An Insider's Guide to the Internet

David D. Clark

M.I.T. Computer Science and Artificial Intelligence Laboratory

Version 2.0 7/25/04

Almost everyone has heard of the Internet. We cruise the web, we watch the valuation of Internet companies on the stock market, and we read the pundits' predictions about what will happen next. But not many people actually understand what it is and how it works. Take away the hype, and the basic operation of the Internet is rather simple. Here, in a few pages, is an overview of how it works inside, and why it works the way it does.

Don't forget—the Internet is not the World Wide Web, or e-mail. The Internet is what is "underneath" them, and makes them all happen. This paper describes what the Internet itself is, and also tells what actually happens, for example, when you click on a link in a Web page.

1. Introduction to the Internet

The Internet is a communications facility designed to connect computers together so that they can exchange digital information. For this purpose, the Internet provides a basic communication service that conveys units of information, called *packets*, from a source computer attached to the Internet to one or more destination computers attached to the Internet. Additionally, the Internet provides supporting services such as the naming of the attached computers. A number of high-level services or *applications* have been designed and implemented making use of this basic communication service, including the World Wide Web, Internet e-mail, the Internet "newsgroups", distribution of audio and video information, and file transfer and "login" between distant computers. The design of the Internet is such that new high-level services can be designed and deployed in the future.

The Internet differs in important ways from the networks in other communications industries such as telephone, radio or television. In those industries, the communications infrastructure--wires, fibers, transmission towers and so on—has been put in place to serve a specific application. It may seem obvious that the telephone system was designed to carry telephone calls, but the Internet had no such clear purpose. To understand the role of the Internet, consider the personal computer, or PC. The PC was not designed for one application, such as word processing or spreadsheets, but is instead a general-purpose device, specialized to one use or another by the later addition of software. The Internet is a network designed to connect computers together, and shares this same design goal of generality. The Internet is a network designed to support a range of applications, depending on what software is loaded into the attached computers, and what use that software makes of the Internet. Many communication patterns are possible: between pairs of computers, from a server to many clients, or among a group of co-operating computers. The Internet is designed to support all these modes.

The Internet is not a specific communication "technology", such as fiber optics or radio. It makes use of these and other technologies in order to get packets from place to place. It was intentionally designed to allow as many technologies as possible to be exploited as part of the Internet, and to incorporate new technologies as they are invented. In the early days of the Internet, it was deployed using technologies (e.g. telephone circuits) originally designed and installed for other purposes. As the Internet has matured, we see the design of communication technologies such as Ethernet and 802.11 wireless that are tailored specifically to the needs of the Internet—they were designed from the ground up to carry packets.

2. Separation of function

If the Internet is not a specific communications technology, nor for a specific purpose, what is it? Technically, its core is a very simple and minimal specification that describes its basic communication model. Figure 1 provides a framework that is helpful in understanding how the Internet is defined. At the top of the figure, there is a wide range of applications. At the bottom is a wide range of technologies for wide area and local area communications. The design goal of the Internet was to allow this wide range of applications to take advantage of all these technologies.

The heart of the Internet is the definition of a very simple service model between the applications and the technologies. The designer of each application does not need to know the details of each technology, but only this basic communication service. The designer of each technology must support this service, but need not know about the individual applications. In this way, the details of the applications and the details of the technologies are separated, so that each can evolve independently.

2.1. The basic communication model of the Internet

The basic service model for packet delivery is very simple. It contains two parts: the *addresses* and the *delivery contract*. To implement addressing, the Internet has numbers that identify end points, similar to the telephone system, and the sender identifies the destination of a communication using these numbers. The delivery contract specifies what the sender can expect when it hands data over to the Internet for delivery. The original delivery contract of the Internet is that the Internet will do its best to deliver all the data given to it for carriage, but makes no commitment as to data rate, delivery delay, or loss rates. This service is called the *best effort* delivery model.

This very indefinite and non-committal delivery contract has both benefit and risk. The benefit is that almost any underlying technology can implement it. The risk of this vague contract is that applications cannot be successfully built on top of it. However, the demonstrated range of applications that have been deployed over the Internet suggests that it is adequate in practice. As is discussed below, this simple service model does have limits, and it is being extended to deal with new objectives such as real time delivery of audio and video.

2.2. Layering, not integration.

The design approach of the Internet is a common one in Computer Science: provide a simplified view of complex technology by hiding that technology underneath an *interface* that provides an abstraction of the underlying technology. This approach is often called layering. In contrast, networks such as the telephone system are more *integrated*. In the telephone system, designers of the low level technology, knowing that the purpose is to carry telephone calls, make decisions that optimize that goal in all parts of the system. The Internet is not optimized to any one application; rather the goal is generality, flexibility and evolvability. Innovation can occur at the technology level independent of innovation at the application level, and this is one of the means to insure that the Internet can evolve rapidly enough to keep pace with the rate of innovation in the computer industry.

2.3. Protocols

The word *protocol* is used to refer to the conventions and standards that define how each layer of the Internet operates. The Internet layer discussed above is specified in a document that defines the format of the packet headers, the control messages that can be sent, and so on. This set of definitions is called the Internet Protocol, or IP.

Different bodies have created the protocols that specify the different parts of the Internet. The Internet Engineering Task Force, an open working group that has grown up along with the Internet, created the Internet Protocol and the other protocols that define the basic communication service of the Internet. This group also developed the protocols for early applications such as e-mail. Some protocols are defined by academic and industry consortia; for example the protocols that specify the World Wide Web are mostly developed by the World Wide Web Consortium (the W3C) hosted at the Computer Science and Artificial Intelligence laboratory at MIT. These protocols, once developed, are then used as the basis of products that are sold to the various entities involved in the deployment and operation of the Internet.

3. Forwarding data—the Internet layer

3.1. The packet model

Data carried across the Internet is organized into *packets,* which are independent units of data, no more than some specified length (1000 to 2000 bytes is typical), complete with delivery information attached. An application program on a computer that needs to deliver data to another computer invokes software that breaks that data into some number of packets and transmits these packets one at a time into the Internet. (The most common version of the software that does this is called Transmission Control Protocol, or TCP; it is discussed below.)

The Internet consists of a series of communication links connected by relay points called *routers*. Figure 2 illustrates this conceptual representation. As figure 3 illustrates, the communication links that connect routers in the Internet can be of many sorts, as emphasized by the hourglass. They all share the basic function that they can transport a packet from one router to another. At each router, the delivery information in the packet, called the *header*, is examined, and based on the destination address, a determination is made as to where to send the packet next. This processing and forwarding of packets is the basic communication service of the Internet.

Typically, a router is a computer, either general purpose or specially designed for this role, running software and hardware that implements the forwarding functions. A high-performance router used in the interior of the Internet may be a very expensive and sophisticated device, while a router used in a small business or at other points near the edge of the network may be a small unit costing less than a hundred dollars. Whatever the price and performance, all routers perform the same basic communication function of forwarding packets.

A reasonable analogy to this process is the handling of mail by the post office or a commercial package handler. Every piece of mail carries a destination address, and proceeds in a series of hops using different technologies (e.g. truck, plane, or letter carrier). After each hop, the address is examined to determine the next hop to take. To emphasize this analogy, the delivery process in the Internet is called *datagram* delivery. While the post-office analogy is imperfect in a number of ways, it illustrates a number of other features of the Internet: the post office carries out other services to support the customer besides the simple transport of letters, and the transport of letter requires that they sometimes cross jurisdictional boundaries, in particular between countries.

3.2. Details of packet processing.

This section discusses in more detail the packet forwarding process introduced in the previous section.

The information relevant to packet forwarding by the router is contained in a part of the packet header called the *Internet header*. Each separate piece of the header is called a *field* of the header. The important fields in the Internet header are as follows:

Source address: the Internet address of the origin of the packet.
Destination address: the Internet address of the destination of the packet.
Length: the number of bytes in the packet.
Fragmentation information: in some cases, a packet must be broken into smaller packets to complete its progress across the Internet. Several fields are concerned with this function, which is not discussed here.
Header checksum: an error on the communications link might change the value of one of the bits in the packet, in particular in the Internet header itself. This could alter important information such as the destination address. To detect this, a mathematical computation is performed by the source of the packet to compute a *checksum*, which is a 16-bit value derived from all the other fields in the header. If any one of the bits in the header is modified, the checksum computation will yield a different value with high probability.
Hop count: (technically known as the "time to live" field.) In rare cases, a packet may not proceed directly towards the destination, but may get caught in a loop, where it could travel repeatedly among a series of

routers. To detect this situation, the packet carries an integer, which is decremented at each router. If this value is decremented to zero, the packet is discarded.

Processing in the router

The processing of the packet by each router along the route from source to destination proceeds as follows, each step closely related to the fields discussed above.

1) The packet is received by the router from one of the attached communications links, and stored in the memory of the router until it can be processed. When it is this packet's turn to be processed, the router proceeds as follows.

2) The router performs the checksum computation, and compares the resulting value with the value placed in the packet by the source. If the two values do not match, the router assumes that some bits in the Internet header of the packet have been damaged, and the packet is discarded. If the checksum is correct, the router proceeds as follows.

3) The router reads the hop count in the packet, and subtracts one from it. If this leads to a result of zero, the packet is discarded. If not, this decremented value is put back in the packet, and the checksum is changed to reflect this altered value.

4) The router reads the destination address from the packet, and consults a table (the *forwarding table*) to determine on which of the communications links attached to the router the packet should next be sent. The router places the packet on the transmission queue for that link.

5) When the packet reaches the head of the transmission queue, the router transmits the packet across the associated communications link, towards either a next router, or towards the computer that is the final destination of the packet.

Processing in the source and destination computers

The source and destination computers are also concerned with the fields in the Internet header of the packet, but the operations are a little different.

The source computer creates the Internet header in the packet, filling in all the fields with the necessary values. The source must have determined the correct destination address to put in the packet (see the discussion on the Domain Name System, below), and, using rules that have been specified, must select a suitable hop count to put in the packet.

The destination computer verifies the values in the header, including the checksum and the source address. It then makes use of an additional field in the Internet header that is not relevant when the router forwards the packet: the next-level protocol field.

As discussed above, packets carried across the Internet can be used for a number of purposes, and depending on the intended use, one or another intermediate level protocol will be used to further process the packet. The most common protocol is Transmission Control Protocol, or TCP, discussed below; other examples include User Datagram Protocol, or UDP, and Real Time Protocol, or RTP. Depending on which protocol is being used, the packet must be handed off to one or another piece of software in the destination computer, and the next-level protocol field in the Internet header is used to specify which such software is to be used.

Internet control messages

When some abnormal situation arises, a router along a path from a sender to a receiver may send a packet with a control message back to the original sender of the packet. This can happen when the hop count goes to zero and the packet is discarded, and in certain other circumstances when an error occurs and a packet is lost. It is not the case that every lost packet generates a control message--the sender is supposed to use an error recovery mechanism such as the one in TCP, discussed below, to deal with lost packets.

3.3. Packet headers and layers.

The Internet header is not the only sort of header information in the packet. The information in the packet header is organized into several parts, which correspond to the layers, or protocols, in the Internet design. First comes information that is used by the low-level technology connecting the routers together. The format of this will differ depending on what the technology is: local area network, telephone trunk, satellite link and so on. Next in the packet is the information at the Internet layer we have just discussed. Next comes information related to higher protocol levels in the overall design, as discussed below, and finally the data of interest to the application.

4. TCP -- intermediate level services in the end-node

The delivery contract of the Internet is very simple: the *best effort* service tries its best to deliver all the packets given it by the sender, but makes no guarantees—it may lose packets, duplicate them, deliver them out of order, and delay them unpredictably. Many applications find this service difficult to deal with, because there are so many kinds of errors to detect and correct. For this reason, the Internet protocols include a transport service that runs “on top of” the basic Internet service, a service that tries to detect and correct all these errors, and give the application a much simpler model of network behavior. This transport service is called Transmission Control Protocol, or TCP. TCP offers a service to the application in which a series of bytes given to the TCP at the sending end-node emerge from the TCP software at the receiving end-node in order, exactly once. This service is called a *virtual circuit* service. The TCP takes the responsibility of breaking the series of bytes into packets, numbering the packets to detect losses and reorderings, retransmitting lost packets until they eventually get through, and delivering the bytes in order to the application. This service is often much easier to utilize than the basic Internet communication service.

4.1. Detailed operation of TCP

TCP is a rather more complex protocol than IP. This discussion describes the important functions, but of necessity omits some of the details. Normally, a full chapter or more of a textbook is required to discuss all of TCP.

When TCP is in use, the packet carries a *TCP header*, which has information relevant to the functions of TCP. The TCP header follows the Internet header in the packet, and the higher-level protocol field in the Internet header indicates that the next header in the packet is the TCP header. The fields in the header are discussed in the context of the related function.

Loss detection and recovery: Packets may be lost inside the network, because the routing computation has temporarily failed and the packet has been delivered to the wrong destination or routed aimlessly until the hop count is decremented to zero, or because the header has been damaged due to bit errors on a communication link, or because a processing or transmission queue in a router is full, and there is no room to hold the packet within one of the routers. TCP must detect that a packet is lost, and correct this failure. It does so as follows.

Conceptually each byte transmitted is assigned a *sequence number* that identifies it. In practice, since a packet can carry a number of bytes, only the sequence number of the first byte is explicitly carried in the sequence number field of the TCP header. When each packet is received by the destination end node, the TCP software looks at the sequence number, and computes whether the bytes in this packet are the next in order to be delivered. If so, they are passed on. If not the packet is either held for later use, or discarded, at the discretion of the TCP software.

The TCP at the destination sends a message back to the TCP at the origin, indicating the highest sequence number that has been received in order. This information is carried in the *acknowledgement* field in the

TCP header in a packet being transmitted back from destination of the data towards the source. If the source does not receive the acknowledgment in reasonable time, it transmits the data again, and this repeats until either some copy of the packet finally makes it to the destination, or the application making use of the TCP terminates the activity and reports an unrecoverable error.

Flow and congestion control: The term *flow control* describes the mechanism that attempts to insure that the sender of data does not transmit faster then the destination can receive. The term *congestion control* describes the mechanism that attempts to insure that the sender of data does not transmit faster than the routers in the network can process and transmit the packets. A router that receives packets faster than it can transmit them along the next communication link must hold those packets temporarily. A router that is holding a number of packets for transmission is called *congested*. Congestion control is a critical aspect of the Internet; since any attached end-node can in principle transmit at will, it is possible for more packets to arrive at a router than can be carried across the outgoing communications link. Both flow control and congestion control are implemented in TCP.

Packets flowing back from the destination of the data to the source carry, in addition to the acknowledgment field, the *flow control* field. This field conveys a count of the number of bytes that the sender can send that have not been acknowledged. In other words, at any instant, there are some number of bytes that the sender has transmitted to the destination, but for which the acknowledgment has not yet arrived back. The sender must limit the number of such bytes to the value noted in the flow control field. The assumption behind this mechanism is that the receiver will allocate a holding area, or *buffer,* large enough to contain this many bytes, and if the receiver falls behind in processing the incoming packets, they can sit in this buffer. If the sender exceeds the flow control limit, the extra packets will usually just be discarded at the receiver.

The implementation of congestion control is rather more complex. Apart from the flow control limit passed back from the receiver, the sender maintains another estimate of the suitable sending limit called the "congestion limit". The congestion limit is never allowed to grow larger than the flow control limit from the receiver, but is often smaller. When the sending TCP starts to transmit packets to the receiving TCP, it makes an initial guess as to a suitable congestion limit. The initial guess is small: only one or two packets. It sends only this many packets into the Internet, and then waits for an acknowledgement packet to return from the TCP at the receiver. As long as packets are successfully acknowledged, the congestion limit is adjusted upward, at first rapidly and then more slowly. If an acknowledgment fails to arrive, the sending TCP assumes that some packet was lost because a router along the path was sufficiently congested that it had no further space in its transmission queue, and had to discard it. The sending TCP retransmits the packet, as was discussed above under loss detection and recovery, but additionally adjusts its congestion limit. Depending on the details of the situation, the sending TCP will cut the congestion limit in half or, more drastically, cut it to the small limit it used as its initial guess. Reducing the limit has the approximate result of cutting the average sending rate similarly, so the final consequence of this whole mechanism is that the rate at which the sending TCP transmits packets moves up and down in an attempt to find a rate that does not congest any router.

Error detection: a transmission error on a data link can damage a bit in a packet. The Internet protocol specifies a checksum function that is used to detect if a bit in the Internet header is altered, but this only applies to that header. TCP employs a similar checksum function to validate the TCP header as well as all the data bytes that follow it in the packet. The sending TCP computes the checksum and stores it in the packet; the receiving TCP recomputes it and compares this value with the one in the packet, discarding the packet if the two are different. A packet thus discarded is later retransmitted because of the loss detection and recovery mechanism discussed above.

Reliable open and close: The Internet service was described as a datagram service. At that level, the sender transmits a packet to a destination address, and need not know whether that destination is prepared to receive it, or indeed whether it even exists. Most high-level services need to know that the receiver is there and functioning. For this reason, before data is exchanged, the TCP software at the two ends of the communication exchange a sequence of packets with each other (containing no data but just TCP headers with specific values in certain fields) to insure each end that the other is there. This is called "making a connection". Similarly, when each end is done sending data, the TCP notifies the TCP at the other end that this is so. When both ends are done sending data, and all such data has been acknowledged back to its sender, then the TCPs "close the connection" by a final exchange of packets and a delay to insure that all is well.

TCP can carry data in two directions at once. Each end can send and receive at the same time, with separate flow and congestion limits in each direction. Packets carrying data in one direction can at the same time acknowledge data flowing in the other direction.

Overall, the resulting behavior of TCP is as follows. When some software at the sending end gives TCP a sequence of bytes to transfer to the receiver, the sending TCP opens a connection and starts to break these bytes into packets, attaching a TCP header to each and then handing them on to the Internet software for formatting and forwarding as described in section 3. TCP continues to transmit these packets so long as acknowledgments arrive. Based on the pattern of acknowledgments and losses, it may retransmit packets, and send at a faster or slower rate. If the flow control limit from the receiver goes to zero, it will temporarily suspend transmission. It will continue this process so long as there is further data to send. Data may or may not flow in both directions, based on the nature of the application.

4.2. Other intermediate protocols

TCP is the most commonly used protocol to enhance the basic communication service of the Internet, but it is not the only one. In some cases, the high-level application works best if it is built directly on the basic IP communication service. In this case, a simple interface to that service is used, called the User Datagram Protocol, or UDP. For real time services that are concerned more with delivery within a delay bound rather than completely reliable delivery, an alternative to TCP has been developed and standardized, called Real Time Protocol, or RTP. RTP carries timing information in the header, but does not implement retransmission to deal with lost packets. RTP is now being deployed as part of some audio and video applications on the Internet.

5 Layers and modularity—more about the organization of the Internet

Figure 4 provides a different visualization of the layering in the Internet. It shows the different software modules that implement the different layers, and illustrates the difference between a router and an end-node. A router has software specific to all of the lower level communications technologies in use (in the jargon of the trade, "device drivers" or "link-level software"). Above these elements, it has the Internet forwarding software that looks at the Internet level header and decides how to send the packet onward. The end-node must also have some link-level software and some software at the Internet level, but here the function is not forwarding but origination and termination. Above that is found TCP software (in those cases where the application needs it), and then the application software itself.

5.1. The end to end arguments

In terms of division of responsibility, the router, which implements the relay point between two communication links, has a very different role than the computer or end-node attached to the Internet. In the Internet design, the router is only concerned with forwarding the packets along the next hop towards the destination. The end-node has a more complex set of responsibilities related to providing service to the application. In particular, the end-node provides additional services such as TCP that make it easier for the application (such as the World Wide Web) to make use of the basic packet transfer service of the Internet.

TCP is implemented in the end-nodes, but not in the packet forwarding software of the routers. The routers look only at the Internet information, such as the delivery addresses, when forwarding packets. Only the end-nodes look at the TCP information in the packets. This is consistent with the design goals of the Internet, and is a very important example of layered design. While TCP provides a very simple service that most high-level applications find easy to use, some applications cannot make use of TCP. TCP always delivers data in order, and insists on retransmitting lost packets until they get through. This can, on occasion, cause delays of several round trips while losses are recovered. For applications such as real time Internet telephony, these occasional delays disrupt the communication flow much more than a short "glitch" in the data stream due to a missing packet. So most but not all high-level services use TCP. If TCP were implemented in the routers, it would be much harder for the high-level service to bypass it and use some other sort of transport service. So the design principle of the Internet has been to push functions out of the network to the extent possible, and implement them only in the end-node. By doing so, the high-level service can easily modify them or replace them by adding new software to the end-node. This is another means by which the Internet can evolve rapidly. For a broad base of high-level services, installing new services can be done without any need to modify the routers.

The above example illustrates a set of general design principles called the *end to end arguments*. The design approach of the Internet moves functions out of the network and onto the end-nodes where possible, so that those functions can be modified or replaced without changing the routers. Any set of users who invent a new application can run that application by installing the code for it on their end-nodes. Were it necessary to modify routers, or other network elements not under the control of the individual user, change would not be as rapid, and would be under the control of the network operator, not the user.

This principle has many implications. One example is security. The Internet header is distinct from the TCP and higher level headers in the packet. Those parts of the packet only concerned with end-node functions can be encrypted before they are forwarded across the Internet without fear that this will disrupt the forwarding processing in the routers. Using end to end encryption, the end-node can thus take responsibility for insuring the integrity and privacy of its own data.

5.2. Routing and forwarding -- functions specific to routers

There are also functions that are implemented in the routers, but not the end-nodes. The router must make a decision as to how to forward each packet as it arrives. In order to do this, it must have *forwarding tables* that specify, for each destination address, what the preferred path is onward towards that point. The routers compute the best routes to all the addresses in the network, in order to construct this table. This requires that all the routers send messages to other routers describing what links in the Internet are currently in operation, and what routers these links connect. This results in a collective decision-making, the *routing computation,* to select the best overall routes. Routers perform this task in the background, at the same time that they forward packets. If a low-level communications link fails or a new one is installed, this routing computation will construct new routes as appropriate. This adaptation is part of making the Internet robust in the face of failure.

End-nodes do not participate in the routing computation. They know only the identity of the router or routers closest to them, and send the packets to this first router. This division of responsibility makes it possible to replace the routing computation (which has happened several times in the life of the Internet) without having to change the software in the end-nodes, an almost impossible task if it had to be done in a co-ordinated way for all the millions of end-nodes on the Internet.

6. The Domain Name System

There are a few Internet services on which the end-nodes depend, in addition to the basic packet forwarding performed by the routers. The most important of these is the Domain Name System.

Packets carry source and destination addresses, but users do not use addresses of this form when they invoke services. Applications tend to use names that are a little more "user-friendly", composed of characters rather than integers. For example, the mail system uses names like "ddc@lcs.mit.edu", where "ddc" is a user name, and "lcs.mit.edu" is (more or less) the name of the mail server for that user. Before mail can be sent, the software on the sending end-node must first translate the string "lcs.mit.edu" into an Internet address, so it can format the packets.

In order to do this, it first sends a few packets to a service called the Domain Name System, or DNS, asking that this translation be done. This step is normally hidden from the user, unless it fails and an obscure error message results. But every time a piece of e-mail is sent, or a browser fetches a Web page, a name in character format is first translated into an Internet address.

The DNS names are organized in a hierarchy. For example, the name “lcs.mit.edu” is organized as follows. Starting from the right, the “highest” part of the name is “edu”. That name is associated with educational institutions. There are other common “top level domain” names, or TLDs: the name “com” is associated with businesses, “gov” with the US government, and the standard two-letter country abbreviations can be used to identify countries. Within the “edu” domain, the name “mit” is associated with the Massachusetts Institute of Technology. Within “mit”, the name “lcs” is a machine at the Laboratory for Computer Science. (Some people believe the names would make more sense the other way around, with the “edu” first, but it is too late to switch now).

The mechanics of translating a DNS name into an Internet address is achieved using DNS name servers, which are simply computers located throughout the Internet that are programmed for this purpose. The servers are organized into a hierarchy that matches the hierarchy of the names. There are a set of servers called the “root servers” that are prepared to receive a request to translate the top level names. Given the name “edu”, they return the Internet address of a server that knows about names inside the “edu” domain. A query to that server with the name “mit” returns the Internet address of a server that knows names inside MIT. Next, a query to that server with the name “lcs” returns the Internet address of a machine inside LCS. In the case of LCS, that machine is prepared to receive mail for ddc@lcs.mit.edu, and the translation of a name into an address is done.

There is only one further step to make this all work right. All end-nodes, in order to translate a name to an address, must be able to find a root server to start the process. But this fact is easy to get. Any DNS server anywhere in the Internet knows the Internet address of a root server. So if the end-node can find any DNS server, it can then find the root, and start the search down the hierarchy. So how can an end-node find any DNS server? When a host is first configured to operate on the Internet, there are a few facts that have to be supplied. Either the user must type them in manually, or they are downloaded by a special trick into the host over its network interface before it starts sending real Internet packets. The host must know its own Internet address, the address of a nearby router (so that it can send packets out into the Internet) and the address of a nearby DNS server. Those are the facts that have to be supplied when a new machine is attached to the Internet for the first time. Normally, with these three pieces of information, the software on the end-node can send packets to figure out everything else that it needs to know.

7. The design of high level services

The Internet itself, as an entity built of links and routers, is concerned with the delivery of packets. Applications such as the World Wide Web exist at a higher level. While to the consumer applications may be viewed as a part of the Internet, technically they run "on top of" the basic communication service of the Internet, specifically on top of TCP.

7.1. The World Wide Web as an example

A case study of a World Wide Web interaction illustrates some of the complexity of the overall design.

A Web server (an end-node attached to the Internet) stores "Web pages", and makes them available for retrieval on request. The pages have names, called URLs (Universal Resource Locators). These names, which have a rather ugly form, have become quite familiar over the few years; an example would be "http://www.ana.lcs.mit.edu/papers/this-document.html". These names are advertised so that potential readers can discover them; they also form the basis of cross-references or " links" from one Web page to another; when a user positions the mouse over a link and "clicks" it, the matching URL is used to move to the associated page.

When a user wants to retrieve a web page, the software must first consult the Domain Name System (discussed above) to translate the string (for example) "www.ana.lcs.mit.edu" into an Internet address, which will be the Web server storing the page "papers/this-document.html”. The software at the user's computer (the so-called "browser") then opens a TCP connection to the server machine, and sends a retrieval request that contains, among other things, the string "papers/this-document.html". There is a protocol called HyperText Transfer Protocol, or HTTP, that provides the rules and format for messages requesting a Web page. The server, on receiving the HTTP request with the name "papers/this-document.html", finds the matching file on its disk, and passes that file through the software that implements HTTP and down to the software that implements TCP. The TCP software breaks the file into packets, as described above, and sends these packets across the Internet. These packets are then re-assembled at the receiving end-node, and the resulting file is handed to the browser software for processing.

This somewhat complex story illustrates the layered nature of the Internet design. The transfer of a Web page involves actions at several different layers at the same time:

IP: At the Internet level, packets are received and transmitted when a Web page is requested. At this layer, the only relevant factors are the Internet addresses in the packets.

TCP: The TCP software takes a unit of data (a file, a request for a Web page or whatever) and moves it across the Internet as a series of packets. It does not examine these bytes to determine their "meaning"; in fact, the bytes might be encrypted.

HTTP: HTTP makes use of TCP to move requests and replies. In contrast to TCP, it looks at the contents of requests, understands the format and "meaning" of these requests, and thus retrieves the correct Web page. It then transfers the Web page in question. However, it does not know the format or "meaning" of the page itself. The page could be a traditional Web page, an image, music, and so on. The HTTP standard does not define how pages are represented. The HTTP standard only specifies how the format of the page is indicated, so that HTTP can find the right interpreter for the page.

HTML: the most common representation of a Web page is HTML, which stands for HyperText Markup Language. A Web page, as stored on the server, is not the literal representation of what the user sees on the screen of the browser. Rather, a Web page is a series of instructions as to how that image on the screen is to be constructed. These instructions include text and images, information about relative location, size, and font, the specification of links to other pages, layout information about background, colors, and so on. Depending on the size (for example) of the screen of the browser, these instructions will cause the page to be presented in somewhat different ways. HTML is the "language" in which Web pages are specified.

All browsers include software that understands HTML, so that arriving Web pages can be interpreted and displayed on the screen. HTML is not the only page format. Images, for example, are encoded in a number of different ways, indicated by the name of the standard: GIF, JPEG etc. These are alternative formats that can be found inside a Web page. Any format is acceptable, so long as the browser includes software that knows how to interpret it.

Reprogramming the browser: There are means to modify the behavior of the browser as it is being used. One that has received a lot of publicity recently is JAVA. JAVA is a programming language that is used (among other purposes) as part of the Web. JAVA provides a means to transfer new software from a server to a browser that can interpret Web pages. JAVA eliminates the need for pre-agreement as to the format of a Web page. The creator of a page can invent a new format if it suits his need, and transfer the interpreter to the browser just before transferring the page itself. This innovation is intended to provide greater creative freedom to the designer of the Web page. It has the implication that HTTP cannot know, in all cases, the format and "meaning" of the bytes being transferred across the Internet, because the "meaning" may not be defined in any standards document, but only by the "behavior" of the matching JAVA code. If running the JAVA code has the consequence that the receiving bytes are converted and played through the audio system of the computer, then it is possible to presume that the "meaning" of the bytes was to be "sounds", but this is not knowable by any of the other layers of the software: HTTP, TCP and so on. This flexibility is part of what allows the Internet to evolve to support new sorts of applications.

7.2. Delivery modes: Unicast, multicast and streaming

The preceding discussion of Web retrieval illustrates one possible pattern of data delivery. The designers of the Web categorize this as "unicast pull" mode: "pull" because the user initiates the request for the page, or "pulls" it at the time it is desired, and "unicast" because the packets crossing the network go to only one recipient. There are other alternatives.

Multicast

The term "multicast" is used to describe a pattern of packet distribution in which several destinations receive copies of the packets coming from a source. Multicast is an Internet-level service, implemented in the routers. It is intended to be more efficient than unicast for a number of cases. One example is multi-way teleconferencing, where multicasting is used to implement the equivalent of a conference call. Another example is the delivery of audio and video to multiple recipients. What the radio and television industry would call "broadcast" fits into this model.

The Internet design community reserves the word "broadcast" for a circumstance in which a packet is delivered to *all* end-nodes within some scope. Broadcast is used across local area networks to transmit low level control information, but broadcast to all the millions of end-nodes on the Internet is not a useful concept. It would lead to uncontrolled overload, jamming of end-node, and collapse of the Internet. Wide-area replication of packets only makes sense if receivers have expressed interest in getting the packets. This concept of sending to a group of receivers who have requested the packets (a so-called "multicast group") is important enough that the term "multicast" was coined to capture it. Joining a multicast group is similar to tuning a radio to a particular station, except that on the Internet, an end-node can receive from multiple multicast groups at once.

Streaming

The basic communication service of the Internet today is the simple "best effort" service that makes no assurance about transmission rate or loss rates. For some emerging uses that involve the transmission of audio and video in real time, this service is insufficient. The term *real time* describes the circumstance in which data is being transferred across the network within a bounded delay, and "played back" or presented to the user as it is being transferred. Audio and video streams have a specific data-rate requirement, and unless the Internet can offer some assurance that it can sustain that rate from sender to receiver, the audio or video stream cannot be delivered. Using TCP, the sender would slow down if congestion were to occur, which would disrupt the real time delivery of the data. (Note that just because the data is audio or video, it does not necessarily follow that the data need be transmitted in "real time". The data can be transferred "in bulk" and stored on a disk at the receiver, to be "played back" from the disk once it has been received correctly. In this case, TCP could be used to move the data across the Internet.)

The forwarding of real time packets is sometimes called *streaming*. The Internet is now being augmented to support real time service as well as best effort service as part of the basic communication service in the router.

7.3. Enhancing the Web

The Web, as described, is strictly a transfer between two end-nodes: a server and a client. There are approaches to augment this simple model for a number of reasons.

One approach is the so-called "Web proxy". This is illustrated in figures 5 and 6. The concept here is that a Web user will redirect all his Web retrieval requests to a "Web proxy server", also called a "Web cache server". This server keeps a copy of any page recently retrieved by a customer, so that if a second customer asks for the same page, this request can be satisfied from the local copy on the cache. This improves the service seen by the customer, since the page returns (hopefully) sooner. It also helps the Internet Access provider (IAP) supporting the customer, because the wide area capacity that the IAP purchases from an ISP is not used up moving the same file more than once. In some cases, the IAP may interpose a Web proxy in the path from the user to the Internet, so that the user does not even know that his requests are being redirected to the proxy.

The consequence of a proxy server is that the IAP serving the consumer will intercept the request for the Web page and originate the transmission of the page back to the consumer, using the locally remembered copy. Many providers of Web pages want the queries to come to their own server, so they can insert advertising, gather information on the user, and so on. For this reason, many Web pages are marked as “non-cacheable”, which means that the proxy is not supposed to keep a copy, but to retrieve the page from the original source each time it is requested.

A related activity is the creation of servers that store long-term copies of popular Web sites as a service for the provider of the page. These servers are sometimes called "mirror" servers, since they provide an identical copy of what is stored on the master server. The intention here is similar: to reduce the delay of retrieving the page by having a copy of it located close to the consumer. In this case, the page is stored there because of a contract between the provider of the page and the operator of the Web storage server.

The Web cache and the Web mirror illustrate that there are multiple reasons to put Web servers at points "inside" the network -- it can be a case of performance improvement and a business opportunity, perhaps at the same time.

7.4. E-mail

The description of the Internet given above was of high level transfers between two end-nodes, with the Internet itself providing only simple packet forwarding. Not all applications work this way. An important example is electronic mail, or e-mail. Since many users are not connected full time, if mail was transferred in one step from origin to destination, the transfer could only be successful during those occasional periods when both parties just happened to be connected at the same time. To avoid this problem, almost all mail recipients make use of a server (called a mail server or a "POP server") to receive their mail and hold it until they connect. They then collect all their mail from their server. The concept is that the mail server is always attached and available, so anyone can send mail to it at any time, and the receiver can retrieve mail from it at any time. This eliminates the necessity for the sender and the receiver to be attached at the same time. Most mail is actually transferred in three steps, from sending end-node to the mail server serving that sender, then to the mail server serving the recipient, and then to the final end-node. This is illustrated in figure 7.

7.5. Different applications are different

Different high-level services have different designs. The pattern of mail distribution does not resemble the pattern of Web page retrieval. As new services are invented, there will be new patterns of distribution and storage. For example, some emerging proposals for the distribution of "TV" programming across the Internet contemplate a two-stage pattern in which the first part is non-real time reliable (e.g. TCP) pushing of the content out to "TV caches", and the second part is a real-time "pull" of the information based on a streaming protocol. The proposal is such that the cache can be located at any point in the system, from the ISP or IAP to the disk on an end-node of the user.

8. Why is the Internet the way it is?

The previous discussion is rather mechanical—it describes how the Internet works, but not *why* it works that way. The *why* is as important as the *how*. Here is a short analysis of a few of the important design features of the Internet.

8.1 The why of packets

Packets are not the only way to organize a network. The telephone system works differently—at the beginning of a telephone call some transport capacity is dedicated to that call, enough to carry the voice information, and these resources are dedicated for the duration of the call. This is called a *circuit* network, in contrast to a packet network.

One advantage of packets is that no transport capacity is used unless there is data to send. In a circuit network, capacity is reserved for the circuit even if the end points have nothing to send at the moment. Many computer applications are *bursty*: they send their data in bursts separated by quiet periods. For example, when a user is cruising the Web, the transfer of each page is a burst of data, and nothing is sent while the user looks at the page and decides what to do next. With packets, the bursts of data for lots of users can be interleaved, which leads to more efficient use of the transport capacity.

Since packets can accommodate a range of data communication patterns, they can better accommodate the new application with a different pattern. So packets are part of the objective of designing for change, and supporting the unknown application yet to come.

8.2 The why of end-to-end

We discussed above one of the central design tenets of the Internet, the *end to end argument.* This line of reasoning says that if some function (e.g. application-level processing) can be implemented at the edge of the network, in an end-node, rather than in the routers in the network, edge placement is preferable. The benefit of this approach is that the network itself remains simpler and not constrained to only certain applications. While this may mean that any specific application is less efficient, it improves the chance that a new application can be added to the network without needing to modify the routers. So again, this is a tradeoff in which the ability to evolve is preferred over the optimization of the applications of today.

8.3 The why of stateless design

In the Internet, there is no concept of a “call”. Unlike the telephone system, where there is a “setup phase” at the beginning of a call where resources are reserved for that call along the path from source to destination, there are no reservations in the Internet. The router forwards each packet as it gets it, but does not have any advance knowledge that the packet is coming, or any log that it came. We call this a *stateless* design, since in the language of Computer Science, such stored information is often called *state.* (For a system to be in on or another state, there must be some information to distinguish the various states. If there is no stored information, there are no distinct states.)

The advantage of the stateless design is, again, simplicity. If there was a setup phase before sending a bunch of packets, the router could establish some state information for the packets, perhaps precomputing the forwarding decision or reserving some resources. This might be more efficient in specific cases, but also more constraining and more complex. For example, if state describing a flow of packets is set up along a path from source to destination, and then some link in the path fails, the state information in some of the routers would have to be removed, and state would have to be installed in some new routers. This is actually quite hard to do in a reliable and quick way. The design of the Internet went for simplicity.

8.4 The why of loose specification.

The Internet makes few commitments to the user about the quality or consistency of service. The *best effort* service commitment is that the Internet will do the best it can, but there is no specification of how good that is. The network may lose packets, deliver them out of order, delay them, and so on if conditions require. Why is this a good idea? This approach puts an extra burden on the application designer, who must design the application to cope with these variations. But tolerating these variations has a benefit. A wide range of technologies can be put to use to carry Internet packets—fast and slow, reliable and lossy, and so on. If the Internet specified a more constrained and pre-determined level of service, then some of these technologies might be excluded all together, which would mean that there would be situations where the Internet was not available at all. The view of the designers is that “poor” service is better than no service at all, and that the application and the user (the human) should determine what is “too poor”, not a Internet-wide specification.

8.5 A preference for flexibility and change.

There is a common theme through much of the above discussion. The Internet designers, when making design choices, preferred to optimize for change, not to optimize any single application. The telephone network is optimized for telephone calls. The Internet is optimized to adapt to the application that might happen tomorrow.

9. Conclusions -- implications

9.1. Implications of the layered structure

Because of the layered structure of the Internet design, the different sorts of service providers cannot always see the parts of the information that is not relevant to them. The ISP cannot always see the higher level information in the packets (for example, it may be encrypted.) The higher-level service provider (a Web server, for example) cannot see the routing information in the routers, and cannot determine what the topology and capacity of the Internet is. These limitations have policy implications—for example it is difficult for the IAPs to control access to objectionable material over the Internet, or to meter the delivery of content for which fees must be paid.

9.2. Content can have multiple representations.

In a digital system, a piece of material of one "type", e.g. a piece of music, can be stored and transmitted in multiple representations. Different encodings can be used to achieve different fidelity, different tolerance for bit errors, and different transmission efficiency ("compressed" representations must be "expanded" before being used, so they require more processing power but less transmission capacity.)

9.3. Delivery modes are not tied to type of content.

The way in which material is encoded or represented in digital format is not linked in any way to the mode of transport across the Internet. A given file can be (at the Internet level) unicast or multicast, and can be (at the end-to-end transport level) delivered reliably by TCP or in real time.

9.4. Many controls cannot be implemented “in” the net

Higher level concerns such as limiting the access to objectionable material or preservation of intellectual property rights have often been implemented in the end-node. This follows from several considerations. First, the Internet level mechanisms, the routers, and thus the ISPs, cannot always tell what is being transported at the higher levels, since it may be in an unknown representation or encrypted. Second, new applications can come into operation quickly, and the ISPs will not necessarily know what the nature of that application is. The end-node is where that application runs, and thus the end node is a natural locus of control.

9.5. The Internet evolves rapidly

As has been illustrated above by several examples, the Internet is designed to evolve rapidly. The goal is to support new high-level services without requiring changes to the routers, to allow individual end-nodes to install and operate new software without negotiation without other parties, and to change the routers as necessary when there are new service demands. As much of the router as possible is usually implemented in software, so that it can be changed without a hardware upgrade.

The modes in which the Internet operates today are not necessarily what we will see in a small number of years. New applications, new delivery modes, new Internet level services, and new data representations can all be anticipated. We can also anticipate new billing and cost-allocation mechanisms.

Glossary

Address (Internet): Packets crossing the Internet carry a destination address to indicate where the packet is going, and a source address to indicate where it is coming from. Current Internet addresses are 4 bytes (32 bits) long. (See the definition of *bit*.) They are by custom written down by taking each byte of the address, and writing it as an integer in the range of 0 to 255, separated by periods (or, in the terminology of the field, dots). Thus, an Internet address might be written as 18.26.0.120. There is a proposal for a new generation of Internet packet header, called IPv6, with much larger addresses. This expansion may be necessary to deal with all the computers that will be connected to the Internet over the next decade.

Application: a user-visible high-level service. Applications are normally implemented as software that runs on end-nodes attached to the Internet.

Bit: the smallest unit of information stored and transmitted in a computer system. A bit can take on only one of two values, normally expressed as 0 or 1. A number of bits can be used in sequence to store an integer, a character, and so on. For example, in eight successive bits (called a *byte*) a total of 256 different patterns of 0's and 1's can occur. So a byte could be used to store an integer in the range of 0 to 255, or one of 256 characters. Since there are less than 256 characters in normal English text, including lower and upper case letters, punctuation and numbers, it is traditional to store a text document in a computer by organizing the memory of the computer into bytes, and putting one character in each successive byte.

Byte: see bit.

Checksum: a mathematical computation performed on data (a sequence of bytes) to yield a small numerical value. The outcome of this computation depends on all the data, so changing any one bit of the data will change the computed value. The checksum can thus be used to detect if any part of a message has been damaged as it is transmitted across the network.

Congestion: the condition where a router has received more packets than the outgoing link can carry away, so that the memory of the router is forced to store these packets until the condition passes. If the congestion persists, the router may have to discard some of the stored packets.

Datagram: a term used to describe a packet in one particular sort of network, in which the routers in the network make a separate forwarding decision for each packet as it is received. The alternative would be to require that the sender notify the network before sending a stream of packets, so that the network could pre-compute how all of these packets are to be processed.

Encryption: the act of disguising data before sending it across the network, so that it is meaningless to anyone who does not have the means (the *encryption key*) to reverse the encryption process.

End-node: a computer or other device that is attached to the Internet for the purpose of transmitting and receiving packets and participating in the high-level services of the Internet. A personal computer (PC) is an example of an end-node, as is a Web server.

Field: term used to describe each separate part of a packet header, such as a destination address. A field will typically be some specified number of bytes in length.

Header: control information such as a destination address that is placed at the beginning of a packet to permits its proper processing by the various protocols that deal with it.

Interface: a term used in Computer Science to describe what parts of a facility or system are externally visible. The usual goal is to make a complex system easy to understand and use by creating a simple interface to that system. A well-designed interface will preserve most of the useful features of the system while hiding the irrelevant ones.

Multicast: the process by which a packet sent from a source is delivered to multiple recipients. The routers implement this process by forwarding the incoming packet along multiple outgoing links as appropriate to fan out to all the intended destinations.

Protocol: the term used to describe the various technical specifications of the Internet. A protocol is a specification of permitted behavior by the various components of the Internet such that the resulting overall operation of the network is correct, and the participants can accomplish their intent.

Packet: a unit of information, typically no more than 1000 to 2000 bytes, with a series of headers at the front, that is transmitted across the Internet from a source to one or more destination.

Queue: a part of the memory of a computer (for example a router) where packets can be held until they can be processed or transmitted. Packets are held in a queue if they cannot be sent at once due to congestion.

Router: the physical device (typically a computer with special software) that connects communication links together to form the Internet, and forwards packets from one link to the next.

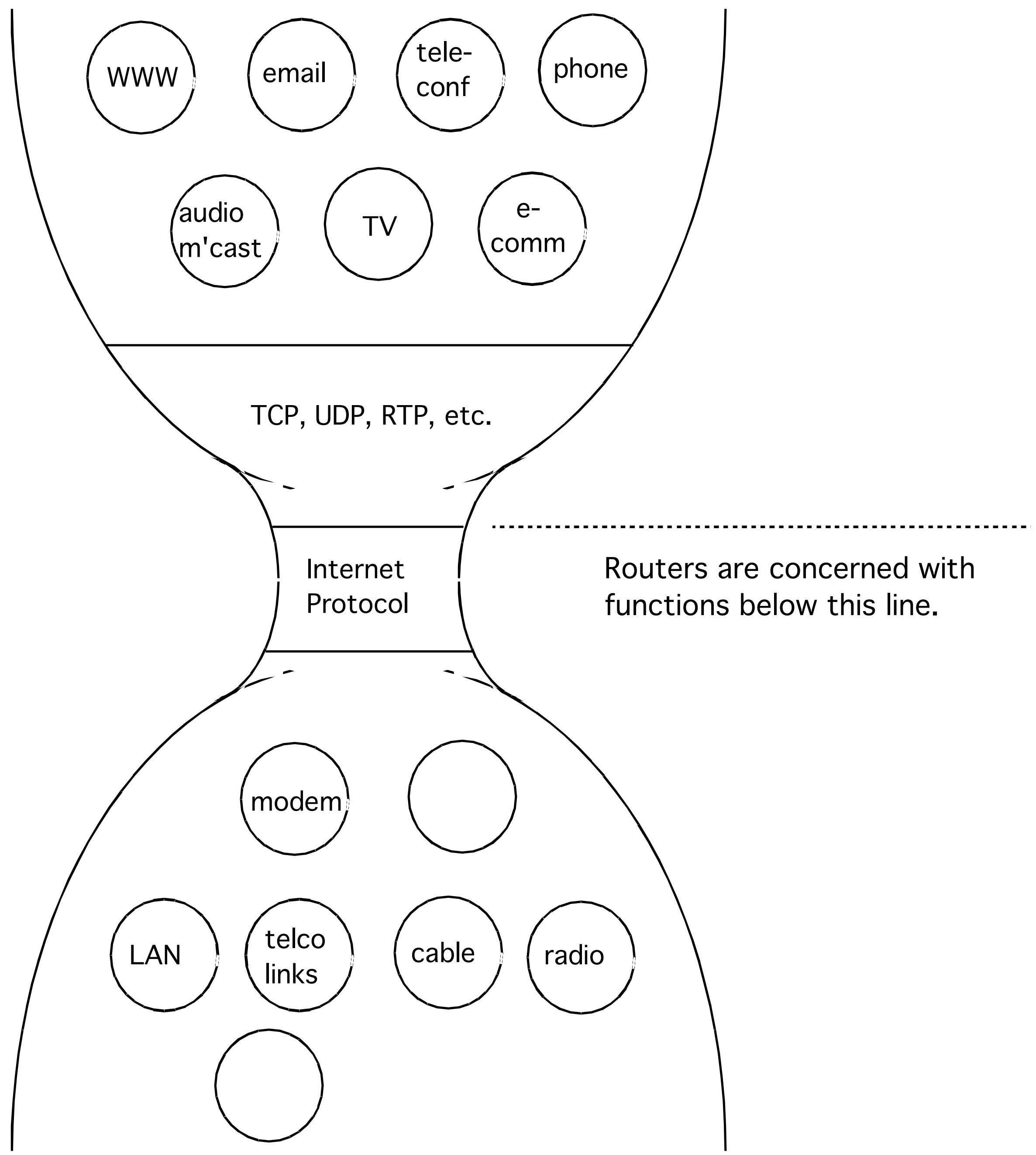

Figure 1: The "hour-glass" model of the Internet

An illustration of the organization of the Internet protocols. There are a number of high-level services shown at the top, which are to be implemented using the various network technologies at the bottom. The Internet protocol, illustrated at the narrow point in the figure, specifies a set of basic communication services. The high level services make use of this basic service, and the network technologies support this service. By this specification, the details of each part are hidden from the other.

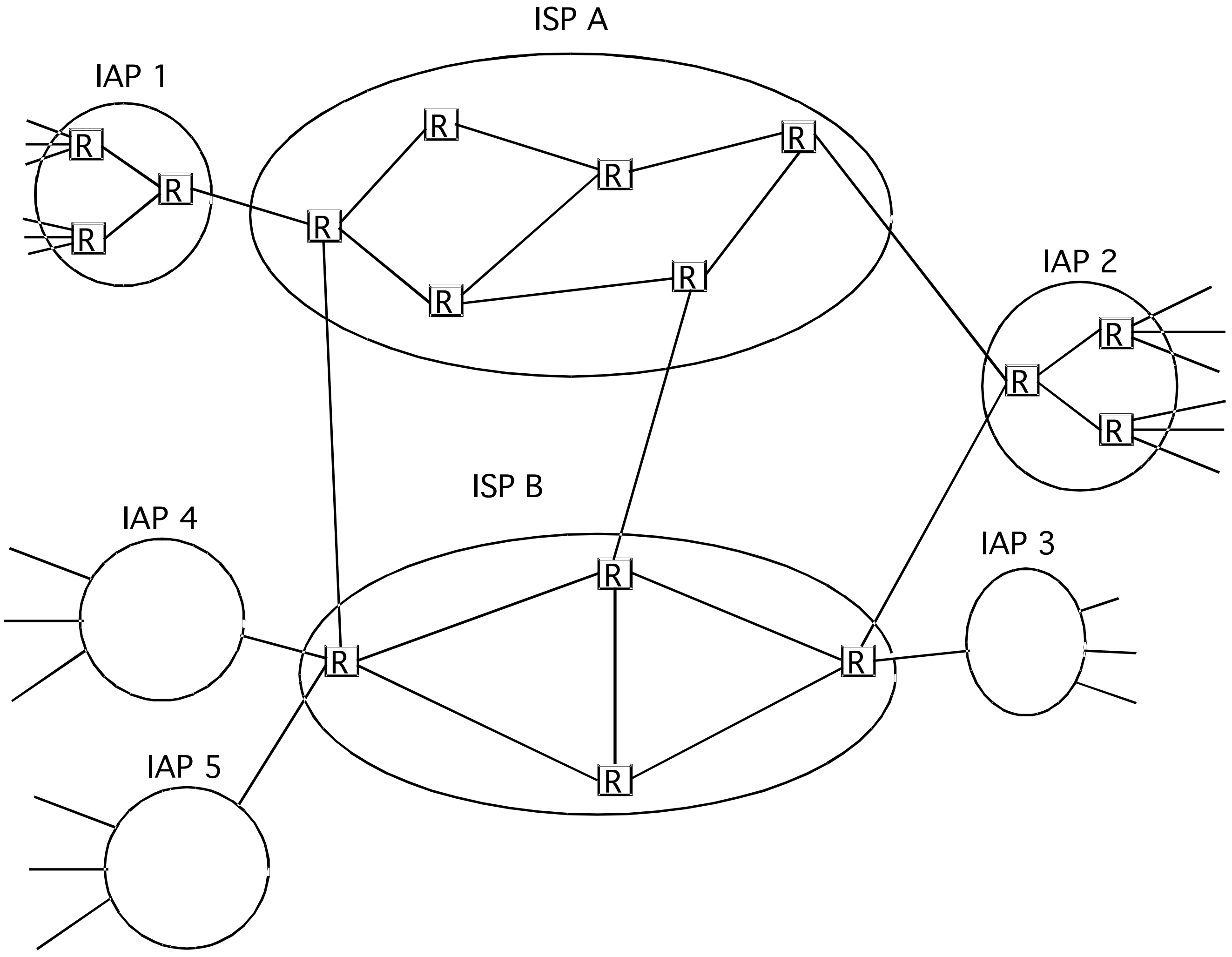

Figure 2: The Internet as a mesh of links and routers

An illustration showing that the Internet is composed of links connected by routers. The links are shown here conceptually as point-to-point connections such as a line provided by a telephone company.

Different parts of the Internet are operated by different entities: Internet Service Providers (ISPs) and Internet Access Providers (IAPs), which provide connections to the end-node computers that attach to the Internet. Although not illustrated here, other entities such as institutional users can also operate parts of the Internet

IAPs attach to ISPs to carry their traffic. IAPs can attach to more than one ISP, and ISPs can interconnect in more that one point. Traffic between 2 IAPs (e.g. IAP 1 and 3) may have to cross more than one ISP.

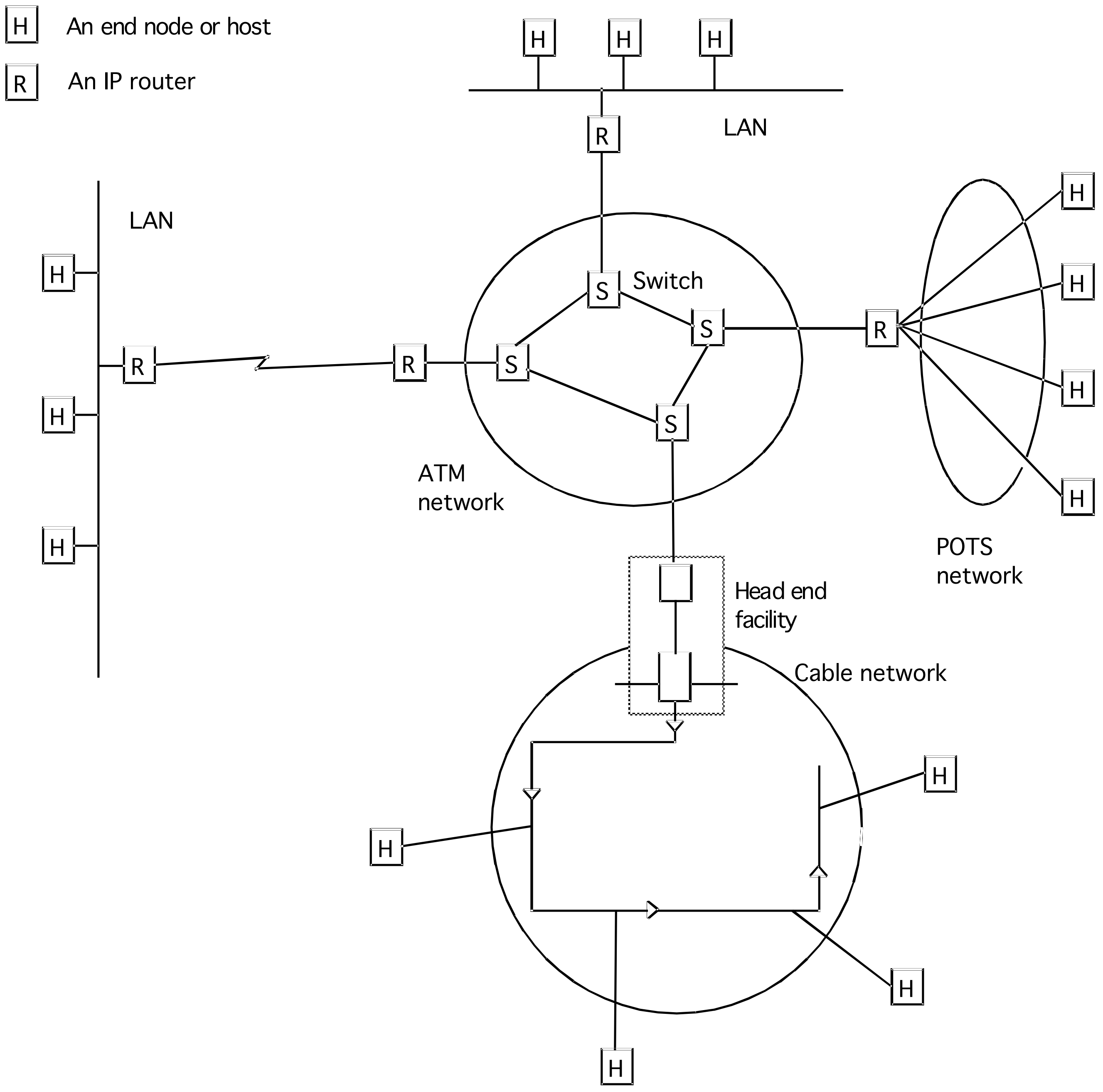

Figure 3: The Internet is built out of different sorts of communication technology

An illustration of several different sorts of networks connected by routers. The networks include a local area network (LAN), a switched network based on the ATM technology, a dialup access network and a cable television distribution facility.

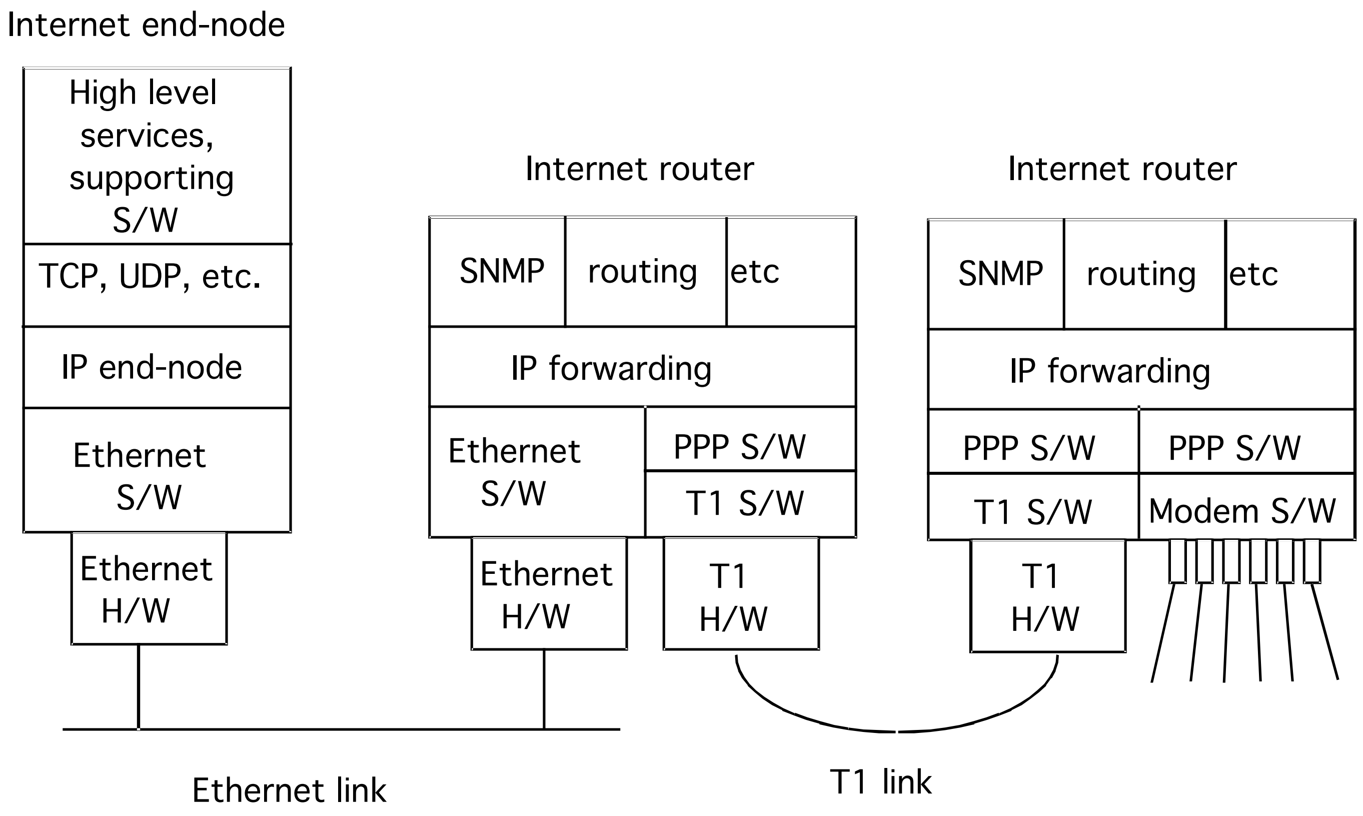

Figure 4: Implementing the protocol layers of the Internet

Illustration of the hardware and software modules found in end-nodes and routers. The illustrated end-node is attached to an Ethernet Local Area Network, and has software to operate this Ethernet. The IP (Internet Protocol) software uses this Ethernet software to transmit and receive packets across the LAN. The intermediate protocol software (TCP, UDP, etc.) uses the IP software to send packets. The high level service software uses the intermediate software.

Inside the router, there is no high-level software, and no intermediate protocol software to support it. Depending on which networks are connected to the routers (Ethernet, T1 links and modems are illustrated here) there is matching software. The IP forwarding software uses that network-specific software to receive and transmit packets. There is also routing software to perform the routing computation, SNMP (Simple Network Management Protocol) to allow human operators to control the router, and so on.

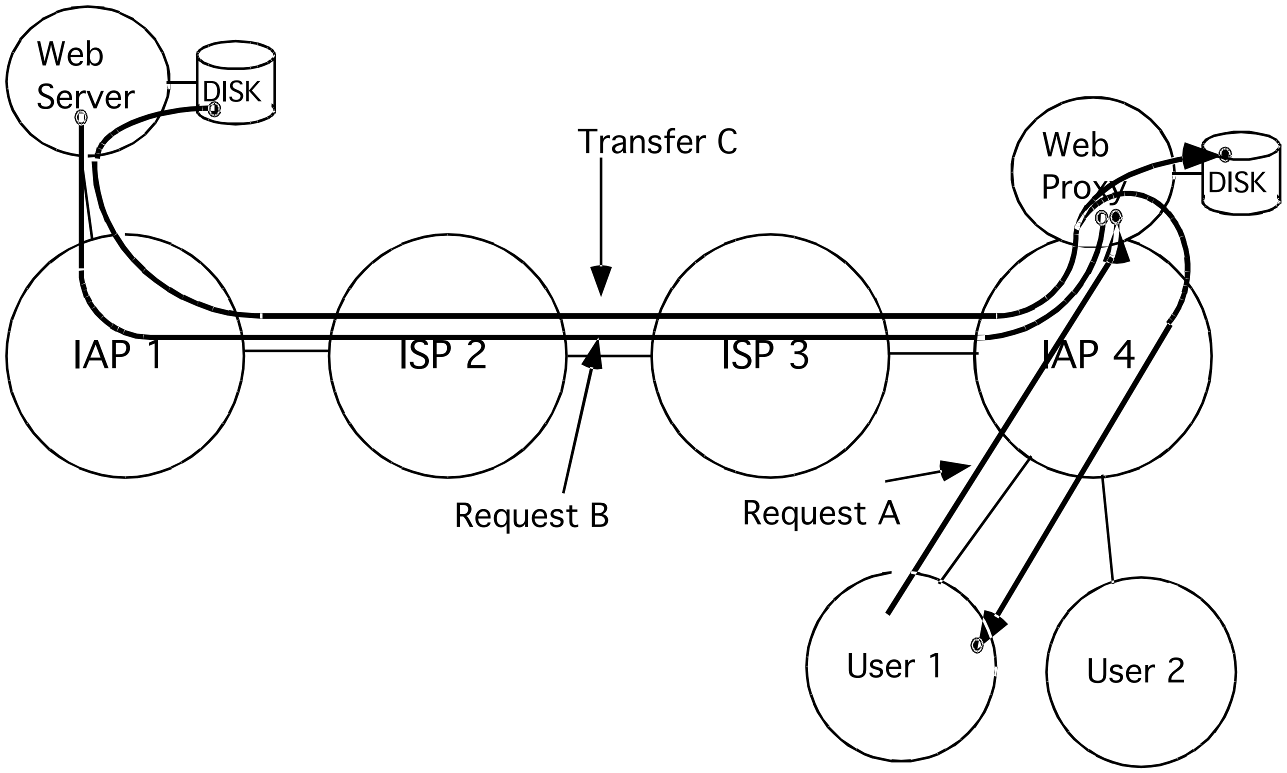

Figure 5: Retrieval of a Web page from a server via a proxy

This figure illustrates the transfer of a Web page from a Web server to a user (User 1) making use of a Web proxy operated by the IAP that serves the user (IAP 4). If there were no Web proxy in place, the request would go directly from the user to the Web server, and the Web page would be transmitted back directly to the user. Using the Web proxy, User 1 requests the page from the proxy (Request A). In this case, the page is assumed not to be on the proxy, so it is requested from the original server by the proxy (Request B). The required page is transmitted back to the proxy, where it is stored on disk, and also sent on to User 1.

Figure 6 illustrates the case when User 2 subsequently requests the same Web page.

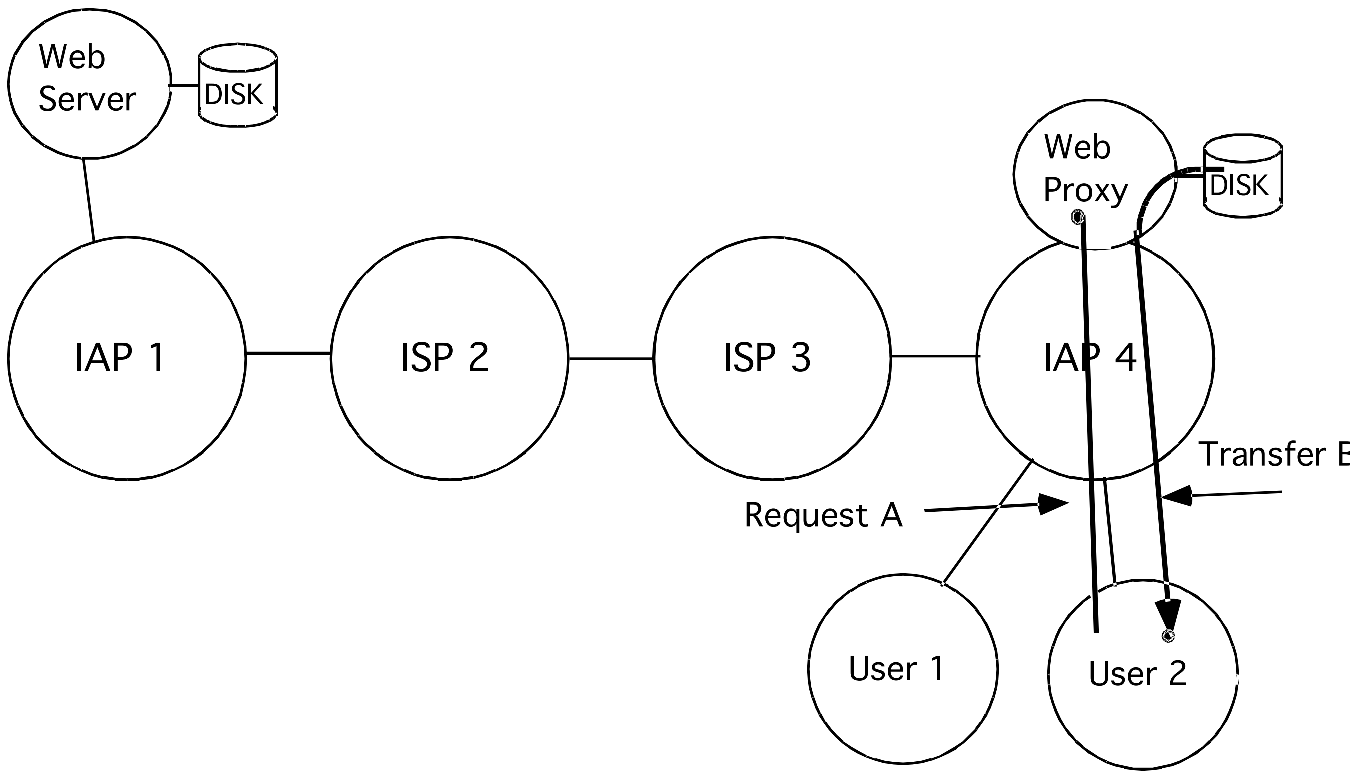

Figure 6: Retrieval of a Web page located on a proxy server

This figure illustrates the case of a Web retrieval in which the needed page is located on the proxy, because some user has previously requested it. In this case, User 2 requests a page from the proxy, and since a copy of the page is currently stored on the proxy, no transfer occurs from the server, but the page is transferred from the proxy to the user.

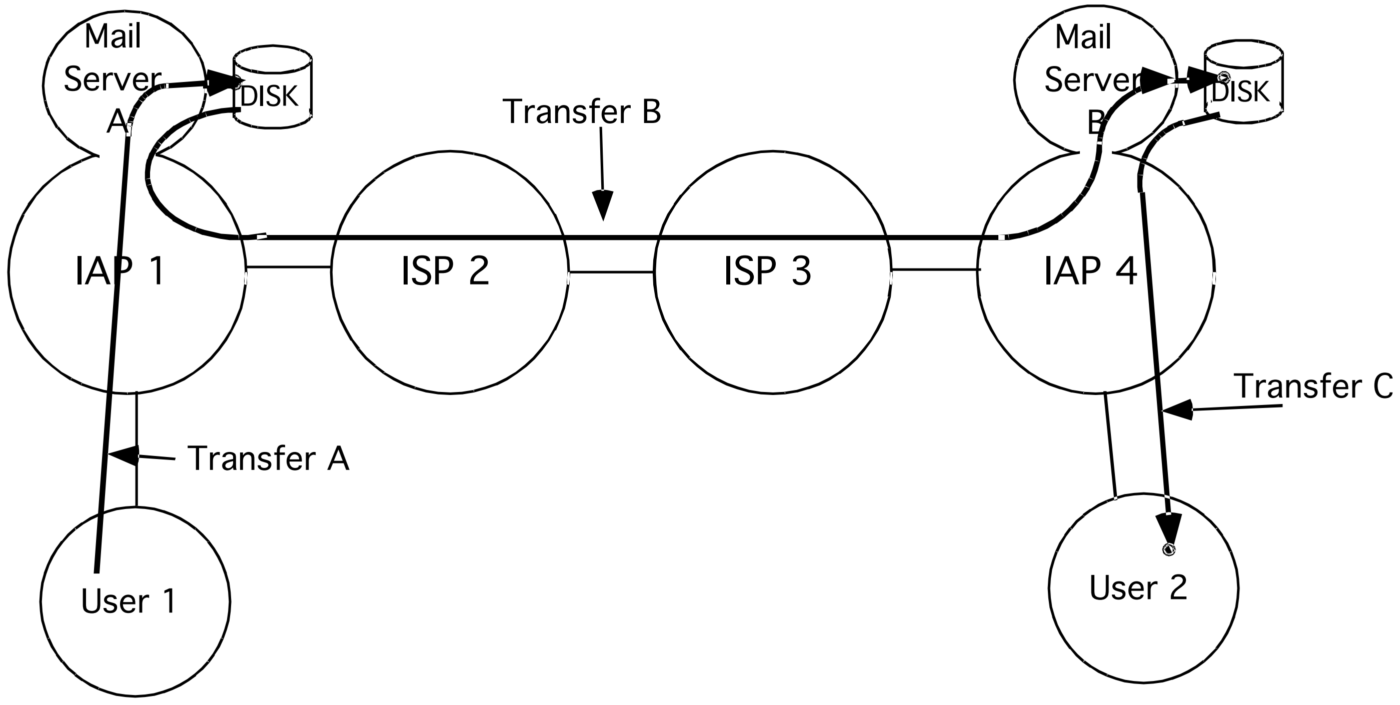

Figure 7: Typical pattern of mail transmission

This figure issustrates the typical sequence of transmission that constitute the sending of mail. In principle, User 1 could transfer the mail directly to User 2. In practice, the users will take advantage of mail servers that are operated by their respective IAPs, IAP 1 and IAP 4. The mail is transmitted to Mail Server A, after which User 1 can disconnect from the Internet. Server A will attempt to transmit the mail to server B. When this succeeds, the mail will be stored in Mail server B until User 2 connects to the Internet and requests it. In response to this request, server B will transmit the mail to User 2. In this sequence, the transfers A and B are initiated by the sender, while transfer C is initiated by the receiver.