested router has to waste some time just trashing them. It is thus in the best interest for all senders that are loosing packets to slow transmission rates.

The sending TCP starts with a value for the Congestion Window, which is the maximum number of bytes it will ever leave on the Internet (packets it has sent but not received an acknowledgement for; once an acknowledgement is received the sender knows those packets are not on the Internet). Before each transmission it compares the Congestion Window with the current value of the Flow Control Window and takes the smaller of the two. It then subtracts the packets that are already on the Internet, and sends no more than that number.

If an acknowledgement for a packet is not received by a timeout period, the sending TCP assumes the packet has been dropped due to congestion (it might have been dropped because corrupted data was detected, but that is less likely). The sending TCP then decreases its Congestion Window by a factor of two, and soldiers on. Each time a packet acknowledgement times out, the Congestion Window is decreased by a factor of two until it reaches a value of one.

When a series of packets has been delivered successfully (as evidenced by the receipt of acknowledgements) the Congestion Window is increased. However, the

increase is linear, not proportional, i.e. it is increased by a constant value each time an acknowledgement is received.

Experiment and altruism

The Internet (and its precursor) had been running for eight years before congestion control was invented and implemented. The choice of an exponential decrease and a linear increase in transmission rate was chosen by trial and error, not by any theory. There are many variants of these congestion reduction algorithms as well as other techniques that are implemented by routers. The take home lesson to be made here is that the construction of networks is a combination of clever guesses and discovering what works and what does not work by experience. Building networks is an experimental science and an evolutionary process.

A remarkable feature of Congestion Control is that it is apparently self-imposed. The sender TCP decides to help reduce congestion, and if all sender TCPs do this the situation is improved for all. If only one sender TCP does not moderate transmission rate in response to congestion, the average level of congestion will still decrease, and the single rogue TCP will maintain a high rate of transmission at the expense of the others. However, while traffic on the Internet is not controlled by a central authority, it is monitored. Members that do not adhere to commonly accepted practices and cause a significant degradation of network flow are eventually detected and notified of deviation from these norms (but a low volume "cheat" could get away with selfish behavior). The Internet is a very loose and forgiving organization, but there are limits. It is easy to effectively remove an offending host or domain from the Internet by refusing to accept traffic to or from the offender.

Dependence on cooperation is a common situation, not at all unique to the Internet. As one example, there is always some small risk that a vaccination against a disease will have some undesirable effect. If only one individual refuses to be vaccinated, that individual will reap most of the benefits of vaccination, since no one in their community will acquire the disease and thus they are unlikely to be infected. However, the one that is not vaccinated will not share the risk of the unpleasant side effect of the vaccination. Of course if too many individuals refuse to be vaccinated, everyone will lose.

Applications

The function of all the protocol levels we have described up to now is to supply data to an application, but to make use of data you must have an application. The Internet is now used with an immense number of applications, but the following table lists a few core applications and the "well known ports" associated with them.

PORT	APPLICATION	FUNCTION
7	Echo	tests for a operational connection between two hosts
20, 21	FTP (File Transfer Program) port 21 used for commands, port 20 for transmission	transmits files between users

	of files	
23	Telnet	emulators a terminal: enables users to log in to another host as if they were local users
25	SMTP (Simple Mail Transfer Program)	sends mail between hosts
43	whois	obtains information about an IP address
80	HTTP (HyperText Transfer Protocol)	requests and sends hypertext documents, i.e. Web pages.
110	pop3 (Post Office Protocol version 3)	saves and delivers mail to users that log in sporadically, e.g. a typical home subscriber.
119	nntp (userNet News Transfer Protocol)	transmits news documents

We will pick just one application and follow part of the transaction between the two host computers to get a flavor for the process. Typically the computers exchanging information form an asymmetric pair; the client is the machine that wants something and initiates a request and the server is the machine that replies and supplies the data the client has requested. However, both machines are transmitting data to each other, and in some applications the distinction between client and server may be blurred.

The protocol we will examine is the Hyper Text Transfer Protocol (HTTP). The client application is called a web browser and the server application is call a web server. After a TCP connection has been established on port 80, the client will request the data that will be presented to the user as a web page. A typical conversation between client and server might be:

request by client:

```
GET /index.html HTTP/1.0
User-Agent: Mozilla/4.5 (Macintosh; I; PPC)
Accept: image/gif
Accept: image/jpeg
```

translation:

Transmit file: index.html (usually the home page of a site); I am using HTTP protocol v 1.0.
I am a Mozilla (Netscape) version 4.5 browser running on a Mac PPC computer
I understand an image file with gif format.
I understand an image file with jpeg format.

response by server:

```
HTTP/1.0 200 OK
Date: Fri, 16 Apr 1999 05:20:30 GMT
Server: Apache/2.0
Content-Type: text/html
Content-Length: 5551
```

```
Last-Modified: Mon, 12 Apr 1999 01:04:21 -0800
>>>  data ...
```

translation:

I will use HTTP version 1.0; everything is OK.
The present date and time in GMT is ...
I am an Apache version 2.0 web server.
I am sending text in html format.
The total length is 5551 bytes.
Text was last modified on (local date and time with offset from GMT)
the actual data follows:

The syntax for this exchange is of course specified by the HTTP protocol. A remarkable feature of the protocol is that it is in English (more or less), i.e. human readable text coded in ASCII. It would have been more "efficient", at least in terms of numbers of bits required, to use a binary code that was sufficient in length to distinguish between the alternatives. However, the advantage of plain text exceeded the price of longer length in the minds of the developers. The main advantage of text is that anyone who understands English can read and understand the protocol. This makes it far easier to find and correct software errors. This comment suggests that software error might be a significant problem in network and application development, and so it is.

Note that a considerable amount of information is exchanged between client and server, and of course each knows the other's IP address. Some of the information might be considered peripheral, but most of it can be useful. If the server knows the browser type and version, it can return documents that the browser can display properly. In the best of worlds all browsers would be able to process the same code, however, even if different venders agreed on a common format, older versions of browsers will not be able to handle new formats.

Hyper Text Markup Language (HTML)

The next step is to translate the hypertext into a format for the human to read. The web browser application performs this translation. Thus the web browser is client specific. It must know operating system calls needed to present the message in a form that is as close as possible to the format desired by the author of the document. The early web browsers had the functional goal of displaying text in a way that preserved the meaning of the material. The inventors of the World Wide Web assumed a very mixed audience of client machines that had greatly different abilities; monitor screen size, color, text fonts, etc. HTML was designed to emphasize function, and not appearance. Thus, the author could not know the exact size of a font, or when text would be broken and continued on the next line. The only promise was that the reader could see everything and the basic design intent of the presentation would be apparent. The design of HTML is frustrating to graphics professionals accustomed to complete control of an image, e.g. an advertisement on a magazine page. In order to gain control, most commercial web documents use HTML code that has the effect of forcing the browser to present data in a fixed format. If the user's display is the right size and the browser window is large enough the effect is achieved, otherwise the result can be ugly. Demanding control has it’s downside.

HTML example

A markup language is a method for indicating, in symbolic manner, the format that should be used for the actual presentation of text (or other data). The basic idea is that formatting commands are mixed in with the data but encoded in a manner that permits them to be differentiated from the data. The concept is obvious when you look the following example of an HTML document.

```
<HTML>
     <HEAD>
          < M E T A       H T T P - E Q U I V = " c o n t e n t - t y p e "
CONTENT="text/html;charset=iso-8859-1">
          <META NAME="generator" CONTENT="GoLive CyberStudio">
          <TITLE>Desoxynucleic Acid</TITLE>
     </HEAD>
     <BODY BGCOLOR="#FFFF99">
          <CENTER>
<P><B><FONT  FACE="Geneva"  SIZE="4">Desoxyribose  nucleic
acid</FONT><FONT FACE="Geneva" SIZE="5">(DNA)</FONT></B></P>
          </CENTER>
<P><FONT SIZE="4">The main thread describes the structure of
DNA in simple terms, avoiding chemistry and physics as much as
possible.  An <A HREF="Intro_2.html">introduction</A> tells why
I think DNA is important. If you already know, or don't want to
read what I think about it, move on ahead.</P>
          <img src=î../Example.gifî width="227" height="164">
</HTML>
```

Format commands are enclosed in arrow brackets, usually in pairs. The first command turns the format option on and the second, preceded by a slash, turns it off. Since formatting is coded explicitly, any actual format information in the HTML data is ignored, i.e. text that is broken into several lines is presented as a single line. The meaning of some of the commands used above are:

```
<HTML>              HTML  data follows
<HEAD>              not displayed or put in special box at top
of browser
<META...>           not displayed; for browser or programmer
<BODY BGCOLOR>      the color of the body background
<CENTER>            center the following text
<P>                 new paragraph follows
<B>                 bold the following text
<FONT FACE=...>     text font and size are ...
<A HREF=...>        anchor:  hypertext reference marker name...
<img src=...>       image source is ...
```

Most of the format commands are fairly obvious, and thus not very interesting. The exception is <A HREF="Intro_2.html">introduction</A> which defines a link to an anchor, a location in another document; this is what hypertext is all about. In this example the text "introduction" is displayed by the browser in a special format, usually underlined and in a contrasting color. When the user places the mouse over this text and clicks, the browser replaces the data in the browser window with data from the file Intro_2.html. Thus hypertext allows the user to jump to and from different parts of a document. Note that the browser has to send a new request to the URL the existing page came from in order to get the new file. However, the command could just as well have requested a jump to a page at a new URL.

The command <img src= tells the browser that an image should be displayer, and specifies the source file containing the image. Just as with the hypertext command the browser has to request the image file to be sent across the Internet. Since the browser can't know what images are going to be requested until it parses the entire HTML page, it may take some time to construct the document. To speed the process a little, the size of the image (in pixels) is indicated in the command, which allows the browser to insert an empty frame and then continue to construct the document as it waits for the actual image to be returned. This explains why web pages are sometimes first displayed containing empty rectangles, with the images appearing later.

Word processors also use a mark up language to produce formatted text. However, most modern word processors hide the mark up code from the user, and just present the text on the writer's screen as it formatted by the mark up code. Most applications used to write html documents now do the same, so you can produce html documents and know nothing about the underlying code.

Image formats: GIF and JPEG

The “gif” extension for the image file name in the previous example tells the browser that the image will be in the Graphical Interchange Format. The GIF format works best on images with sharp edges and large patches of constant color, e.g. the diagram at the bottom of Fig html. The first step in constructing the GIF file is to catalog all the colors in the image. If the image had every possible color that a high quality computer monitor could represent, it would require one byte for each of the three screen phosphor colors, red, green, and blue, for a total of 24 bits. However, most diagrams contain only a few colors say 16 for example, which requires only 4 bits to enumerate. Thus, for this example the routine constructing the GIF file first generates a look up table to translate image specific color indexes 0 to 15 into real 24 bit screen colors.

The compression routine then scans along sequential lines across the image. We are assuming there are large patches of the same color, and thus these lines will often pass through the same color at many sequential points. Instead of just repeating the same color index over and over again, the routine uses a variation of the LZ (Lempel-Zev) compression algorithm: a string of points with constant color is represented as a single word. One scheme for constructing that word would be to give the color followed by the number of points with that color. There are no computer screens that are wider than 4000 pixels (that I am aware of) and thus12 bits will code for the widest possible patch of one color. The 4 bits for the color index and the 12 bits for the length of that

color segment on the scan line are a total of only 16 bits or 2 bytes. If you look at the GIF encoded diagram in the bottom panel of Fig html you will see that the great majority of horizontal scan lines contain only 1 to 8 colors, and thus only 2 to 16 bytes. For a simple image like the one in Fig html, a strong GIF compression scheme can also be loss-less, i.e. no information is lost in the compression.

The other common image compression algorithm used for web pages is the Joint Photographic Experts Group (JPEG) format. This algorithm was designed for images with colors that gradually merge into each other, thus containing few sharp edges, e.g. the human face. The JPEG encoding algorithm is quite complicated and very different from GIF. It is typically lossy, i.e. some details of the image are lost in the compression. Applications that produce JPEG files allow the user to specify the compression factor. Factors of from 10 to 30 are typically achieved with almost no apparent loss of quality to the human observer.

Layers: review and summary

In Figure layerChron we follow an imaginary text message as it is sent from my computer to a computer at the National Library of Medicine. This simplified example, illustrates the process of encapsulating the data in nested protocols. The data remains in the two top level layers, here IP and TCP, throughout the trip, but the lower network and physical layers are changed as the data moves through different intermediate networks on its way to the destination.

Figure layerChron

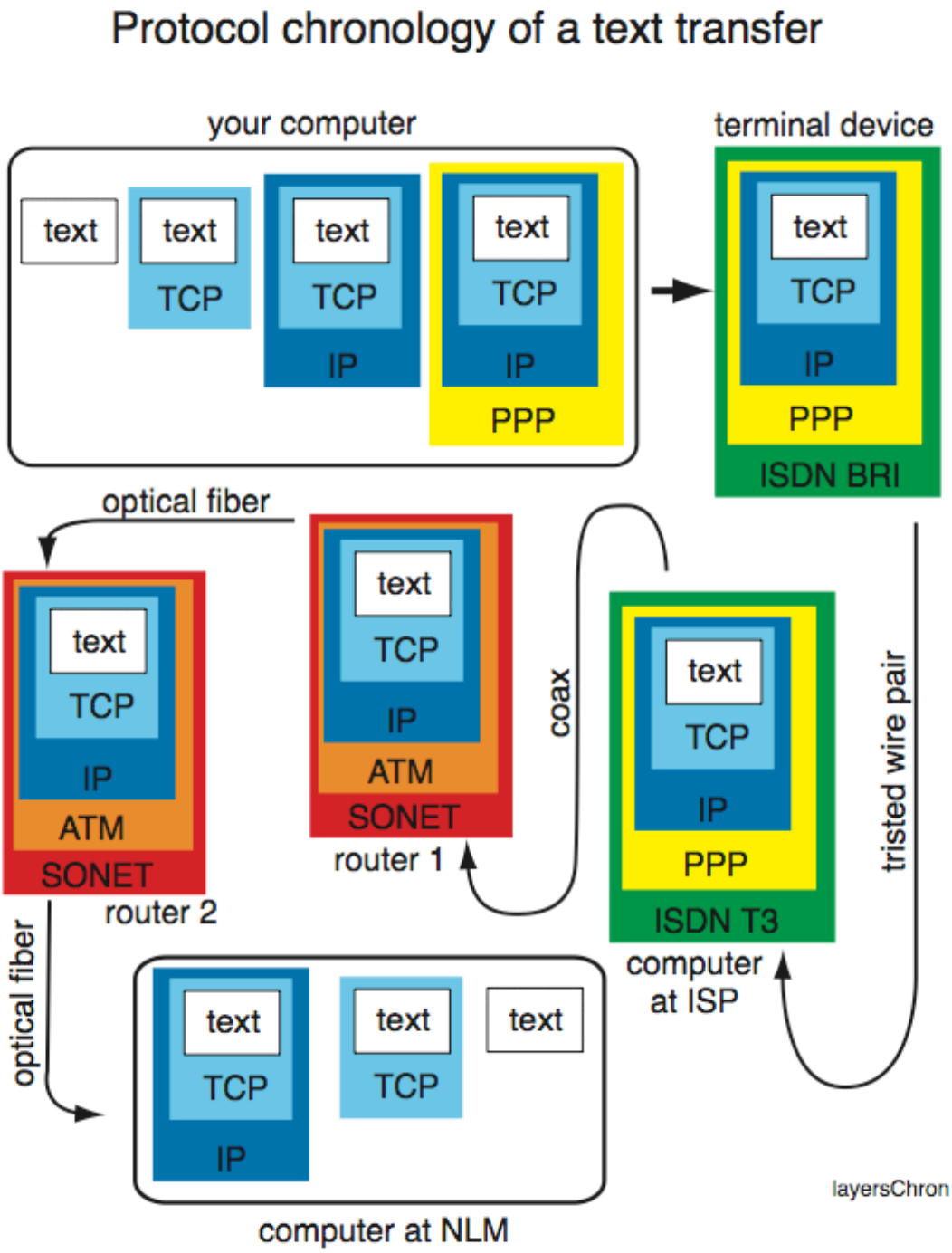


Figure layerChron. Text data is sent from my computer to a computer at the National Library of Medicine. The high level layers remain for the entire trip, but the lower layers are replaced as the message passes through different networks. The layer structure in the Figure represents the message as it leaves each computer (or router).

A real message from California to the National Library of Medicine would pass through many more links. As an illustration of this domains and IP addresses of routers that handled an actual message is listed below using data obtained by the application whatroute (see appendex). A few of the interesting routers are shown bolded.

```
 0 the router on my LAN                          (192.168.1.100)
 1 adsl-63-201-39-254.dsl.snfc21.pacbell.net     (63.201.39.254)
 2 core3-g2-0.snfc21.pbi.net                     (206.171.134.130)
 3 bb1-g1-0.snfc21.pbi.net                       (209.232.130.28)
 4 bb2-p12-0.snfc21.pbi.net                      (64.161.124.50)
 5 sl-gw28-stk-10-0.sprintlink.net               (144.232.229.9)
 6 sl-bb22-stk-8-1.sprintlink.net                (144.232.4.117)
 7 sl-bb21-sj-5-1.sprintlink.net                 (144.232.8.190)
 8 p3-1.snjpca1-cr6.bbnplanet.net                (4.24.238.141)
 9 p3-0.snjpca1-br2.bbnplanet.net                (4.24.9.153)
10 p2-0.lsanca2-br2.bbnplanet.net                (4.24.8.26)
11 p9-0.crtntx1-br2.bbnplanet.net                (4.24.5.62)
12 p15-0.crtntx1-br1.bbnplanet.net               (4.24.10.113)
13 p9-0.iplvin1-br2.bbnplanet.net                (4.24.10.214)
14 p15-0.iplvin1-br1.bbnplanet.net               (4.24.10.153)
15 p13-0.phlapa1-br1.bbnplanet.net               (4.24.10.181)
16 p15-0.phlapa1-br2.bbnplanet.net               (4.24.10.90)
17 so-0-0-0.washdc3-nbr2.bbnplanet.net           (4.24.10.185)
18 p6-0.washdc3-cr2.bbnplanet.net                (4.24.4.138)
19 p2-0.nlm2.bbnplanet.net                       (4.24.66.66)
20 NLM                                           (130.14.130.17)
```

Router 0 is on my desk; it's a small device costing about $100 which allows computers on my Local Area Network (in my study) to connect to one ISP on one telephone line. The other routers, which do the heavy lifting, are much larger and cost several $100,000 (more or less).

Router 1 is my ISP, Pacific Bell in San Francisco[2]. The packets are handed off to Sprint at router 5 and then to BBN at router 8. Router 15 is in Philadelphia while router 18 is in Washington DC where they reach the National Library of Medicine. If the same message was sent to the same location a few minutes later, it could easily take a different route, because the routing decisions are made on the fly at each router, and depend on the traffic and delays on the network. Thus if you were using a web browser to look at different pages on the NLM site, each page might take a different route. A long message (like a series of graphics that make up a Web page) is broken up

[2] Pacific Bell has since been bought out by SBC. Corporations seem to change names so often that I propose they should be known as aliases that would be semi-permanent. When a company is taken over the fact would be indicated by an entry in a table, presumably maintained by the SEC. In this way the customer would not be bothered by corporate shuffling, while investors could follow the purchase and sales by access of the table (on the Internet of course).

into packets, and even the packets can take different routes. This illustrates how different the Internet is from the classical telephone network, where all messages exchanged in one connection are carried by one physical connection.

The list of routers illustrates another characteristic of the Internet; packets travel from one commercial carrier to another as they move from source to destination. Thus, there has to be a mechanism to reimburse the different carriers in proportion to the traffic they carry. Conventional long distance telephone traffic also relies on multiple commercial carriers, especially after the Bell Telephone System monopoly was broken up into smaller corporations by the US Government. However, the conventional telephone system was built with a robust mechanism to allocate costs, since it requires establishing a dedicated circuit, which you then for pay by the second whatever is transmitted. As we have seen, the Internet is completely different, there is no connection, so you must pay by the traffic transmitted. It is a marvel (to this author) that the reimbursement system used by Internet carriers works as well as it does.

Figure levelSum

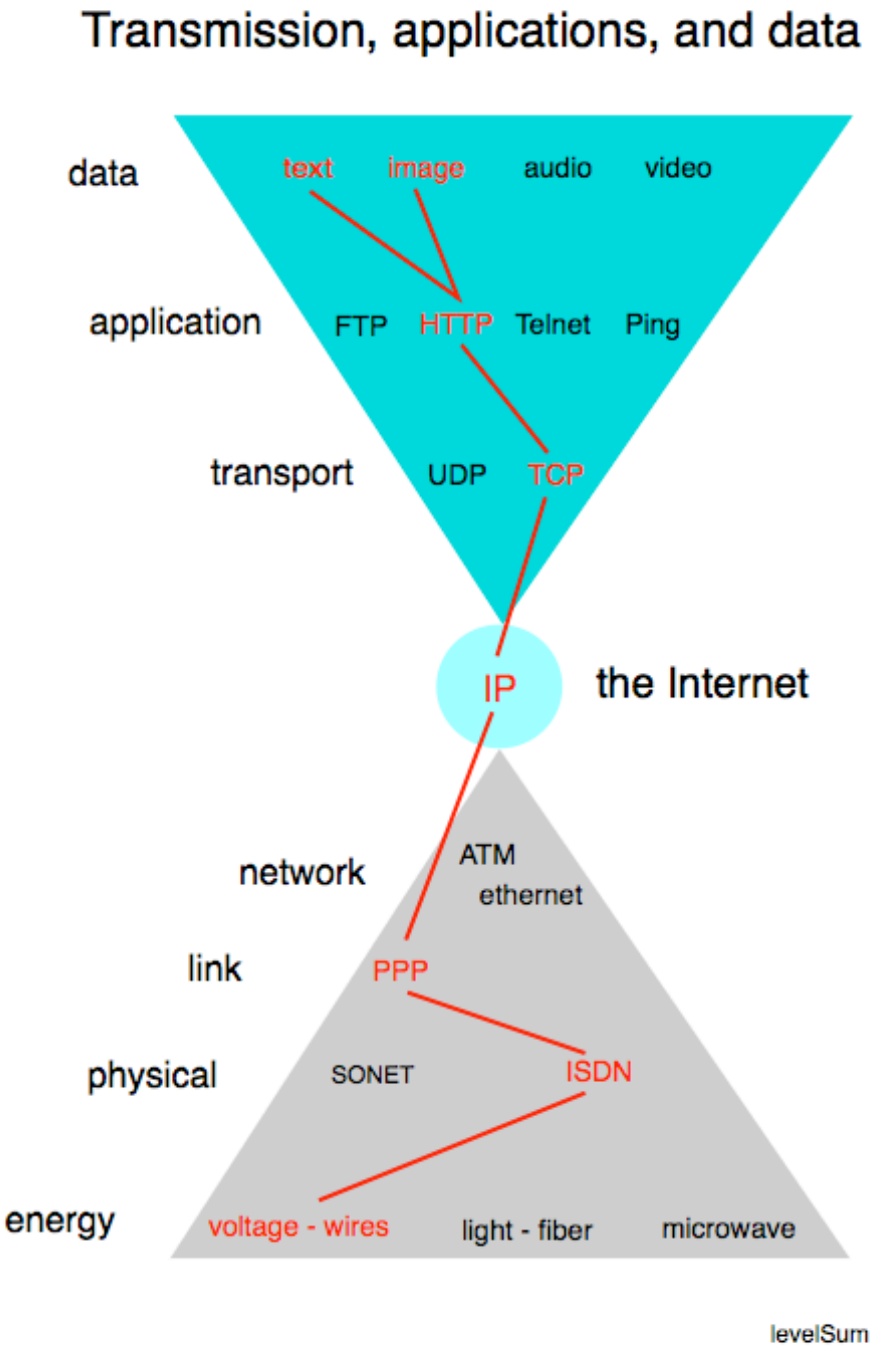


Figure levelSum. The layers involved when a web page is sent from an ISP to a computer using an ISDN telephone line. The alternatives are both physical (indicated on the lowest lines and labeled energy) and symbolic, i.e. formats and protocols.

Figure levelSum is a more symbolic diagram of the layers used by data moving over the Internet. The black line segments link transmission modes, protocols and formats that are involved when a web page is transmitted on an ISDN phone line. At each level a number of alternatives are indicated, but many more are possible.

Chapter summary

The topology of the Internet is complex and there is no central switchboard, but each message contains its address. The router at each node reads the address and forwards the message to another node based on suggestions from neighboring routers.

A message is often broken into smaller packets which are transmitted across the Internet, and reassembled into the original message. Transmission of small packets of data is more efficient than transmitting a mixture of small and very long messages.

Messages are contained in nested layers of protocols, with each layer having a specific function. The lower layers reflect the structures of the physical link and network, and are thus changed as a message travels through different types of systems. The Internet Protocol layer, which contains the IP address of sender and receiver, is the defining layer of the Internet.

The Domain Name System assigns a nested series of names to each numerical IP address. The domain names are easier for humans to remember and they reveal the tree structure of the address assignment.

The Internet does not guarantee delivery of a message. However, the TCP protocol layer attempts to maximize the probability of delivery by requiring an acknowledgement for receipt of each packet. If an acknowledgement is not received, the packet is retransmitted. The sender increases the rate at which packets are transmitted if they are received but decreases the transmission rate if they are not. This voluntary control is essential for the Internet to function well.

Data is sent and received by applications that use the protocol layers. Often the raw data has patterns, which mean that it can be compressed into a smaller data stream. Graphics, audio, and video data are usually compressed by coding.