The Internet Protocol

Part One : The Foundations

by Shvetima Gulati

July 2000, Editor In Chief's Note. Welcome to the launch of a brand new ACM Crossroads monthly column! The Connector column will cover various computer networking concepts and Internet-specific technologies. Our first columnist is Shvetima Gulati and we look forward to learning about this hot topic each month...so make sure you come back again and again to see what's new! If you would like an email reminder about the release of new articles on the Crossroads website, then simply subscribe to xrds-announce, our announcemenst mailing list.

Basic Concepts

Most network applications are described as either **client-side applications** or **server-side applications**. A Web browser is an example of a client-side application. It receives its data from a remote program known as a Web server application. In general, a server-side application provides the services that a client-side application demands. The term *server* might also refer to the powerful hardware devices on which software-server applications are executed. To clarify this distinction, I will use the term **server machine** for the hardware platform and either server-side application or just server for a software program running on that platform.

A **protocol** is set of rules and conventions used to impose a standardized, structured language for the communication between multiple parties. For example, a protocol might define the order in which information is exchanged between two parties. In fact, a data exchange can *only* take place between two computers using the same protocol.

Transmitting data across computer networks is an arduous task. Network functionality has been decomposed into modules called **layers** to simplify and separate the tasks associated with data transmission. Each layer is a unit of code that performs a small, well-defined set of tasks. A **protocol suite** (or **protocol stack**) is a set of many such layers, and is usually a part of the operating system kernel on machines connected to the Internet.

A protocol stack is organized such that the highest level of abstraction resides at the top layer. For example, the highest layer may deal with streaming audio or video frames, whereas the lowest layer deals with raw voltages or radio signals. Every layer in a stack builds upon the services provided by the layer immediately below it.

The terms protocol and **service** are often confused. A protocol defines the exchange that takes place between identical layers of two hosts. For example, in the TCP/IP stack, the transport layer of one host talks to the transport layer of another host using the TCP protocol. In contrast, a service is the set of functions that a layer offers to the layer above it. For example, the TCP layer provides a reliable byte-stream service to the application layer above it.

Each layer of the protocol stack adds a header containing layer-specific information to the data packet. A header for the network layer might include information such as source and destination addresses. The process of prepending data with headers is called **encapsulation**. Figure 1 shows how data is encapsulated by various headers. During **decapsulation** the inverse occurs: the layers of the receiving stack extract layer-specific information and process the encapsulated data accordingly. It is interesting to note that the process of encapsulation increases the overhead involved in transmitting data. It is necessary to achieve a balance between information and overhead when designing a protocol. Too few layers increase implementation complexity; too many layers can result in high overhead.

The TCP/IP Stack

Figure 1(a) depicts part of the TCP/IP stack used by all systems connected to the Internet. At the bottom are the data link and physical layers which consist of a network interface card and a device driver. The physical layer deals with voltages and the data link layer provides useful services like framing, error detection, error correction, and flow control. Together they are responsible for getting raw bits across a physical link. One important aspect of the Internet stack is that it has no restrictions about the physical medium over which it runs. This characteristic provides the TCP/IP protocol its adaptability and flexibility.

The network layer protocol known as the **Internet Protocol** (IP) can be described as the common thread that holds the entire Internet together. It is responsible for moving data from one host to another, using various cost-based techniques (or 'routing' algorithms). Layers above the network layer take a datastream and break it into chunks of a predetermined size known as **packets** or **datagrams**. These datagrams are then sequentially passed to the network layer, whose job is to route these chunks to the desired destination. Prior to transmitting data, the network layer might subdivide or fragment it into smaller packets for ease of transmission. When all the pieces finally reach the destination, they are reassembled by the network layer into the original datagram. Detailed discussions of the Internet Protocol can be found in RFCs 1958, 1122, and 2600.

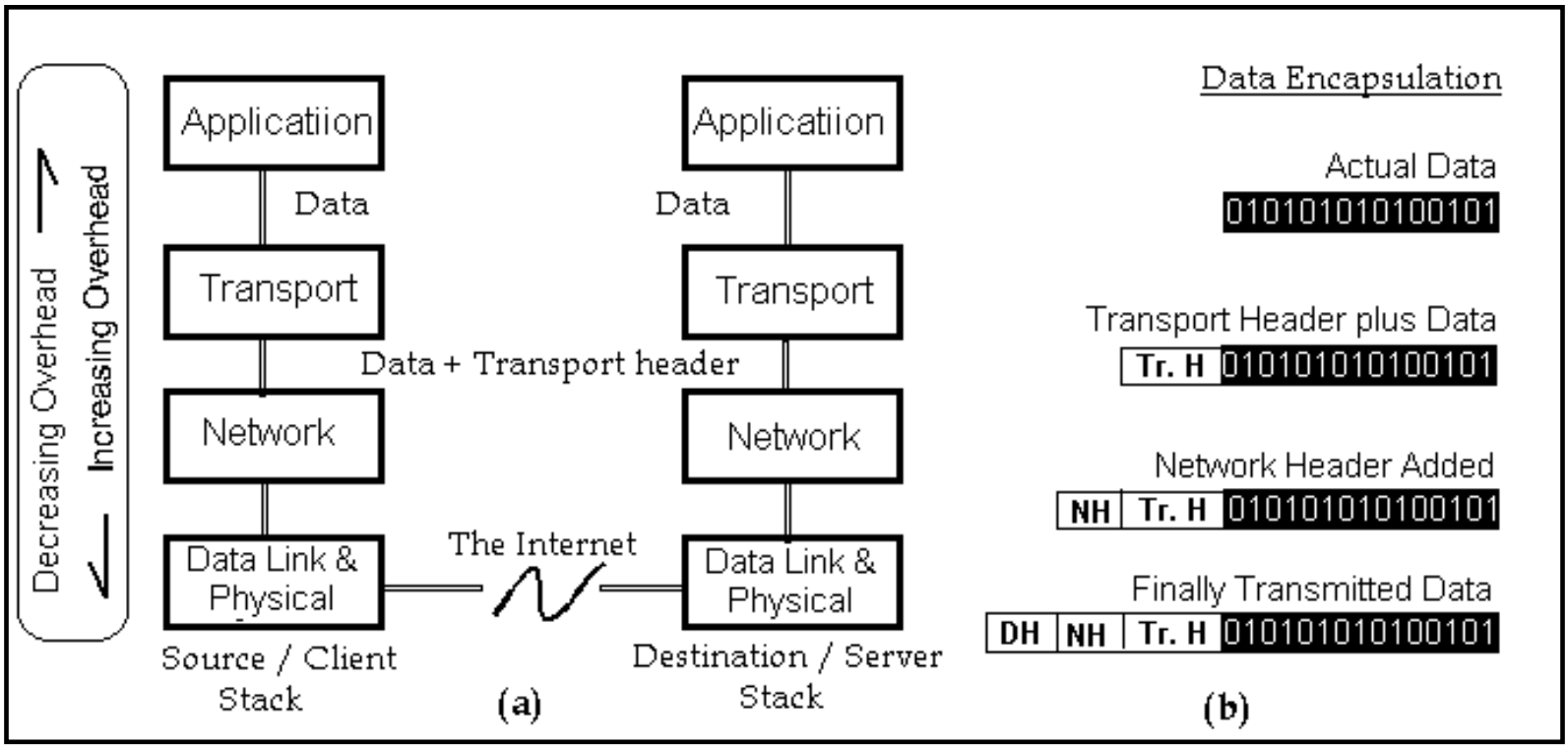

Figure 1: How data travels through the TCP/IP stack

On the source node, the transport layer receives data from the application layer and splits it into chunks that are then passed to the network layer. At the destination node, the transport layer receives these segments of data and reassembles them before passing them to the appropriate process or application. Further details about how data travels through the protocol stack are discussed below.

Note that the transport layer is the first end-to-end layer of the TCP/IP stack. This characteristic means that the transport layer of the source host can communicate directly with its peer on the destination host, without worrying about 'how' data is moved between them. These logistics are already handled by the network layer. The layers below the transport layer understand and carry information required for moving data across links and subnetworks. In contrast, at the transport layer or above, one node can specify details that are only relevant to its peer layer on another node. For example, it is the job of the transport layer to identify the exact process to which data is to be handed over at the remote end. Clearly, this detail is irrelevant for any intermediate router. But it is essential information for the transport layers at both the ends.

The application layer is the layer with which end users normally interact. Whereas the lower three layers are usually implemented as a part of the kernel, the application layer is a user process. Some application-level protocols that are included in most TCP/IP implementations, include:

- **Telnet** for remote login;
- **FTP** for file transfer;
- **SMTP** for mail transfer.

What is the Internet Protocol?

As discussed above, IP is the standard that defines the manner in which the network layers of two hosts interact. These hosts may be on the same network or reside on physically distinct heterogeneous networks. In fact, IP was designed from the very beginning with inter-networking in mind.

IP provides a *connectionless*, *unreliable*, *best-effort* packet delivery service. Its service is called connectionless because it resembles the Postal Service or Western Union more than it does the telephone system [10, 11]. IP packets, like telegrams or mail messages, are treated independently. Each packet is stamped with the addresses of the receiver and the sender. Routing decisions are made on a packet-by-packet basis. IP is quite different from connection-oriented and circuit switched phone systems that explicitly establish a connection between two users before any conversation (data exchange) takes place and maintain a connection for the entire length of exchange.

A best-effort delivery service means that packets might be discarded during transmission, but not without a good reason. Erratic packet delivery is normally caused by the exhaustion of resources or a failure at the data link or physical layer. In a highly reliable physical system such as an Ethernet LAN, the best-effort approach of IP is sufficient for transmission of large volumes of information. In the geographically distributed and highly diverse Internet, which is subject to many vagaries of operation, IP delivery is insufficient and needs to be augmented by a higher-level protocol to provide satisfactory performance.

Finally, do not let the 'unreliable' description alarm you: ensuring reliability is the responsibility of the higher-layer protocols, such as TCP (Transmission Control Protocol). For more details, see RFCs 793, 1122, 1323. If less reliability is acceptable and a higher speed (effective data rate) is required, then the User Datagram Protocol (UDP) is used instead at the transport layer (see RFC 768).

The IP Packet

When learning how a protocol works, it is helpful to study its packet structure and learn how different fields effect the processing of datagrams. All IP packets or datagrams consist of a header part and a text part (payload). Returning to the postal service analogy, the 'header' of the IP packet can be compared with the envelope and the 'payload' with the letter inside it. Just as the envelope holds the address and information necessary to direct the letter to the desired destination, the header aids in the routing of IP packets.

The payload has a maximum size limit of 65536 bytes per packet. It might consist of error and/or control protocols like the Internet Control Message Protocol (ICMP) (see RFC 792). To better understand control protocols, suppose that the postal system fails to find the destination on your letter. They would be obliged to send you a message indicating that the recipient's address was incorrect. Note that this message would reach you through the same postal system that tried to deliver your letter. ICMP works the same way: it packs control and error messages inside IP packets.

See Figure 5-45, page 413 of [10] or page 46, Figure 5-45
Figure 2: The IPv4 packet header [10]

The IP Header consists of a 20-byte fixed part plus a variable part. Its size is optimized to maximize the packet processing rate without utilizing excessive resources. The header begins with a 4-bit version field that keeps track of the version of the IP protocol to which the datagram belongs. This field helps smooth the transition from one version of IP to another, which can take months or even years.

The header also includes the header length field (IHL, 4-bits), which describes how long the header is in 32-bit words. The maximum possible value of this field is 15, which limits the header to 60 bytes (or 15 * 32bit/8bits per byte). This limit would restrict the options field (i.e., the variable part of the header) to 40 bytes.

An IP packet contains a source and a destination address. The source address designates the originating node's interface to the network, and the destination address specifies the interface for an intended recipient or multiple recipients (for multicasting). These addresses are in the form of 32-bit binary strings. (See *IP Addresses and Addressing Scheme* below).

The header also consists of a **Time to Live (TTL)** that is used to limit the life of the packet on the network. Imagine a situation in which an IP packet gets caught in the system and becomes undeliverable. It would then consume the resources indefinitely. The entire Internet could be brought to a halt by a blizzard of such reproducing but undeliverable packets. The TTL field maintains a counter that is normally initialized to thirty count and is decremented each time the packet arrives at a routing step. If the counter reaches zero, the packet discarded.

Three fields in the IPv4 header are devoted to fragmentation. If the network layer receives a transport datagram that too large to transport, it subdivides the data into smaller sized chunks. These fragments are controlled by the DF, MF and Fragment Offset fields in the header. The *DF* stands for *Don't Fragment*, which an order to routers not to fragment the datagrams. This bit set when the destination node incapable of reassembling the fragments. The *MF* field stands for More Fragments, which indicates that the current packet does not contain the final fragment in the datagram. Only the final fragment in a series has this bit turned off. The Fragment Offset field specifies order in which a particular fragment belongs in the current datagram.

All fragments except the last one in a datagram must be a multiple of 8 bytes, which the elementary fragment unit. Since 13 bits are provided for this field there a maximum of 8192 possible fragments per datagram. Fragmentation and reassembly can take a considerable amount of time at the intermediate routers. I will address this issue in my next

column when I discuss IP version 6.

The Type of Service field is not commonly used. It allows the host to specify a tradeoff between fast service and reliable service. Further details on other fields in the IP header can be found in RFC 791.

IP Addresses and Addressing Scheme

As discussed above, every host and router on the Internet has an address that uniquely identifies it and also denotes the network on which it resides. No two machines can have the same IP address. To avoid addressing conflicts, the network numbers have been assigned by the InterNIC (formerly known simply as NIC). Recent changes, though, have been made in this process; for more details see [3].

See Figure 5-47, page 416 of [10] or page 48, Figure 5-47

Figure 3: IPv4 address classes [10]

The various formats for an IP address are shown in Figure 3. Blocks of IP addresses are assigned to individuals or organizations according to one of three categories--Class A, Class B, or Class C. The *network* part of the address common for all machines on a local network. It similar to a postal code that used by a post office to route letters to a general area. The rest of the address on the letter (i.e., the street and house number) are relevant only within that area. It only used by the local post office to deliver the letter to its final destination. The *host* part of the IP address performs this same function. To quickly peruse:

- Class A format: 126 networks with 16 million hosts each
- Class B format: 16,382 networks with up to 64K hosts each
- Class C format: 2 million networks with 254 hosts each
- Class D format: Used for multicasting, in which a datagram directed to a multiple hosts.
- Class E format: Reserved for future use.
- Hosts use the address 0.0.0.0 when they are being booted, this address is not used afterwards. Also the addresses with all zeros or all ones in the host part are reserved for special purposes.

Corporate networks tend to consist of two to three hundred computers per site. A corporation will almost never need sixty-four thousand hosts, for which a class B address designed! This rigid assignment scheme has resulted in large quantities of unused but still unavailable addresses. This waste is a cause of increasing alarm in the Internet Engineering Task Force. I will return to this concern in my next article.

The host part of an IP address can further be split into a sub-network address and a host address. Subnetworks permit organizations to manage groups of addresses more effectively. An excellent tutorial about IP addressing and subnetting can be found at [7].

Future Directions

The next Connector will explore IP routing and limitations of the current implementation. The column will also examine the improvements offered by its successor: IP Version 6.

References

1 Braden R., ed. "Requirements for Internet Hosts -- Communication Layers," RFC 1122. October 1989.

2 Carpenter B., ed. "Architectural Principles of the Internet," RFC 1958. June 1996.

3 Domain Name FAQ http://www.internic.net/faq.html

4 Jacobsen, Braden, Borman, ed. "TCP Extensions for High Performance," RFC 1323. May 1992.

5 Postel, Jon, ed. "Transmission Control Protocol," RFC 793. September 1981.

6 --, ed. "User Datagram Protocol," RFC 768. August 1980.

7 --, ed. "Internet Control Message Protocol," RFC 792. September 1981.

8 --, ed. "Internet Protocol," RFC 791. September 1981.

9 Reynolds, Braden, ed. "Internet Official Protocol Standards," RFC 2600. March 2000.

10 Tanenbaum A.S. *Computer Networks, 3/e.* Prentice Hall. 1996.

11 Understanding IP addressing http://www.3com.com/nsc/501302.html

Definitive Resources in this area include:

- TCP/IP Illustrated, Volume 1 : The Protocols by W. Richard Stevens. 1994.
- TCP/IP Illustrated, Volume 2 : The Implementation by Gary R. Wright and W. Richard Stevens. 1995.
- TCP/IP Network Administration by Craig Hunt, from O'Reilly. Second edition, 1998.
- Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architecture by Douglas Comer. Fourth edition, 2000.

Less Academic Resources include:

- TCP/IP for Dummies by Candace Leiden and Marshall Wilensky. Third edition, 1999.
- Sams Teach Yourself TCP/IP in 24 hours by Joe Casad, Bob Willsey, and Art Cooper. 1998.
