MIT 18.996: Topic in TCS: Internet Research Problems **Spring 2002**

Lecture 2 — February 13, 2002

Lecturer: Ravi Sundaram *Scribes: Zeeshan Syed, Asfandyar Qureshi*

2.1 History

The precursor to the Internet was an academic experiment in the 1960s. Funded by the Defence Advanced Research Projects Agency (DARPA), the proposal for the ARPANET was first published by Lawrence G. Roberts in 1966[1]. Influenced by his contact with Leonard Kleinrock at MIT[2], Roberts envisioned ARPANET as a wide area packet switching communications network. Work on similar concepts had proceeded in parallel at the RAND corporation and at NPL, and some of their standards were adopted by DARPA.

By the end of 1969, four host computers were connected together into the initial ARPANET, and the budding Internet was off the ground. One by one computers at UCLA (hooked up by September), the Stanford Research Institute (SRI - October), the University of California Santa Barbara (November) and finally the University of Utah (December) were brought online, linked together by 56kbps links. On the 29th of October, the first packets were sent by Charley Kline at UCLA as he tried logging into SRI. This initial attempt resulted in the system crashing as the letter G of LOGIN was entered.

Nonetheless, computers were added quickly to the ARPANET during the following years. In 1970 the first host-to-host protocol dubbed the Network Control Protocol (NCP) was published by the Network Working Group[3]. Later that year, the first cross-country links were added between UCLA, RAND and MIT. In 1973 The ARPANET went international with connections to University College in London, England and the Royal Radar Establishment in Norway.

In March of 1972 Ray Tomlinson at BBN wrote the basic email message send and read software, motivated by the need of the ARPANET developers for an easy coordination mechanism. Email quickly became the most popular application for over a decade. The ARPANET became a high-speed digital post office as people used it to collaborate on research projects and discuss topics of various interests.

However the rapid growth of the ARPANET resulted in a need for improved software and protocols to handle the increasing complexity. In particular, NCP relied on ARPANET to provide end-to-end reliability. If any packets were lost, the protocol (and presumably any applications it supported) would come to a grinding halt. The collaboration between Robert Kahn (at DARPA) and Vinton Cerf (at Stanford) led to the present day protocols. They published the specification for the Transmission Control Protocol (TCP) and the Internet

[1]G. Roberts, MIT: “Towards a Cooperative Network of Time-Shared Computers” (1966)

[2]Leonard Kleinrock, MIT: “Information Flow in Large Communication Nets” (May 1961)

[3]C.S. Carr, S. Crocker, V.G. Cerf, “Host-Host Communication Protocol in the ARPA Network,” in AFIPS Proceedings of SJCC

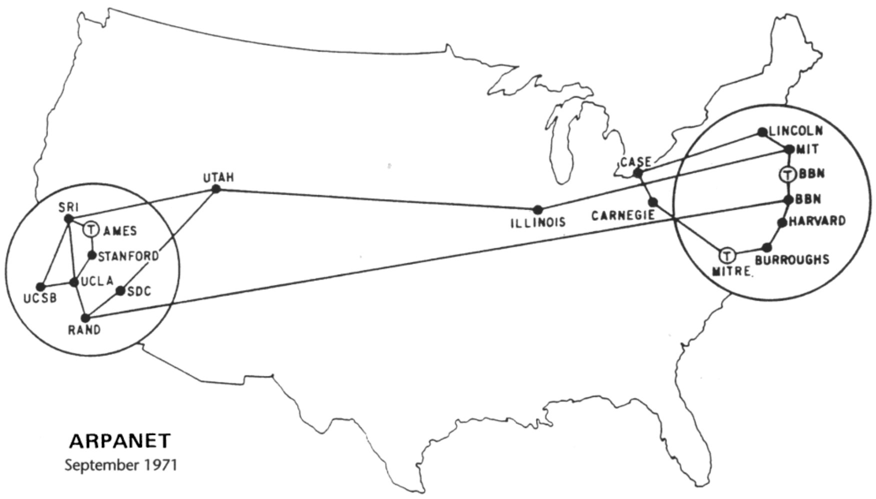

Figure 2.1. ARPANET 1971

Protocol (IP) in 1978[4]. The two protocols together provided the means to transmit packets reliably to any host on the network.

By 1982 ARPANET had split into MILNET (the military part) and what later evolved into the Internet. Around this time, in the early 1980s, networks such as USENET, BITNET, CSNET and a host of others came into existence. With the exception of BITNET and USENET, these early networks (including ARPANET) were purpose-built - intended for, and largely restricted to, closed communities of scholars. Agreements between DARPA and others early in the decade allowed the interconnection of some of these networks.

In the early 1980s, the National Science Foundation (NSF) explicitly announced their intent to serve the entire higher education community. As a result of their funding the NSFNET emerged in 1986 - this would later form the communications backbone of the early Internet. The NSF also established super-computing centers - at Princeton, Cornell, Pittsburgh, UIUC and UCSD - in order to provide high-computing power to everyone. This resulted in an explosion of connections, especially from universities. In fact, although in 1986 there were a litle over 5,000 internet hosts, this number had grown to over 28,000 by the next year. However, NSF had prohibited usage of the NSFNET backbone - the national-scale segment of the NSFNET - for purposes "not in support of Research and Education."

On the second of November 1988, an internet worm was released across the networks affecting some 6,000 of the 60,000 computers online. Eventually stopped and disassembled by researchers at Berkley and MIT, the worm exposed some serious security holes in the growing network. In response to the needs exhibited during this incident, the Computer Emergency Response Team (CERT) was formed by DARPA.

There had been less than 200 computers online in 1978. By 1989, there were well over 100,000 computers. This meant that having a single table of hosts was no longer feasible,

[4]V. G. Cerf and R. E. Kahn, "A protocol for packet network interconnection", IEEE Trans. Comm. Tech., vol. COM-22, V 5, pp. 627-641, May 1974

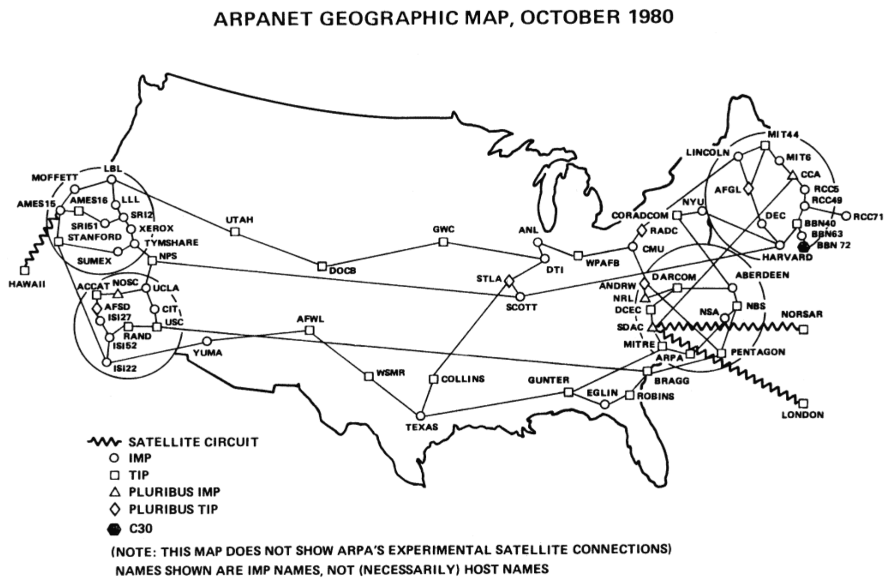

Figure 2.2. ARPANET 1980

and the Domain Name System (DNS) was invented by Paul Mockapetris of USC/ISI in 1984. The DNS permitted a scalable distributed mechanism for resolving hierarchical host names (e.g. www.akamai.com) into an Internet address.

The increase in the size of the Internet also challenged the capabilities of the routers and created problems of bandwidth congestion which had not been considered by TCP/IP. By 1989, TCP had been revised to provide congestion control to avoid collapsing the network. Routers were also modified to handle the increasing complexity. Originally, there was a single distributed algorithm for routing that was implemented uniformly by all the routers in the Internet. As the number of networks in the Internet exploded, this initial design could not expand as necessary, so it was replaced by a hierarchical model of routing, with an Interior Gateway Protocol (IGP) used inside each region of the Internet, and an Exterior Gateway Protocol (EGP) used to tie the regions together.

In 1990, a happy victim of its own unplanned, unexpected success, the ARPANET was decommissioned, leaving the vast network-of-networks called the Internet. The number of hosts exceeded 300,000 at this time. Nevertheless, even in 1990, commercial traffic was still banned on the NSFNET backbone. During this time, there was a real fear among researchers that the Internet would collapse under its own weight and so there was a reluctance to increase the traffic on it. Eventually, in 1991 the commercial ban was lifted and this led to the development of a great many Internet applications.

At the University of Minnesota, a team led by computer programmer Mark MaCahill released "gopher" in 1991, the first point-and-click way of navigating the files of the Internet in 1991. Originally designed to ease campus communications, gopher was freely distributed on the Internet. MaCahill called it "the first Internet application my mom can use." 1991 was also the year in which Tim Berners-Lee, working at CERN in Switzerland, posted the first computer code of the World Wide Web in a relatively innocuous newsgroup, "alt.hypertext." Marc Andreesen and a group of student programmers at NCSA (the National Center for Supercomputing Applications located on the campus of University of Illinois at Urbana Champaign) would eventually develop a graphical browser for the World Wide Web called Mosaic. Andreesen would go on to found Netscape.

Eventually, the government began to withdraw from the maintenance of the Internet. In May 1993, there was the last NSFNET solicitation for private Network Access Points (NAPs). InterNIC was created by NFS the same year to provide specific Internet services such as those related to registration and host databases. Managing the Internet, therefore, eventually moved into the private domain with less and less government supervision.

Nonetheless, during a March 1999 CNN interview, the vice-president Al Gore boasted that he had taken "the initiative in creating the Internet."

The phenomenal growth of the Internet had resulted in the need for a new communications backbone. In 1995, the NSFNET backbone was replaced by vBNS - a high performance backbone service developed by MCI, linking certain universities and research centers at 155Mbps and higher.

The Internet has come a long way from the four hosts that initially came online in 1969. In 2002, there were an estimated 350 million hosts - a number which is projected to grow considerably. The initial designers of the protocols and architecture which make up the Internet did not consider the multitude of issues that arise out of the present level of complexity: routing is sub-optimal, connections and hosts are unreliable, there is no inherent security, no directory or index and there is an ever shrinking address space.

The inherent decentralized nature of the Internet brings with it many challenges and opportunities. Along the way, many people have tweaked the original architecture to enable it to handle new situations, but predictably enough, at present the Internet does not deliver optimal performance.[5]

2.2 Internet Protocols

2.2.1 Layering of Networks

Developing software for networks is a complicated task by virtue of the sheer number of different underlying services that must be provided. These include sending bits across various transmission media, routing data between machines in different networks, providing error detection, correction and reliable connectivity, regulating flow and defining application formats. Most of these functions are largely network independent and sufficiently generic to be standardized. Consequently, to reduce design complexity, networks are organized as a series of **layers** or levels, each built upon the one below it. The actual number, names, contents and functions of each layer may differ from network to network but their purpose is always to offer certain services to higher layers, shielding them from the details of how the

[5] This section is primarily based on the following two sources: [1] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff, "A Brief History of the Internet" and [2] Robert H. Zakon, "Hobbes' Internet Timeline", http://www.zakon.org/robert/internet/timeline

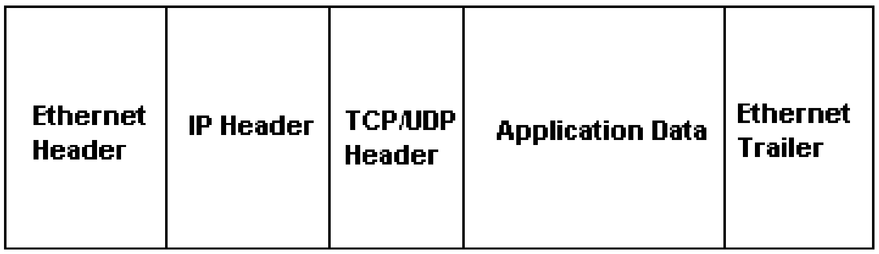

Figure 2.3. Example of a Packet with Headers and Trailers

offered services are actually implemented. In this way, layers implement abstractions and present a well-defined set of services to applications built on top of them.

Each layer in a network typically adds extra information to the original packet it receives from the layer directly above it (as shown in figure 2.3). This is intended only for the "eyes" of the corresponding layer at the other end and is called a **header** when added to the beginning of the packet and **traile**r when tagged on at the end. The part of the data that belongs to the layer directly above is called the **payload**. Header and trailers normally contain control information such as as sizes, times and other such fields. They may also contain sequence numbers (to allow the corresponding layer at the destination machine to deliver messages in the right order if the lower layers do not maintain sequence). All this leads to the size of a typical packet being restricted to between 46 and 1500 bytes.

Entities comprising the corresponding layers on different machines are called **peers**. The rules and conventions used in communication between these are known as **protocols**. In other words, a protocol is fundamentally an agreement between the communicating parties on how communication should proceed (by specification of the format and meaning of messages sent between peers).

The layered network architecture is discussed in more detail in Appendix A.

2.2.2 End-to-End Argument

The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. The basic thrust of this argument is that an application knows best what its real communication requirements are, and for a lower layer to try to implement any feature beyond minimally transporting the data risks implementing something that is not quite what is needed, in which case, the application will end up re-implementing that function in a different way that it finds more useful.

The argument is discussed in considerable detail elsewhere[6].

[6]J.H. Saltzer, D.P. Reed and D.D. Clark. "End-to-End Arguments in System Design", MIT-LCS

2.2.3 Contemporary Protocols

This section provides a brief discussion of some of the commonly used protocols. A more in-depth description of each can be found in the RFC (Request For Comment) document associated with each protocol[7].

2.2.3.1 Internet Protocol (IP)

This provides all of Internet's data transport services, with all other protocols ultimately either layered atop IP, or used to support IP from below.

IP is the Internet's most basic protocol. In order to function in a TCP/IP network, a network segment's only requirement is to forward IP packets. In fact, a TCP/IP network can be defined as a communication medium that can transport IP packets. Almost all other TCP/IP functions are constructed by layering atop IP.

IP is a datagram-oriented protocol, treating each packet independently. This means each packet must contain complete addressing information. Also, IP makes no attempt to determine if packets reach their destination or to take corrective action if they do not. IP only checksums the header, the contents of the packet are not validated in this way.

IP provides several services:

- Addressing: IP headers contain 32-bit addresses which identify the sending and receiving hosts. These addresses are used by intermediate routers to select a path through the network for the packet.

- Fragmentation: IP packets may be split, or fragmented, into smaller packets. This permits a large packet to travel across a network which can only handle smaller packets. IP fragments and reassembles packets transparently.

- Packet timeouts: Each IP packet contains a Time To Live (TTL) field, which is decremented every time a router handles the packet. If TTL reaches zero, the packet is discarded, preventing packets from running in circles forever and flooding a network. This is typically set to 30 at the start.

- Type of service: IP supports traffic prioritization by allowing packets to be labelled with an abstract type of service. This is rarely used.

- Options: IP provides several optional features, allowing a packet's sender to set requirements on the path it takes through the network (source routing), to trace the route a packet takes (record route), and to label packets with security features.

2.2.3.2 Internet Control Message Protocol (ICMP)

This is a required protocol tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation or malfunction. Of course, since ICMP uses IP, ICMP packet delivery is unreliable, so hosts can't count on receiving ICMP packets for any network problem. Some of ICMP's functions are:

[7] The relevant RFCs can be found at : www.ietf.org, www.w3.org and www.freesoft.org

- Error announcement: Network errors are announced, such as a host or entire portion of the network being unreachable, due to some type of failure. A TCP or UDP packet directed at a port number with no receiver attached is also reported via ICMP.
- Congestion announcement: When a router begins buffering too many packets, due to an inability to transmit them as fast as they are being received, it will generate ICMP *Source Quench* messages. Directed at the sender, these messages should cause the rate of packet transmission to be slowed. Of course, generating too many Source Quench messages would cause even more network congestion, so they are used sparingly.
- Timeout announcement: If an IP packet's TTL field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. TraceRoute is a tool which maps network routes by sending packets with small TTL values and watching the ICMP timeout announcements.
- Assistance with troubleshooting: ICMP supports an *Echo* function, which just sends a packet on a round–trip between two hosts. Ping, a common network management tool, is based on this feature. Ping will transmit a series of packets, measuring average round–trip times and computing loss percentages.

2.2.3.3 Transmission Control Protocol (TCP)

This is largely intended to make up for IP's deficiencies by providing reliable, stream-oriented connections that hide most of IP's shortcomings. The TCP/IP protocol suite gets its name because most such protocols are based on TCP, which is in turn implemented on IP.

TCP adds a great deal of functionality to the IP service it is layered over:

- Streams: TCP data is organized as a stream of bytes, much like a file. The datagram nature of the network is concealed. A mechanism (the Urgent Pointer) exists to let out-of-band data be specially flagged.
- Reliable delivery: Sequence numbers are used to coordinate which data has been transmitted and received. TCP will arrange for retransmission if it determines that data has been lost.
- Network adaptation: TCP will dynamically learn the delay characteristics of a network and adjust its operation to maximize throughput without overloading the network.
- Flow control: TCP manages data buffers, and coordinates traffic so its buffers will not overflow. Fast senders will be stopped periodically to keep up with slower receivers.

No matter what the particular application, TCP almost always operates *full-duplex*. The algorithms described below operate in both directions, in an almost completely independent manner. It's sometimes useful to think of a TCP session as two independent byte streams, traveling in opposite directions. No TCP mechanism exists to associate data in the forward and reverse byte streams. Only during connection start and close sequences can TCP exhibit asymmetric behavior (i.e. data transfer in the forward direction but not in the reverse, or vice versa).

TCP uses a 32-bit sequence number that counts bytes in the data stream. Each TCP packet contains the starting *sequence number* of the data in that packet, and the sequence number (called the *acknowledgment number*) of the last byte received from the remote peer. With this information, a sliding-window protocol is implemented. Forward and reverse sequence numbers are completely independent, and each TCP peer must track both its own sequence numbering and the numbering being used by the remote peer.

TCP uses a number of control flags to manage the connection. Some of these flags pertain to a single packet, such as the URG flag indicating valid data in the Urgent Pointer field, but two flags (SYN and FIN), require reliable delivery as they mark the beginning and end of the data stream. In order to insure reliable delivery of these two flags, they are assigned spots in the sequence number space. Each flag occupies a single byte.

Each endpoint of a TCP connection will have a buffer for storing data that is transmitted over the network before the application is ready to read the data. This lets network transfers take place while applications are busy with other processing, improving overall performance.

To avoid overflowing the buffer, TCP sets a *Window Size* field in each packet it transmits. This field contains the amount of data that may be transmitted into the buffer. If this number falls to zero, the remote TCP can send no more data. It must wait until buffer space becomes available and it receives a packet announcing a non-zero window size.

Sometimes, the buffer space is too small. This happens when the network's bandwidth-delay product exceeds the buffer size. The simplest solution is to increase the buffer, but for extreme cases the protocol itself becomes the bottleneck (because it doesn't support a large enough Window Size). Under these conditions, the network is termed an LFN (Long Fat Network - pronounced elephant). RFC 1072 discusses LFNs.

When a host transmits a TCP packet to its peer, it must wait a period of time for an acknowledgment. If the reply does not come within the expected period, the packet is assumed to have been lost and the data is retransmitted. The obvious question - How long do we wait? - lacks a simple answer. Over an Ethernet, no more than a few microseconds should be needed for a reply. If the traffic must flow over the wide-area Internet, a second or two might be reasonable during peak utilization times. If we're talking to an instrument package on a satellite hurtling toward Mars, minutes might be required before a reply. There is no one answer to the question: How long?

All modern TCP implementations seek to answer this question by monitoring the normal exchange of data packets and developing an estimate of how long is "too long". This process is called Round-Trip Time (RTT) estimation. RTT estimates are one of the most important performance parameters in a TCP exchange, especially when you consider that on an indefinitely large transfer, all TCP implementations eventually drop packets and retransmit them, no matter how good the quality of the link. If the RTT estimate is too low, packets are retransmitted unnecessarily; if too high, the connection can sit idle while the host waits to timeout.

2.2.3.4 User Datagram Protocol (UDP)

This provides users access to IP-like services. UDP packets are delivered just like IP packets - connection-less datagrams that may be discarded before reaching their targets. UDP is useful when TCP would be too complex, too slow, or just unnecessary.

UDP provides a few functions beyond that of IP:

- Port Numbers: UDP provides 16-bit port numbers to let multiple processes use UDP services on the same host. A UDP address is the combination of a 32-bit IP address and the 16-bit port number.
- Checksumming: Unlike IP, UDP does checksum its data, ensuring data integrity. A packet failing checksum is simply discarded, with no further action taken.

2.2.3.5 Simple Mail Transfer Protocol (SMTP)

This is the Internet's standard host-to-host mail transport protocol and traditionally operates over TCP, port 25. In other words, a UNIX user can type telnet hostname 25 and connect with an SMTP server, if one is present.

SMTP uses a style of asymmetric request-response protocol popular in the early 1980s, and still seen occasionally, most often in mail protocols. The protocol is designed to be equally useful to either a computer or a human, though not too forgiving of the human. From the server's viewpoint, a clear set of commands is provided and well-documented in the RFC. For the human, all the commands are clearly terminated by newlines and a HELP command lists all of them. From the sender's viewpoint, the command replies always take the form of text lines, each starting with a three-digit code identifying the result of the operation, a continuation character to indicate another lines following, and then arbitrary text information designed to be informative to a human.

If mail delivery fails, sendmail (the most important SMTP implementation) will queue mail messages and retry delivery later. However, a backoff algorithm is used, and no mechanism exists to poll all Internet hosts for mail, nor does SMTP provide any mailbox facility, or any special features beyond mail transport. For these reasons, SMTP isn't a good choice for hosts situated behind highly unpredictable lines (like modems). A better-connected host can be designated as a DNS mail exchanger, then arrange for a relay scheme. Currently, there two main configurations that can be used. One is to configure POP mailboxes and a POP server on the exchange host, and let all users use POP-enabled mail clients. The other possibility is to arrange for a periodic SMTP mail transfer from the exchange host to another, local SMTP exchange host which has been queuing all the outbound mail. Of course, since this solution does not allow full-time Internet access, it is not too preferred.

2.2.3.6 Simple Network Management Protocol (SNMP)

This is essentially a request-reply protocol running over UDP (ports 161 and 162), though TCP operation is possible. SNMP is an asymmetric protocol, operating between a management station (smart) and an agent (dumb). The agent is the device being managed - all its software has to do is implement a few simple packet types and a generic get-or-set function on its MIB variables. The management station presents the user interface. Simple management stations can be built with UNIX command-line utilities. More complex (and expensive) ones collect MIB data over time and use GUIs to draw network maps.

An SNMP operation takes the form of a Protocol Data Unit (PDU),which is basically a fancy word for packet. SNMP version 1 supports five possible PDUs:

- *GetRequest/SetRequest* supplies a list of objects and, possibly, values they are to be set to (SetRequest). In either case, the agent returns a GetResponse.
- *GetResponse* informs the management station of the results of a GetRequest or SetRequest by returning an error indication and a list of variable/value bindings.
- *GetNextRequest* is used to perform table transversal, and in other cases where the management station does not know the exact MIB name of the object it desires. GetNextRequest does not require an exact name to be specified; if no object exists of the specified name, the next object in the MIB is returned. Note that to support this, MIBs must be strictly ordered sets (and are).
- *Trap* is the only PDU sent by an agent on its own initiative. It is used to notify the management station of an unusual event that may demand further attention (like a link going down). In version 2, traps are named in MIB space. Newer MIBs specify management objects that control how traps are sent.

2.2.3.7 HyperText Transfer Protocol

This operates over TCP connections, usually to port 80, which can be overridden and another port used. After a successful connection, the client transmits a request message to the server, which sends a reply message back. HTTP messages are human-readable, and an HTTP server can be manually operated with a command such as telnet server 80.

The simplest HTTP message is "GET url", to which the server replies by sending the named document. If the document doesn't exist, the server will probably send an HTML-encoded message stating this. This simple method offers poor error handling and has been deprecated in favor of the more elaborate scheme outlined below.

A complete HTTP 1.0 message begins "GET url HTTP/1.0". The addition of the third field indicates that full headers are being used. The client may then send additional header fields, one per line, terminating the message with a blank line. The server replies in a similar vein, first with a series of header lines, then a blank line, then the document proper.

The following is a sample HTTP 1.0 exchange:

```
GET / HTTP/1.0 >
               >
               < HTTP/1.0 200 OK
               < Date: Wed, 18 Sep 1996 20:18:59 GMT
               < Server: Apache/1.0.0
               < Content-type: text/html
               < Content-length: 1579
               < Last-modified: Mon, 22 Jul 1996 22:23:34 GMT
               <
               < HTML document
```

The use of full headers is preferred for several reasons:

- The first line of a server header includes a response code indicating the success or failure of the operation.
- One of the server header fields will be Content-type:, which specifies a MIME type to describe how the document should be interpreted.
- If the document has moved, the server can specify its new location with a Location: field, allowing the client to transparently retry the request using the new URL.
- The Authorization: and WWW-Authenticate: fields allow access controls to be placed on Web documents.
- The Referer: field allows the client to tell the server the URL of the document that triggered this request, permitting savvy servers to trace clients through a series of requests.

In addition to GET requests, clients can also send HEAD and POST requests, of which POSTs are the most important. POSTs are used for HTML forms and other operations that require the client to transmit a block of data to the server. After sending the header and the blank line, the client transmits the data. The header must have included a Content-Length: field, which permits the server to determine when all the data has been received.

2.2.3.8 File Transfer Protocol (FTP)

This is one of oldest Internet protocols still in widespread use and is implemented using the TCP protocol.

The objectives of FTP are to:

- Promote sharing of files (computer programs and/or data
- Encourage indirect or implicit (via programs) use of remote computers
- Shield a user from variations in file storage systems among hosts
- Transfer data reliably and efficiently

FTP servers listen on port 21. FTP uses two connections, one is used as the Data channel and the other is used as the Control channel. Data connections are initiated by the server from its port 20, to a port on the client identified in a PORT command. Files are requested by the client using commands sent on the control channel and they are transmitted using the data channel.

FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

2.3 Addressing

2.3.1 Basic Addressing

Under the existing system, Internet hosts are referenced by means of the **IP address** placed in the IP packet header and used to route packets to their destination. This consists of a 32-bit number, represented by a dotted decimal notation of the form a.b.c.d (for example 10.0.0.1), where a, b, c and d are each 1 byte.

IP addressing makes use of **prefix-based addressing**. The initial prefixes of the IP address can be used for generalized routing decisions. For example, the first 16 bits might identify Akamai, the first 20 bits its office in Boston, the first 26 bits a particular Ethernet in that office and the entire 32 bits a particular host. Additionally, IP addressing also supports **per-interface assignment**. This means that IP addresses are assigned on a per-interface basis so that a single host might have several IP addresses if it has several interfaces. For example, a host with both Ethernet and serial interfaces would have an IP address for each. Thought of another way, an IP address does not really refer to a host. Instead, it refers to an interface.

Each address consists of two parts, a **network number** and a **host number**. During the early days of the internet, the boundaries for these were defined solely on the class of the IP addresses and this addressing scheme was referred to as the **classfull model**. There are five main classes (A to E) and the following discussion focuses on the first three of these.

2.3.1.1 Class A Addressing

A class A network is represented by a 0 in the first bit. This is followed by 7 network bits and 24 host bits. As a result, the initial byte ranges from 0 - 127, leading to a total of 128 possible class A networks (the number is actually less than this as some of the values are reserved) and 16777216 hosts per such network.

2.3.1.2 Class B Addressing

A class B network is represented by a 10 sequence in the first two bits. These are followed by 14 networks bits and 16 host bits. As a result, the initial byte ranges from 128 - 191, leading to a total of 16384 possible class B networks and 655366 hosts per such network.

2.3.1.3 Class C Addressing

A class C network is represented by a 110 sequence in the first three bits. These are followed by 21 network bits and 8 host bits. As a result, the initial byte ranges from 192 - 223, leading to a total of 2097152 possible class C networks and 256 hosts per such network.

2.3.2 Subnets

The constraint that all hosts in a network must have the same network number can cause problems as networks grow. For example, a company starting out with one class C LAN on the Internet may over time end up with many LANs, each with its own router and class C network number. This can be a problem since each time a new network is installed the system administrator has to contact InterNIC to get a new network number, which then needs to be announced. Furthermore, moving a machine from one LAN to another requires changing its IP address, which in turn may mean modifying its configuration files and suffering the scenario whereby some other machine, upon being given the newly-released IP address, may receive data intended for the original machine.

In order to resolve these issues, **subnetting** was introduced as an extension to the original system in the mid-1980s. This refers to the subdivision of a class-based network into several sub-networks (this definition will be updated in the coming sections to reflect the advent of classless routing). As a result of this, the company mentioned above could start out with a class B address instead of a class C address and as its need for machines increases, it could decide to split the 16-bit host number into an 8-bit subnet number and another 8-bit host number, leading to 256 LANs, each with up to 256 hosts. This change will not be visible externally and so allocating a new subnet dos not require contacting InterNIC or changing any external databases.

An important advantage of subnetting is that it drastically reduces the size of the routing tables. This follows from the fact that each router must only keep track of how to reach other subnets and local hosts within its own subnet. When a packet arrives, all that a router needs to do is to perform a Boolean AND of the address in the packet IP header and the router's **subnet mask**. The latter is a 32-bit number that determines how an IP address is split into network and host portions. For example, in a standard class B network, the subnet mask is 255.255.0.0, since the first two bytes are all ones (network bytes) and the last two bytes are all zeros (host bytes). In a subnetted network similar to the one proposed above, the network portion is extended. More specifically, a subnet mask of 255.255.255.0 would subnet the class B address space using its third byte, as would be required to produce 256 LANs each with 256 hosts. By performing an AND of the packet IP address and the subnet mask, the router can get rid of the host part of the number and lookup the result in its table for further redirection if necessary. Since subnet masks are used on a bit-by-bit basis, masks like 255.255.240.0 (4 bits of subnet and 12 bits of host) are perfectly normal.

One thing that is important to keep in mind is that a subnetted network cannot be split into isolated portions. As a result, all subnets must be contiguous, since routing information cannot be passed to non-members. Within a network, all subnets must be able to reach all other subnets without passing traffic through other networks.

2.3.3 Three Bears Problem

Theoretically, under the classfull model presented earlier, 4 billion different IP addresses are possible. At first sight, this might seem to be more than adequate for the 350 million hosts currently on the Internet. However, the practice of organizing the address space by classes wastes millions of them. For most organizations, a class A network, with 16 million addresses is too big, and a class C network, with 256 addresses is too small. A class B network, with 65536 addresses is just right. Inspire by *Goldilocks and the Three Bears* this situation is referred to as the three bears problem in Internet folklore.

As a result of this, an increasing number of class C networks were used and this resulted in an explosion in the size of routing tables, which further complicated the situation under the classfull model.

Finally, along with the two concerns mentioned above, growing IP address demand also generated a need to convert the IP addressing allocation process from a central registry. To deal with all of these issues, classless Inter-Domain Routing was introduced.

2.3.4 Classless Inter-Domain Routing

2.3.4.1 CIDR Basics

CIDR is a replacement for the process of assigning class A, B and C addresses (as discussed earlier) with a generalized network prefix. Instead of being limited to network identifiers (or prefixes) of 8, 16 or 24 bits, CIDR currently uses prefixes anywhere from 13 to 27 buts. Thus blocks of addresses can be assigned to networks as small as 32 hosts or to those with over 500000 hosts. This allows for address assignments that much more closely fit an organization's specific needs.

A CIDR address includes:

- The standard 32-bit IP address
- Information on how many bits are used for the network prefix

This leads to addresses of the form a.b.c.d/m. For example, we may have the CIDR address 201.14.02.48/25. Here, the "/25" indicates that the first 25 bits are used to identify the unique network.

As a result of these developments, the definition of subnetting has been updated to the subdivision of a CIDR block into smaller CIDR blocks. It is important to mention at this point that the prefix boundary for CIDR addresses is not constrained to be greater than the networks natural mask. Networks exist where the prefix boundary contains fewer bits than the network's natural mask and these are called **supernets**. For example, a class C network 198.32.1.0 has a **natural mask** (a mask created by the very definition of the network and host proportions of each class, i.e., Class A has a natural mask of 255.0.0.0, Class B a natural mask of 255.255.0.0 and so on) of 255.255.255.0. The representation 198.32.0.0/16 on the other hand has a shorter mask than the natural mask and is hence a supernet under our definition.

2.3.4.2 Hierarchical Routing Aggregation to Minimize Routing Table Entries

The CIDR scheme (in particular the possibility of creating arbitrary supernets as mentioned in the previous section) enables **route aggregation**, which is the ability to merge multiple prefixes which have some number of leading bits in common. Alternatively, aggregation can also be explained as the situation whereby a single high-level route entry represents many lower-level routes in global routing tables. As an example, consider the scenario where ISP1 has been allocated the CIDR block 204.71.0.0/16. This is further sub-allocated by ISP1 to other customers as shown below:

204.71.1.0/24 allocated to SmallCustomer1 204.71.2.0/24 allocated to SmallCustomer2 204.71.128.0/22 allocated to MediumCustomer 204.71.136.0/20 allocated to LargeCustomer

While exchanging information with other ISPs, ISP1 only needs to announce the single prefix 204.71.0.0/16 and not each one of the individual prefixes used by its customers.

This scheme results in a more organized structure similar to that of telephone networks. A high-level backbone network node only looks at the highest bits and then routes the packet to the specific backbone node responsible for these. This then looks at lower bits and routes the packet to subtending nodes responsible for them. By applying this technique recursively, packets eventually reach their destination. Overall CIDR results in a new, more hierarchical Internet architecture, where each domain takes its IP addresses from a higher level. Registry services can allocate addresses to ISPs, who in turn, sub-allocate it to their customers and so on. In addition to the accompanying reduction in the size of the routing tables, this approach also leads to a system where rather than seeking address assignments directly from InterNIC, it is now possible to rent them out in a decentralized manner.

2.3.4.3 Most Specific Match Routing Rule

Routing to all destinations using the CIDR addressing technique is always done on a longest match basis. A router that has to decide between two different length prefixes of the same network will always follow the longer mask. For example, a router that has the following two entries in its routing table:

198.32.1.0/24 via path 1 198.32.0.0/16 via path 2

will try to match the host 198.32.1.1 with the destination having the longest prefix and consequently deliver the traffic via path 1. In other words, "more specific" networks (in the sense that they are a subset of an aggregate or a CIDR block) are preferred as they give more information about the location of a network.

2.3.4.4 Problems Associated with Aggregation

Aggregation, if not properly applied, could result in the emergence of **black holes**. These occur when traffic reaches and stops at a machine that is not its intended destination, but from which it cannot be forwarded. Such situations may arise, for example, if a customer is connected to a single provider and has an IP address space totally different from it. This could be the result of the customer changing providers and keeping the address previously assigned to him or her. Usually in such situations customers are encouraged or forced to renumber, but if this does not occur, then the new provider cannot aggregate the address and moreover, the old provider cannot aggregate as efficiently as it once did, because a hole has been punched into its address space. The overall effect of using addresses from outside the provider's address space is that more routes must be installed in the global routing tables. Another situation that may lead to similar results is the scenario where customers, in order to prevent being at the whims and fancies of any particular ISP, are connected to multiple providers (i.e., they are **multihomed**) but have IPs assigned by only one of these. Often it is impossible for the other ISPs to aggregate the IPs of these customers as a result of the fact that one or more of the intermediate addresses belongs to some client who is not multihoming to them and any aggregation would result in a black hole. Consequently, ISPs have to list each of these IP address ranges that they have in common with another ISP on top of their own address space. These are often more specific than the address for the CIDR block of the ISP that originally assigned them, and hence the traffic is primarily routed through ISPs that did not issue the IP.

A solution to this is for multihomed clients to acquire IPs from each of the ISPs they are connected to, but this leads to the loss of routes to multihomed organization. If one of the ISPs has problems, then traffic to the multihomed host on the IP obtained from that ISP is lost as it is not advertised anywhere else, which, to a large extent, nullifies the very purpose of multihoming. An alternative approach requires taking an address that is totally different from the address space of all the ISPs a customer is connected to. However, in this case, the drawback is that all routers in the Internet must have a specific route to these newly introduced addresses, leading to very large overall routing tables.

2.3.5 Problems Associated with IPv4 - Scalability, Security and Quality of Service

"*I think there is a world market for maybe five computers*" - Thomas Watson, 1943
"*640K should be enough for anybody*" - Bill Gates, 1981
"*32 bits ought to be enough address space*" - Vinton Cerf, 1977

With the unprecedented and largely unexpected growth of the Internet, it has become increasingly obvious that the days of IPv4 (the current implementation of the IP protocol discussed in earlier sections) are numbered. The problems of **address space depletion** and **control by a central registry** are still very much there and the advent of CIDR has only served as a temporary patch to them. Currently registries are unable to honour large IP block requests and it is expected that by the year 2008 IPv4 addresses will be exhausted. The latter stems from the increase in 3G mobile communication, the demand imposed by emerging nations that missed out on the IPv4 "gold rush" and the onset of home networking. Coupled with this a dramatic increase in the size of the router tables (which stand around 60000 entries these days) due to an **inherent lack of any network or routing considerations**. Static memory concerns aside, this also leads to a degradation of performance as the current algorithms often do not scale linearly. IPv4 also suffers from the lack of any in-built mechanism to deal with the service requirements to be met by the network while transporting a flow. This can also be stated as the **absence of Quality of Service** or QoS and can be largely understood as the limitation to move beyond the "best effort" datagram service available today, which focuses only on a single arbitrary metric: the administrative weight or hop count. Finally, IPv4 is also found to be wanting in the **absence of an intrinsic security mechanism**, which results in expensive host and application modifications to protect against a number of different attacks. In the absence of these, loss of privacy, loss of data integrity, identity spoofing and denial-of-service may occur.

Perhaps the most compelling problem facing the IP Internet is address depletion. Long-term and short-term solutions to this problem are being developed. The short-term solution is CIDR (Classless InterDomain Routing). The long-term solutions consist of various proposals for new internet protocols with larger addresses. It is possible, however, that CIDR will not be adequate to maintain the Internet until the long-term solutions are in place. Consequently another solution has been proposed: some form of address reuse, either complementing CIDR or making it unnecessary. The address reuse solution is to place Network Address Translators (NAT)[8] at the borders of stub domains. Each NAT box has a table consisting of pairs of local IP addresses and globally unique addresses. The IP addresses inside the stub domain are not globally unique. They are reused in other domains, thus solving the address depletion problem. The globally unique IP addresses are assigned according to current CIDR address allocation schemes. The main advantage of NAT is that it can be installed without changes to routers or hosts. However NATs have numerous disadvantages[9] ranging from their inability to handle arbitrary protocols, to the fact that they represent single points of failure and so reduce overall system reliability.

2.3.6 The IPv6/IPng Solution

To deal with the limitations of IPv4, a new protocol called IPv6 or IPng (IP Next Generation) has now been proposed. Presently there exists a IPv6 infrastructure on the Internet, but DNS has not yet been implemented.

IPv6 is fitted with a number of features. For starters, it provides an increased **address space of 128 bits**, allowing globally unique IPs for all devices. In addition, it also offers more efficient routing by virtue of a **hierarchical structure** which leads to aggregated address allocation from the outset. However, multihomed clients represent a breakdown in the hierarchy and presently it is not uncommon for corporations to multihome. IPv6 also reserves fields in its packet headers to handle QoS and security. In order to deal with the issue of QoS the IPv6 header has two fields for this purpose; a **20-bit Flow label and an 8-bit Traffic Class indicator**. Finally, security concerns are addressed by the introduction of **IPSec**, which makes use of a new set of headers that are added to secure the payload of the IP packet. These include the Authentication Header (AH) and the Encapsulating Security Payload (ESP).

2.3.7 The IPv4 to IPv6 Transition

The most important idea behind the transition from IPv4 to IPv6 is that the Internet is too big and decentralized to have a “flag day” - one specified day on which every host and router is upgraded from IPv4 to IPv6. Thus, IPv6 needs to be deployed incrementally in such a way that hosts and routers that only understand IPv4 can continue to function for as long as possible. This process is not expected to be complete before the year 2010. Furthermore, the emergence of NATs has delayed the widespread adoption of IPv6. Even though some people may hate NATs religiously, their use provides the primary functionality of IPv6: more host addresses. NATs have thereby effectively reduced the pressure to implement a more elegant long term solution to the address depletion problem. The increased address space NATs provide has made people less willing to adopt a new basic network protocol. It seems that IPv4 will be around for some time.

[8]K. Egevang, P. Francis, “The IP Network Address Translator (NAT)” - RFC1631

[9]Keith Moore “Things that NATs break,” http://www.cs.utk.edu/ moore/what-nats-break.html

After the transition, however, IPv4 nodes should ideally be able to talk to other IPv4 nodes and some set of other IPv6-capabale nodes indefinitely. Also, IPv6 hosts should be capable of communicating with other IPv6 nodes, even when some of the infrastructure between them may only support IPv4. Two major mechanisms have been defined to help this transition and these are discussed in the following sub-sections.

2.3.7.1 Dual-Stack Operation

The idea behind this is fairly straight-forward. IPv6 nodes run both IPv6 and IPv4 and use the version field in the header to decide which stack should process an arriving packet.

2.3.7.2 Tunnelling

This is the technique used in a variety of situations in which an IP packet is sent as the payload of another IP packet; that is, a new IP header is attached in front of the header of an existing IP packet. For IPv6 transition, tunneling is used to send an IPv6 packet over a piece of the network that only understands IPv4. This means that the IPv6 packet is encapsulated within an IPv4 header that has the address of the tunnel endpoint in its header, is transmitted across the IPv4-only piece of the network and is then decapsulated at the endpoint. The endpoint could be either a router or a host. In either case, it must be IPv6 capable to be able to process the IPv6 packet after decapsulation.

2.4 Routing

The Internet consists of millions of computers any of which can theoretically communicate with any other. However, not all computers are physically linked to each other and so information is communicated from one point to the other by using a store and forward paradigm. Packets of data are transmitted between computers physically linked together and are routed through the network to their final destinations. Various routing protocols define the rules for forwarding packets. This section discusses these protocols.

2.4.1 Topological Model

The Internet's wide area routing architecture is divided into a number of arbitrarily connected Autonomous Systems (AS's). An AS is a set of routers and hosts under a single technical administration (usually a single commercial entity), using one or more interior gateway protocols (IGPs) and common metrics to route packets within the AS and using some exterior gateway protocol (EGP) to route packets to and from other AS's. The use of the term Autonomous System here stresses the fact that, even when multiple IGPs and metrics are used, the administration of an AS appears to other AS's to have a single coherent interior routing plan and presents a consistent picture of which destinations are reachable through it.

Each AS is uniquely identified by a 16-bit integer assigned by InterNIC and used by EGPs to implement policy routing and avoid top-level routing loops. An AS can be thought of as a collection of CIDR IP address prefixes under common technical management. For example, the CIDR block from 208.130.28.0 to 208.130.31.255, as well as network 201.64.75.0 might be AS 2934. The MIT network is AS 3.

Much of the traffic carried within an AS either originates or terminates at that AS (i.e., either the source IP address or the destination IP address of the IP packet identifies a host internal to that AS). Traffic that fits this description is called "local traffic". Traffic that does not fit this description is called "transit traffic". Based on how a particular AS deals with transit traffic, the AS may be placed into one of the following categories:

- A **Stub AS** is only connected to one other AS. For routing purposes, it could be regarded as a simple extension of the other AS. In fact, most networks with a single Internet connection don't have a unique AS number assigned, and their network addresses are treated as part of the parent AS.
- A **Transit AS** has connections to more than one other AS and allows itself to be used as a conduit for traffic (transit traffic) between other AS's. Most large Internet Service Providers are transit AS's.
- A **Multihomed AS** has connections to more than one other AS, but does not allow transit traffic to pass, though its interior hosts may route traffic through multiple AS's. This is the typical configuration for a large corporate network with multiple redundant Internet connections, but which does not wish to pass traffic for others.

2.4.2 Routing Protocol Types

For any given network that can be represented as a connected undirected graph (where edges represent direct connections between network entities) there are two general classes of protocols that can be used to route packets between nodes.

2.4.2.1 Distance-Vector Protocols

This type of routing protocol requires that each router simply inform its neighbours of its routing table. Routers send out periodic routing updates which include a vector of distances to other nodes. For each network path, the receiving routers pick the neighbour advertising the lowest cost, then add this entry into its routing table. As a result, each node has a (policy determined) shortest path tree giving the paths to other reachable nodes.

DV protocol algorithms only make use of information from neighbouring nodes. These protocols tend to be decentralized and iterative, based on algorithms such as Bellman-Ford's algorithm for shortest paths. DV protocols also tend to be relatively scaleable. The basic DV protocol is called the Routing Information Protocol (RIP). The Border Gateway Protocol (BGP) uses an essentially DV algorithm, but with several added twists. It is discussed in more detail later.

Routing based on DV protocols can cause a lot of problems when links go up and down or when nodes advertise incorrect information. This could result in infinite loops and can also de-synchronize the network. Convergence is generally slow in DV protocols. In fact it may take minutes or even hours for the topology of a network to be synchronized regardless of the computational resources available to each node.

2.4.2.2 Linke-State Protocols (LS)

Link-State protocols require each router to maintain at least a partial map of the network. When a network link changes state (up to down, or vice versa), a notification, called a **link state advertisement** is **flooded** throughout the network. All the routers note the change, and recompute their routes accordingly. The Open Shortest Path First (OSPF) protocol is an example of a LS protocol.

Since every node has a good view of the network graph, and updates only require the addition or removal of nodes and edges, LS protocols converge faster to a stable state than DV protocols. They are also more reliable, easier to debug and less bandwidth-intensive than DV protocols.

On the other hand, LS protocols are generally more complex than DV protocols. They also tend to be more intensive in terms of both memory and local computation. LS protocols do not scale as well as DV protocols and so protocols such as OSPF are normally used to route packets internally within an AS. IGPs are generally concerned with optimizing a given path metric and scalability is not a major concern.

2.4.3 Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) is the de facto interdomain routing protocol used to exchange reachability information between Autonomous Systems in the global Internet. It is a distance-vector protocol that allows each AS to override distance-based metrics with policy-based metrics when choosing best routes.

2.4.3.1 Design Motivations

The design of BGP was motivated by three important needs:

- **Scalability**. The division of the Internet into separate routing domains (the AS's) was done during the time of NSFNET. An important requirement for BGP was to ensure that the Internet routing infrastructure remained scalable as the number of connected networks increased.
- **Policy**. The ability for each AS to implement and enforce various forms of routing policy was an important design goal. Some of the consequences of this were the development of the BGP attribute structure for route announcements, the allowing of route filtering and the prioritization of local preferences.
- **Cooperation under competitive circumstances**. BGP was designed in a large part to handle the transition from the NSFNET to a situation where the backbone infrastructure would no longer be run by a single administrative entity. Routing in the Internet would be handled by a large number of mutually competing ISP's, who would loosely cooperate to provide global connectivity. The routing protocol was therefore

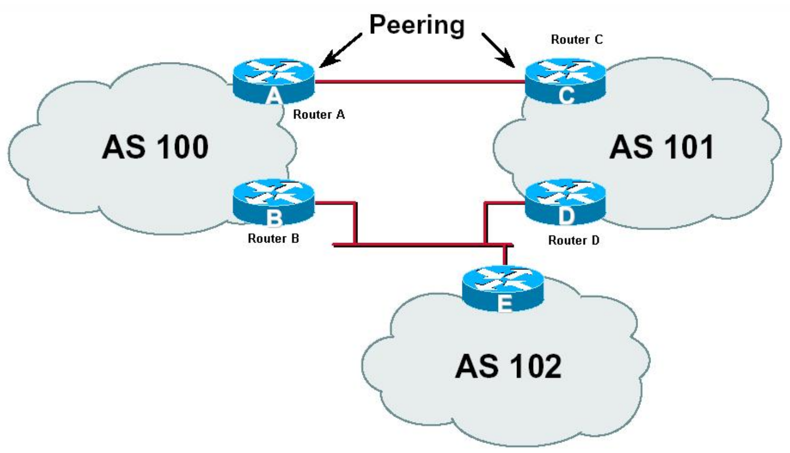

Figure 2.4. Autonomous Systems in BGP

designed to allow AS's to make purely local decisions on how to route packets from among any set of choices.

2.4.3.2 The BGP-4 Protocol

2.4.3.2.1 An Idealized Model

The protocol models the Internet as a collection of Autonomous Systems with peering relations between one another. This forms an undirected graph, where the vertices represent AS's and the edges represent reliable direct connections between AS's. Each vertex maintains a **shortest-path** tree rooted at itself specifying the best routes to other vertices. Routes are selected by each AS based on its own policy and the best routes of its neighbours.

A vertex learns the set of available routes through the route announcements of its neighbours. Vertices send out such announcements whenever their routing tables change. Announcements contain, among other things, information about the AS's that need to be passed through in the route. This gives the distance-vector or path-vector of the route.

Whenever a vertex has more than one route to a given destination, it selects the best one out of them. It first selects on the basis of the local preferences at the vertex, it then picks the route with the shortest distance-vector. Ties are broken arbitrarily.

Route announcements undergo transformations as they pass from one vertex to the next. In particular, whenever a route is passed on, the vertex appends its identifier to the path-vector of the route. Loops can be prevented in this way, since no vertex accepts route announcement in whose path-vector the vertex itself appears.

As announcements propagate through the network, each vertex iteratively improves its view of the available routes. Eventually, the system is expected to converge to a stable state. In this final state each vertex knows the best route to every other vertex. Because of the presence **local preferences** the definition of what makes a **best** path may vary from node to node.

2.4.3.2.2 The Protocol

Two systems (usually two border routers in different AS's) form a reliable TCP connection between one another and exchange messages to open and confirm the connection parameters. The initial data flow is the entire BGP routing table. Later, incremental updates are sent as the routing tables change. There are two kinds of such updates. The first are announcements, which are changes to existing routes or new routes. The other are withdrawals, which inform the peer that named routes are no longer available.

A BGP announcements contain a set of prefixes each of which has a set of attributes associated with it. The as_path attribute is a vector that lists all the AS's (in reverse order) that this prefix's route announcement has been through. Upon crossing an AS boundary, the first router prepends the unique identifier of its own AS to this vector before propagating the announcement further. Routers prevent loops by discarding messages if their AS's are already present in this vector.

A router uses import policies to transform incoming route updates. These import policies include denying an update, or permitting an update and assigning a local preference to indicate how favourable the path is. Similarly, export policies allow a router to determine whether to send its best route to a neighbour and, if it does send the route, what hint it should send to its neighbour on using the route.

BGP routers use the fields in route announcements to determine the routes to use in its routing table. Generally routes learnt externally (by eBGP) are preferred to those learnt internally (by iBGP). Local preferences, representing any arbitrary local policy, are given the highest priority in the selection. The length of the as_path vector is the second most important criteria. This may not be the shortest distance since it is only meant to count the number of AS's that need to be routed through. Routers may also pad the as_path by repeating AS identifiers to make certain routes less desirable to other AS's. Finally, ties are broken arbitrarily. Because of these factors and some vendor specific router implementation decisions, it is almost impossible to determine the best routes just from connectivity relations.

BGP uses TCP, which provides reliable in-order delivery, and so it does not require a periodic refresh of the entire routing table. A BGP speaker must, therefore, retain the current version of the entire BGP routing tables of all of its peers for the duration of the connection. *KeepAlive* messages are sent periodically in both directions to ensure that the BGP session is still running. In the absence of such messages, a BGP session is closed and all the routes learnt from that session are removed from the routing tables of the respective peers.

Notification messages are sent in response to errors or special conditions. If a connection encounters an error condition, a notification message is sent and the connection is closed. However, BGP wasn't designed for rapid fault detection and recovery, so these mechanisms are generally not particularly useful over short time scales.

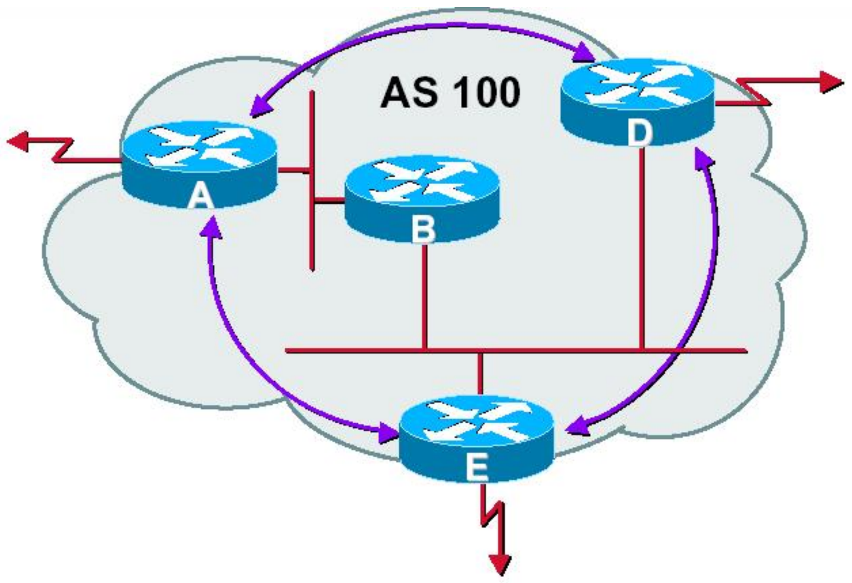

Figure 2.5. iBGP Sessions

2.4.3.2.3 eBGP: Inter-AS Routing

External BGP (eBGP) is the standard mode in which BGP is used. Routers on the borders of a given AS are connected to routers in other AS's using BGP sessions. These routers provide the only entry/exit points from their AS. They implement route filtering rules and exchange a subset of their routes using BGP with routers in the other ASs across the BGP sessions.

2.4.3.2.4 iBGP: Intra-AS Consistency

If a particular AS has multiple BGP speakers and is providing transit service for other ASs (as is generally the case - at least for a small number of routes), then care must be taken to ensure a consistent view of routing within the AS.

In general each AS will have more than one border router that will participate in eBGP sessions with neighbouring AS's. During this process, each eBGP router will obtain information about some subset of the routes the entire AS knows about. Since one of the goals of BGP is to allow each AS to be treated as one abstract monolithic entity, it is imperative that each router in an AS have a complete view of the routes available to the eBGP entire AS.

Consistency is achieved by allowing the border routers to exchange information using iBGP. eBGP routers are connected together within the AS and maintain iBGP sessions with each other. These sessions allow each router to update its tables based on the information available to the others.

It is important to note that iBGP is not an IGP like RIP or OSPF. In fact, the messages sent during an iBGP session are routed within the AS by whichever IGP is operational. iBGP sessions merely provide a means by which routers inside an AS can use the same protocol (BGP) to exchange information for completeness.

2.4.3.2.5 Question of Convergence

Informally a BGP system can be represented by a state S = <G, Policy(G), S0>, comprising an AS graph G = (V, E), containing import and export policies for every vj in V and with an initial state S0 = (c0,1, c0,2,c0,n) where c0,j is the destination originated by vj in state S0. If vj is activated then it gets route announcements from its immediate neighbors and selects its best routes. The state S is said to be final if on the activation of any vj, the system stays in state S. An activation sequence of the system is simply a sequence giving the vertices vj and the order in which they were activated.

A BGP system is said to be convergent if it ends up in a final state independent of the activation sequence. The system is said to be solvable if it has a final state. Unlike pure distance-vector based protocols, such as RIP, BGP is not guaranteed to converge.

BGP policies are currently implemented locally with little or no global knowledge. While most routing policy conflicts are manageable, there is always the possibility that they could cause the protocol to diverge. The routing policies of each individual AS may be locally reasonable but there is no guarantee that the interaction of independently defined policies will be globally reasonable. In theory, a collection of locally well-configured policies can still give rise to global routing anomalies. It is, therefore, possible for a collection of AS's to exchange messages forever without converging to a stable set of routes.

Such BGP divergence could introduce a large amount of instability into the global routing system. The fluctuation of network topology can have a direct impact on endtoend performance. A network that has not yet reached convergence may drop packets, or deliver packets out of order. In extreme cases, a lack of convergence could result in the blackholing of packets (with ASs advertising routes which they essentially don't have - resulting in packets sent along those routes being dropped). Consequently, in order for a BGP based network to work efficiently, it should be convergent. Unfortunately, there are many BGP systems which are not convergent.

2.4.3.2.6 The Bad Gadget System

This section considers an example of an unsolvable BGP system, called 'Bad Gadget'. No execution of the BGP protocol can possibly arrive at a stable routing in this system. The 'Bad Gadget' network is presented in figure 2.6, where each node represents an AS and edges represent connections between the ASs.

Suppose that there is a single destination with d originated by AS 0. The policy is implemented such that each AS prefers the counterclockwise route of length 2 over all other routes to the 0 That is, AS 3 prefers the route 3 - 2 - 0 over 3 - 0.

It can be shown that this system is divergent. The proof is presented in a paper by

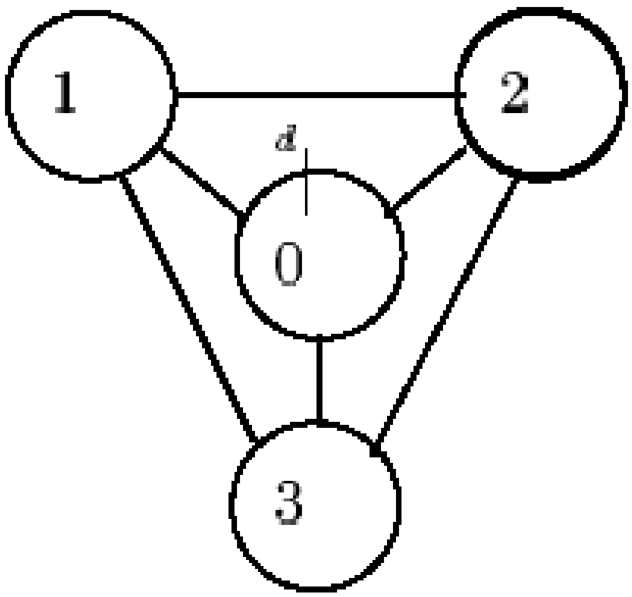

Figure 2.6. Bad Gadget Network

Griffen and Wilfong[10] . An overview is presented here.

An example is first presented below to provide some intuition. Figure 2.7 follows the system through a sequence of four activations. Black nodes represent a node being activated, i.e. advertising its routes. Arrowheads indicate routes.

In (1) no advertisements have been made and no nodes know of any routes. In (2) node 0 advertises all the direct routes to it and each node picks its direct connection to 0 as the best route. In (3) node 1 advertises its routes, which causes node 2 to switch its route to the more preferred counter-clockwise one. In (4) node 3 advertises its route, causing node 1 to switch. Finally, in (5) node 0 advertises again and node 2 switches back. If we follow this reasoning further, we can see evidence that no stable routes are being set up. The formal proof presented in the paper verifies our intuition. An overview of the proof follows.

For this system to have a solution, it must have a final state. It is easy to see for single destination systems that in a final state the graph induced by the as_path at every vertex to a destination is a tree rooted at the destination, and that this final state is reachable by activating all the nodes of the tree in breadth-first order (this is from the proof in the paper). Using this fact, it can be shown that this system is divergent.

Only the sixteen spanning trees rooted at AS 0 need to be considered. Any tree that does not span this graph cannot be a solution because each of AS 1, 2 and 3 have a direct route to d. Furthermore, since this system is symmetric, only the six cases presented in figure 2.8 are relevant.

[10]This example has been analyzed in detail in the following papers: [1] Timothy Griffin and Gordon T. Wilfong, "An Analysis of BGP Convergence Properties" SIGCOMM [2] K. Varadhan, R. Govindan & D. Estrin, "Persistent Route Oscillations in Inter-Domain Routing" ISI TR 96-631

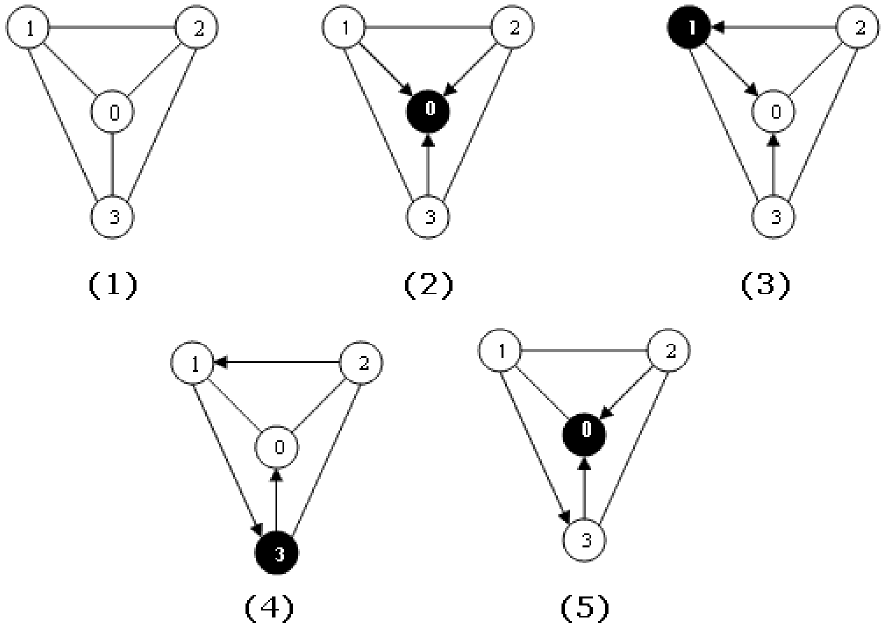

Figure 2.7. Bad Gadget Activation Example

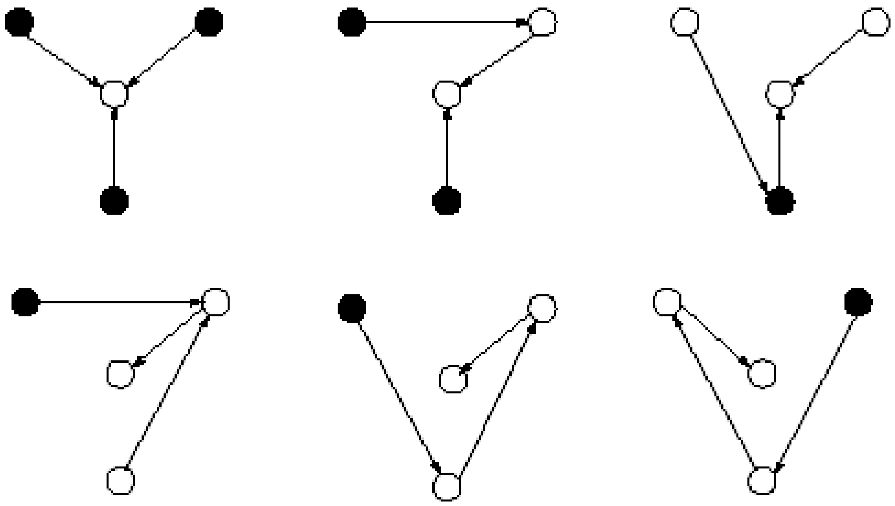

Figure 2.8. Possible Routing Trees for Bad Gadget

In figure 2.8 those AS's that will change their selection of the best route to d are marked with a solid circle. Each marked AS will either pick the counterclockwise route of length 2, or revert to the direct route to d. It is easy to see that the system has no solution.

This divergent system may seem like a contrived example and the presence of real world BGP divergence may be questioned. The next section outlines some aspects of BGP instability on the Internet.

2.4.3.2.7 Observations of Instability

Does BGP diverge in practice? There are horror stories of networks accidentally setting themselves up as sinks for all the traffic and there is some evidence of chronic oscillations. Frequent occurrences of delayed convergence, as high as 50 minutes, have also been observed. Even so, BGP has generally shown to be convergent on the Internet.

Internet routers may experience severe CPU load and memory problems at heavy levels of routing instability. In 1997 for example, many of the commonly deployed Internet routers were based on a relatively light Motorola 68000 series processor. Heavy instability frequently led to problems in memory consumption and queuing delay of packet processing. Frequently, the delays in processing were so severe that routers delayed routing *KeepAlive* packets and were subsequently flagged as down, or unreachable by other routers.

Experience with the NSFNet and widearea backbones has demonstrated that a router which fails under heavy routing instability can instigate a "route flap storm." In this mode of pathological oscillation, overloaded routers are marked as unreachable by BGP peers as they fail to maintain the required interval of *KeepAlive* transmissions. As routers are marked as unreachable, peer routers will choose alternative paths for destinations previously reachable though the "down" router and will transmit updates reflecting the change in topology to each of their peers. In turn, after recovering from transient CPU problems, the "down" router will attempt to reinitiate a BGP peering session with each of its peer routers, generating large state dump transmissions. This increased load will cause yet more routers to fail and initiate a storm that begins affecting ever larger sections of the Internet. Several route flap storms in the past have caused extended outages for several million network customers. The later generation of routers from several vendors provided a mechanism in which BGP traffic was given a higher priority and *KeepAlive* messages persisted even under heavy instability.

Most of this Internet routing instability was a result of software bugs and vendor specific implementation decisions. However, after ISPs deployed updated routers, this instability dropped by several orders of magnitude. Heuristics such as route announcement dampening have also improved the convergent properties of BGP. So although in theory it is possible for BGP to diverge, it has shown empirically that it eventually converges.

Nonetheless, there are frequent and numerous occurrences of delayed convergence. In a study[11], Labovitz et al injected BGP faults (announcements/withdraws) of varied prefix and AS-path lengths into topologically and geographically diverse ISP peering sessions. They noted delays as large as 50 minutes in their study. In fact, routing convergence times after failovers were on the order of tens of minutes.

[11] "Delayed Internet Routing Convergence" C. Labovitz, A. Ahuja, A. Bose & F. Jahanian, Proceedings of SIGCOMM 200

This latency can be understood by looking at the way BGP explores paths. In addition to various vendor specific anomalies, the main reason for long convergence is that path vector protocols consider multiple paths of a given length as opposed to distance vector protocols that consider only one path of a given length. In Labovitz et al they construct an example where every loop free path in the complete mesh is considered - there are an exponential number of such paths. It has been shown that there exists a possible ordering of messages such that BGP will explore all possible AS-paths of all possible lengths. BGP is, therefore, O(N!), where N is the number of default-free BGP speakers in a complete graph with default policy. Furthermore, there is no global knowledge in path vector protocols and thus there are fewer optimization heuristics. It is, therefore, not surprising that convergence usually takes a long time.

2.4.3.2.8 Reachability in an AS

Given a network of Autonomous Systems linked by BGP and a destination which is reachable from one vertex (x), it is sometimes useful to know whether that destination is reachable from some other vertex (y). Reachability is a property that holds if there exists a final state of the system in which there is a route from y to the destination. The determination of whether reachability exists in general in a given BGP system is NP-Complete. This can be demonstrated by a reduction from 3-SAT and the proof is presented in the Griffen and Wilfong paper.

The Routing Arbitration Database (RADB) is intended to provide users of the Internet with inter-AS reachability information. The Internet Routing Registry (IRR of which RADB is a part) maintains individual ISPs' routing information in several public repositories in an attempt to coordinate global routing policy. The IRR's routing policy database expresses routing information at various levels (e.g., individual address prefixes or ASs, etc.) Routes are registered with the RADB voluntarily. Registering a route in the RADB implies that the registrant is able to provide reachability to that destination, i.e., the destination is reachable via the registrants network infrastructure.

In the recent past, there has been a significant interest in studying and modeling the Internet's topology, especially at the level of autonomous systems (ASs). To date, these activities have consisted mainly of analyzing and modeling measurements (mostly from databases such as RADB) to infer the Internet's AS connectivity graph and describe its properties, explaining the origins and causes of some of the observed surprising features, and building simulators that produce graph structures that match those of the measured AS connectivity graphs well. They can also be used to provide a realistic context for fault isolation. In a recent paper by Chang et al[12] , it was noted that these databases show us that a significant number of existing AS connections remain hidden from most BGP routing tables.

Knowledge about the connectivity of the Internet can provide us with a better framework to understand the real-world behaviour of BGP. Unfortunately, much of the information provided to the RADB may be filtered selectively by organizations. For instance, it is known that an increasing number of ISPs rely on the IRR to filter route announcements at border routers (the RIPE database of the IRR is actively used by most ISPs in Europe). Knowledge

[12]H. Chang, R. Govindan, S. Jamin et al, "On Inferring AS level Connectivity from BGP Routing Tables"

of this behaviour may influence what routes are registered with the RADB.

2.4.3.2.9 Other BGP Complexity Results

Many other properties of the BGP protocol have been studied, both mathematically and empirically. Some other BGP complexity results are listed below. A few of these properties are presented below as articulated in the Wilfong and Griffen paper.

For a BGP system S, the evaluation graph, Eval(S), is defined to be the directed graph, where for each reachable state s, there is a vertex labeled by s in Eval(S), and if s and s' are reachable states and s transitions to s' under the activation of some non-empty vertex set A, then there is a directed edge from s to s' labelled by A in Eval(S). A state s in Eval(S) is final if for all vertices (v) in the set of vertices of the BGP system, there exists an edge between s and s labelled v.

- **Asymmetry**

  Given a BGP system S, AS v in S, AS w in S, a destination d1 originated by w, a destination d2 originated by v, does there exist a final state s in Eval(S) in which the route from v to d1 is not the reverse of the route from w to d2? Asymmetry is NP-complete.

- **Solvability/Unsolvability**

  Given a system S, does there exist a final state in Eval(S)? Or can we ascertain that there is no final state in Eval(S)? Solvability and unsolvability are NP-complete.

  Notice that an unsolvable system can never converge to a stable state, since none exists. Operationally, this means that for unsolvable systems, BGP is likely to go into a "protocol oscillation" that will never terminate.

- **Solvability/Unsolvability (SD)**

  Given a system S having only a single destination d originated by some AS w, does there exist a final state in Eval(S)? Or can we ascertain thatthere is no such final state in Eval(S)? These problems are NP-hard.

- **Trapped**

  Given a system S, does Eval(S) contain a trap? We can formally define a trap to be a subgraph of Eval(S) that (1) does not contain a final state, and (2) has no arcs directed out of this subgraph. This problem is NP-hard.

- **K-Robust**

  Given a solvable system S having only a single destination d originated by some AS w, will S remain solvable under any possible failure of k links? This is NP-hard.

- **Uniqueness**

  Given a system S, does there exist exactly one final state in Eval(S)? This is NP-hard.

- **Uniqueness**

  Given a system S having only a single destination d originated by some AS w, does there exist exactly one final state in Eval(S)?

- **Multiplicity**

  Given a system S, do there exist multiple possible final states in Eval(S)? This is NP-hard.

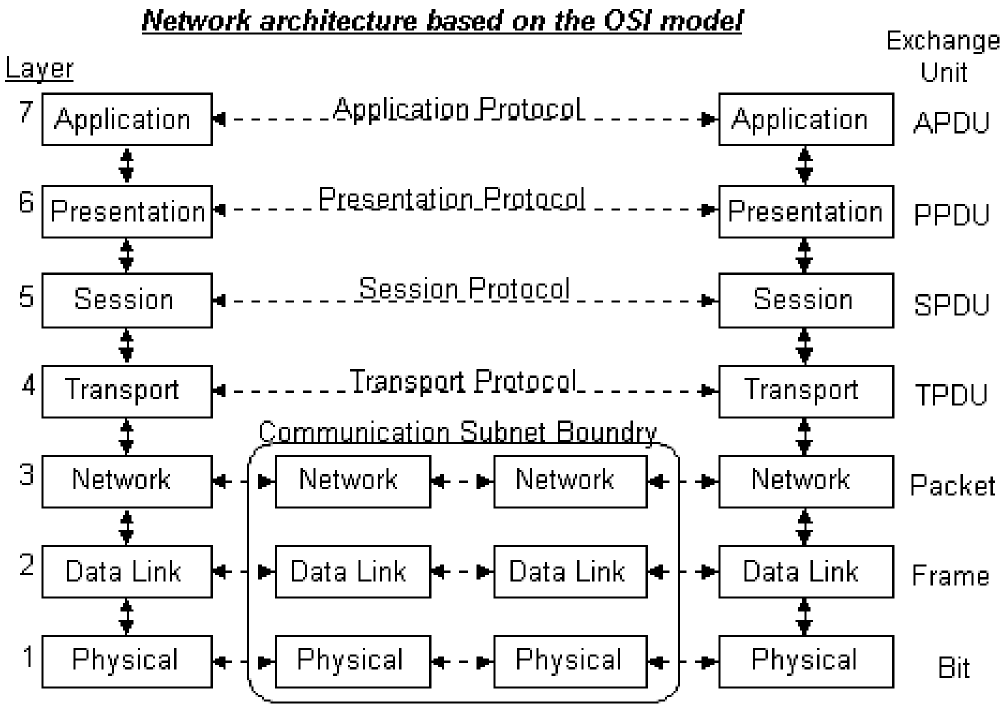

Figure A.1. OSI Reference Model

APDU - Application Protocol Data Unit
PPDU - Presentation Protocol Data Unit
SPDU - Session Protocol Data Unit
TPDU - Transfer Protocol Data Unit

Appendix A

The OSI model consists of:

1. **Physical Layer**: The physical later is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit, it is received by the other side as a 1 bit, not as a 0 bit. Typical questions here ar e how many volts should be used to represent a 1 and how many for a 0, how many microseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established and how it is torn down when both sides are finished, and how many pins the network connector has and what each pin is used for. The design issues here deal largely with mechanical, electrical, and procedural interfaces, and the physical transmission medium, which lies below the physical layer. Physical layer design can properly be considered to be within the domain of the electrical engineer.

   Example: ISDN, CAT 1, ADSL, ATM, FDDI, Coaxial Cables

2. **Data Link Layer**: The main task of the data link layer is to take a raw transmission facility and transform it into a line that appears free of transmission errors in the network layer. It accomplishes this task by having the sender break the input data up into data frames (typically a few hundred bytes), transmit the frames sequentially, and process the acknowledgment frames sent back by the receiver. Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning of structure, it is up to the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame. If there is a chance that these bit patterns might occur in the data, special care must be taken to avoid confusion.

   Example: SLIP, PPP, 802.2 SNAP, Ethernet II

3. **Network Layer**: The network layer is concerned with controlling the operation of the subnet. A key design issue is determining how packets are routed from source to destination. Routes could be based on static tables that are "wired into" the network and rarely changed. They could also be determined at the start of each conversation, for example a terminal session. Finally, they could be highly dynamic, being determined anew for each packet, to reflect the current network load. If too many packets are present in the subnet at the same time, they will get in each other's way, forming bottlenecks. The control of such congestion also belongs to the network layer.

   Example: IPv4, IPv6

4. **Transport Layer**: The basic function of the transport layer, is to accept data from the session layer, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently, and in a way that isolates the session layer from the inevitable changes in the hardware technology. Under normal conditions, the transport layer creates a distinct network connection for each transport connection required by the session layer. If the transport connection requires a high throughput, however, the transport layer might create multiple network connections, dividing the data among the network connections to improve throughput. On the other hand, if creating or maintaining a network connection is expensive, the transport layer might multiplex several transport connections onto the same network connection to reduce the cost. In all cases, the transport layer is required to make the multiplexing transparent to the session layer. The transport layer also determines what type of service to provide to the session layer, and ultimately, the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages in the order in which they were sent. However, other possible kinds of transport, service and transport isolated messages with no guarantee about the order of delivery, and broadcasting of messages to multiple destinations. The type of service is determined when the connection is established. The transport layer is a true source-to-destination or end-to-end layer. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages.

Example: TCP,UDP

5. **Session Layer**: The session layer allows users on different machines to establish sessions between them. A session allows ordinary data transport, as does the transport layer, but it also provides some enhanced services useful in a some applications. A session might be used to allow a user to log into a remote time-sharing system or to transfer a file between two machines. One of the services of the session layer is to manage dialogue control. Sessions can allow traffic to go in both directions at the same time, or in only one direction at a time. If traffic can only go one way at a time, the session layer can help keep track of whose turn it is. A related session service is token management. For some protocols, it is essential that both sides do not attempt the same operation at the same time. To manage these activities, the session layer provides tokens that can be exchanged. Only the side holding the token may perform the critical operation. Another session service is synchronization. Consider the problems that might occur when trying to do a two-hour file transfer between two machines on a network with a 1 hour mean time between crashes. After each transfer was aborted, the whole transfer would have to start over again, and would probably fail again with the next network crash. To eliminate this problem, the session layer provides a way to insert checkpoints into the data stream, so that after a crash, only the data after the last checkpoint has to be repeated.

   Example: RPC Portmapper

6. **Presentation Layer**: The presentation layer performs certain functions that are requested sufficiently often to warrant finding a general solution for them, rather than letting each user solve the problems. In particular, unlike all the lower layers, which are just interested in moving bits reliably from here to there, the presentation layer is concerned with the syntax and semantics of the information transmitted. A typical example of a presentation service is encoding data in a standard, agreed upon way. Most user programs do not exchange random binary bit strings. They exchange things such as people's names, dates, amounts of money, and invoices. These items are represented as character strings, integers, floating point numbers, and data structures composed of several simpler items. Different computers have different codes for representing character strings, integers and so on. In order to make it possible for computers with different representation to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire". The job of managing these abstract data structures and converting from the representation used inside the computer to the network standard representation is handled by the presentation layer. The presentation layer is also concerned with other aspects of information representation. For example, data compression can be used here to reduce the number of bits that have to be transmitted and cryptography is frequently required for privacy and authentication.

   Example: HTTP, FTP, Telnet, DNS, SNMP, NFS

7. **Application Layer**: The application layer contains a variety of protocols that are commonly needed. For example, there are hundreds of incompatible terminal types in

the world. Consider the plight of a full screen editor that is supposed to work over a network with many different terminal types, each with different screen layouts, escape sequences for inserting and deleting text, moving the cursor, etc. One way to solve this problem is to define an abstract network virtual terminal for which editors and other programs can be written to deal with. To handle each terminal type, a piece of software must be written to map the functions of the network virtual terminal onto the real terminal. For example, when the editor moves the virtual terminal's cursor to the upper left-hand corner of the screen, this software must issue the proper command sequence to the real terminal to get its cursor there too. All the virtual terminal software is in the application layer. Another application layer function is file transfer. Different file systems have different file naming conventions, different ways of representing text lines, and so on. Transferring a file between two different systems requires handling these and other incompatibilities. This work, too, belongs to the application layer, as do electronic mail, remote job entry, directory lookup, and various other general-purpose and special-purpose facilities.

Example: E-mail, Newsgroups, Directory Services, Files Services

References

[1] "Computer Networks", Andrew S. Tanenbaum
[2] "Internet Routing Architectures", Bassam Halabi
[3] "Computer Networks: A Systems Approach", Bruce S. Davie, et al
[4] "OSI Reference Model", http://www.rad.com/networks/1994/osi/intro.htm