#1

EE 5516:
Introduction to Communication Networks

Dr. Henry Sendaula

Zexi Liu

September 16th, 2008

Electrical and Computer Engineering
Temple University

1. HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC 1945 [6], improved the protocol by allowing messages to be in the format of MIME-like messages, containing metainformation about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, and virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities.

This specification defines the protocol referred to as "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features.

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods that indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI) [3][20], as a location (URL) [4] or name (URN) , for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail as defined by the Multipurpose Internet Mail Extensions (MIME).

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP [16], NNTP [13], FTP [18], Gopher [2], and WAIS [10] protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

Message Headers:

```
message-header = field-name ":" [ field-value ] CRLF

field-name     = token
field-value    = *( field-content | LWS )

field-content  = <the OCTETs making up the field-value
                  and consisting of either *TEXT or combinations
                  of token, tspecials, and quoted-string>
```

2. TCP Protocol

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP is so central that the entire suite is often referred to as "TCP/IP." Whereas IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two *end systems*, for example a Web browser and a Web server. In particular, TCP provides reliable, ordered delivery of a stream of bytes from one program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer. Among its management tasks, TCP controls message size, the rate at which messages are exchanged, and network traffic congestion.

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application programmer desires to send a large chunk of data across the Internet using IP, instead of

breaking the data into IP-sized pieces and issuing a series of IP requests, the programmer can issue a single request to TCP and let TCP handle the IP details.

TCP is used extensively by many of the Internet's most popular application protocols and resulting applications, including the World Wide Web, E-mail, File Transfer Protocol, Secure Shell, and some streaming media applications.

However, because TCP is optimized for accurate delivery rather than timely delivery, TCP sometimes incurs relatively long delays (in the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages, and it is not particularly suitable for real-time applications such as Voice over IP. For such applications, protocols like the Real-time Transport Protocol (RTP) running over the User Datagram Protocol (UDP) are usually recommended instead.

TCP is a reliable stream delivery service that guarantees delivery of a data stream sent from one host to another without duplication or losing data. Since packet transfer is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends, and waits for acknowledgment before sending the next packet. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires. The timer is needed in case a packet becomes lost or corrupt.

TCP (Transmission Control Protocol) consists of a set of rules: for the protocol, that are used with the Internet Protocol, and for the IP, to send data "in a form of message units" between computers over the Internet. At the same time that the IP takes care of

handling the actual delivery of the data, the TCP takes care of keeping track of the individual units of data "packets" (or more accurately, "segments") that a message is divided into for efficient routing through the net. For example, when an HTML file is sent to you from a Web server, the TCP program layer of that server takes the file as a stream of bytes and divides it into segments, numbers the segments, and then forwards them individually to the IP program layer. The IP program layer then turns each TCP segment into an IP packet by adding a header which includes (among other things) the destination IP address. Even though every packet has the same destination IP address, they can get routed differently through the network. When the client program in your computer gets them, the TCP stack (implementation) reassembles the individual segments and ensures they are correctly ordered as it streams them to an application.

A TCP segment consists of two sections:

- header
- data

The TCP header consists of 11 fields, of which only 10 are required. The eleventh field is optional (pink background in table) and aptly named "options".

TCP Header

Bit offset	**Bits 0–3**	**4–7**	**8–15**								**16–31**
0	Source port										Destination port
32	Sequence number										
64	Acknowledgment number										
96	Data offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size
128	Checksum										Urgent pointer
160	Options (optional)										
160/192+	Data										

3. UDP Protocol

User Datagram Protocol (UDP) is one of the core protocols of the Internet Protocol Suite. Using UDP, programs on networked computers can send short messages sometimes known as datagram (using Datagram Sockets) to one another. UDP is sometimes called the Universal Datagram Protocol. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768.

UDP does not guarantee reliability or ordering in the way that TCP does. Datagram may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient, for applications that do not need guaranteed delivery. Time-sensitive applications often use UDP because dropped packets are preferable to delayed packets. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients. Unlike TCP, UDP is compatible with packet broadcast (sending to all on local network) and multicasting (send to all subscribers).

Common network applications that use UDP include: the Domain Name System (DNS), streaming media applications such as IPTV, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online games.

UDP uses ports to allow application-to-application communication. The port field is a 16 bit value, allowing for port numbers to range between 0 and 65,535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

Ports 1 through 1023 (hex 3FF) are named "well-known" ports and on Unix-derived operating systems, binding to one of these ports requires root access.

Ports 1024 through 49,151 (hex BFFF) are registered ports.

Ports 49,152 through 65,535 (hex FFFF) are used as temporary ports primarily by clients when communicating to servers.

Packet Structure:

UDP is a minimal message-oriented Transport Layer protocol that is currently documented in IETF RFC 768. In the Internet Protocol Suite, UDP provides a very simple interface between the Internet Layer below (e.g., IPv4) and the Application Layer above. UDP provides no guarantees to the upper layer protocol for message delivery and a UDP sender retains no state on UDP messages once sent (for this reason UDP is sometimes called the *Unreliable* Datagram Protocol). UDP adds only application multiplexing and check summing of the header and payload. If any kind of reliability for the information transmitted is needed, it must be implemented in upper layers.

+	**Bits 0 - 15**	**16 - 31**
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

The UDP header consists of only 4 fields. The use of two of those is optional (pink background in table).

Source port

This field identifies the sending port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero.

Destination port

This field identifies the destination port and is required.

Length

A 16-bit field that specifies the length in bytes of the entire datagram: header and data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,535 bytes for the data carried by a single UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes.

Checksum

The 16-bit checksum field is used for error-checking of the header *and data*.

With IPv4

When UDP runs over IPv4, the method used to compute the checksum is defined within RFC 768:

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

In other words, all 16-bit words are summed together using one's complement (with the checksum field set to zero). The sum is then one's complemented. This final value is then inserted as the checksum field. Algorithmically speaking, this is the same as for IPv6.

+	**Bits 0 - 7**	**8 - 15**	**16 - 23**	**24 - 31**
0	Source address			
32	Destination address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160	Data			

The difference is in the data used to make the checksum. Included is a pseudo-header that contains information from the IPv4 header:

The source and destination addresses are those in the IPv4 header. The protocol is that for UDP (see *List of IP protocol numbers*): 17. The UDP length field is the length of the UDP header and data.

If the checksum is calculated to be zero (all 0s) it should be sent as negative zero (all 1's). If a checksum is not used it should be sent as zero (all 0s) as zero indicates an unused checksum.

4. DNS Protocol

The Domain Name System (DNS) is a hierarchical naming system for computers, services, or any resource participating in the Internet. It associates various information with domain names assigned to such participants. Most importantly, it translates humanly meaningful domain names to the numerical (binary) identifiers associated with networking equipment for the purpose of locating and addressing these devices world-wide.

An often used analogy to explain the Domain Name System is that it serves as the "phone book" for the Internet by translating human-friendly computer hostnames into IP addresses. For example, www.example.com translates to 208.77.188.166.

The Domain Name System makes it possible to assign domain names to groups of Internet users in a meaningful way independent of each user's physical location. Because of this, World-Wide Web (WWW) hyperlinks and Internet contact information can remain consistent and constant even if the current Internet routing arrangements change or the participant uses a mobile device. Internet domain names are easier to remember than IP addresses such as 208.77.188.166 or 2001:db8:1f70::999:de8:7648:6e8. People

take advantage of this when they recite meaningful URLs and e-mail addresses without caring how the machine will actually locate them.

The Domain Name System distributes the responsibility for assigning domain names and mapping them to Internet Protocol (IP) networks by designating authoritative name servers for each domain to keep track of their own changes, avoiding the need for a central register to be continually consulted and updated.

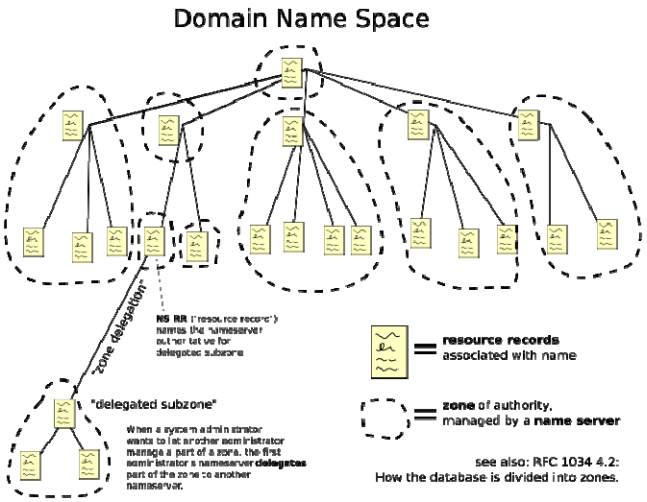
In general, the Domain Name System also stores other types of information, such as the list of mail servers that accept email for a given Internet domain. By providing a world-wide, distributed keyword-based redirection service, the Domain Name System is an essential component of the functionality of the Internet.

Other identifiers such as RFID tags, UPC codes, International characters in email addresses and host names, and a variety of other identifiers could all potentially utilize DNS.

The Domain Name System also defines the technical underpinnings of the functionality of this database service. For this purpose it defines the DNS protocol, a detailed specification of the data structures and communication exchanges used in DNS, as part of the Internet Protocol Suite (TCP/IP). The context of the DNS within the Internet protocols may be seen in the following diagram. The DNS protocol was developed and defined in the early 1980's and published by the Internet Engineering Task Force.

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more *resource records*, which hold information associated with the domain name. The tree sub-divides into *zones* beginning at the root zone. A DNS zone

consists of a collection of connected nodes authoritatively served by an *authoritative DNS name server*. (Note that a single name server can host several zones.)



When a system administrator wants to let another administrator control a part of the domain name space within the first administrator's zone of authority, control can be delegated to the second administrator. This splits off a part of the old zone into a new zone, which comes under the authority of the second administrator's name servers. The old zone ceases to be authoritative for the new zone.

5. SMTP Protocol

Simple Mail Transfer Protocol (SMTP) is a de facto standard for electronic mail (e-mail) transmissions across the Internet. Formally SMTP was first defined in RFC 821 (STD 10) as amended by RFC 1123 (STD 3) chapter 5. The protocol in widespread use today is also known as extended SMTP (ESMTP) and defined in RFC 2821. An update of this standard is currently (2008) pending approval in the IETF.

While electronic mail server software uses SMTP to send and receive mail messages, user-level client mail applications typically only use SMTP for sending messages to a mail server for relaying. For receiving messages, client applications usually

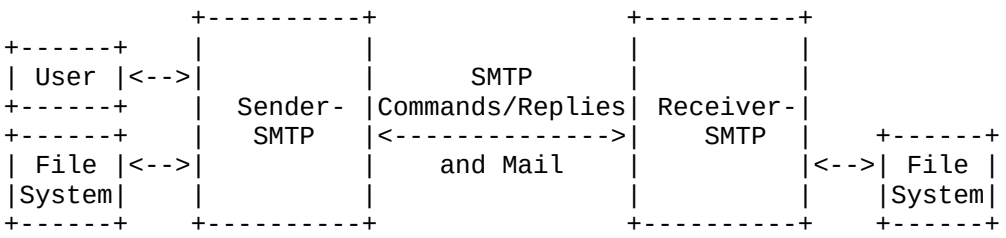
use either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) to access their mail box accounts on a mail server.

SMTP is a relatively simple, text-based protocol, in which one or more recipients of a message are specified (and in most cases verified to exist) along with the message text and possibly other encoded objects. The message is then transferred to a remote server using a procedure of queries and responses between the client and server. Either an end-user's e-mail client, a.k.a. MUA (Mail User Agent), or a relaying server's MTA (Mail Transport Agents) can act as an SMTP client.

An e-mail client knows the outgoing mail SMTP server from its configuration. A relaying server typically determines which SMTP server to connect to by looking up the MX (Mail exchange) DNS record for each recipient's domain name. Conformant MTAs (not all) fall back to a simple A record in the case of no MX. (Relaying servers can also be configured to use a smart host.)

The SMTP client initiates a TCP connection to server's port 25 (unless overridden by configuration). It is quite easy to test an SMTP server using the netcat program (see below).

SMTP is a "push" protocol that cannot "pull" messages from a remote server on demand. To retrieve messages only on demand, which is the most common requirement on a single-user computer, a mail client must use POP3 or IMAP. Another SMTP server can trigger a delivery in SMTP using ETRN. It is possible to receive mail by running an SMTP server. POP3 became popular when single-user computers connected to the Internet only intermittently; SMTP is more suitable for a machine permanently connected to the Internet.



Model for SMTP Use

Figure 1

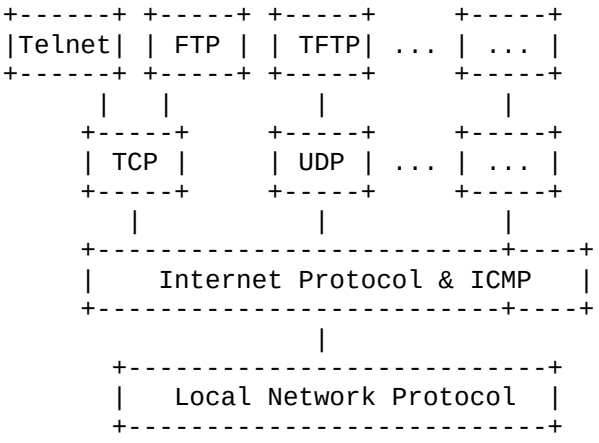
6. IP Protocol

The Internet Protocol (IP) is a protocol used for communicating data across a packet-switched internetwork using the Internet Protocol Suite (TCP/IP).

IP is the primary protocol in the Internet Layer of the Internet Protocol Suite and has the task of delivering datagram's (packets) from the source host to the destination host solely based on its address. For this purpose the Internet Protocol defines addressing methods and structures for datagram encapsulation. The first major version of addressing structure, now referred to as Internet Protocol Version 4 (IPv4) is still the dominant protocol of the Internet, although the successor, Internet Protocol Version 6 (IPv6) is actively deployed world-wide.

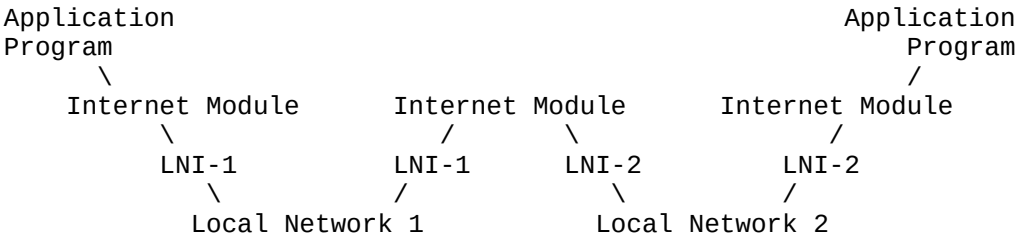
Data from an upper layer protocol is encapsulated inside one or more packets/datagram's (the terms are basically synonymous in IP). No circuit setup is needed before a host tries to send packets to a host it has previously not communicated with (this is the point of a packet-switched network), thus IP is a connectionless protocol. This is

quite unlike Public Switched Telephone Networks that require the setup of a circuit before a phone call may go through (connection-oriented protocol).



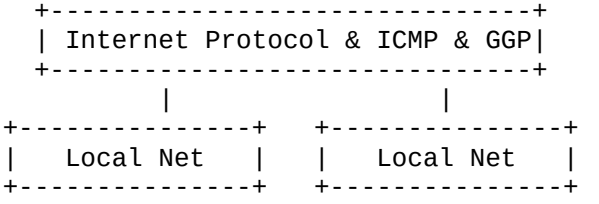
Protocol Relationships

Figure 1.



Transmission Path

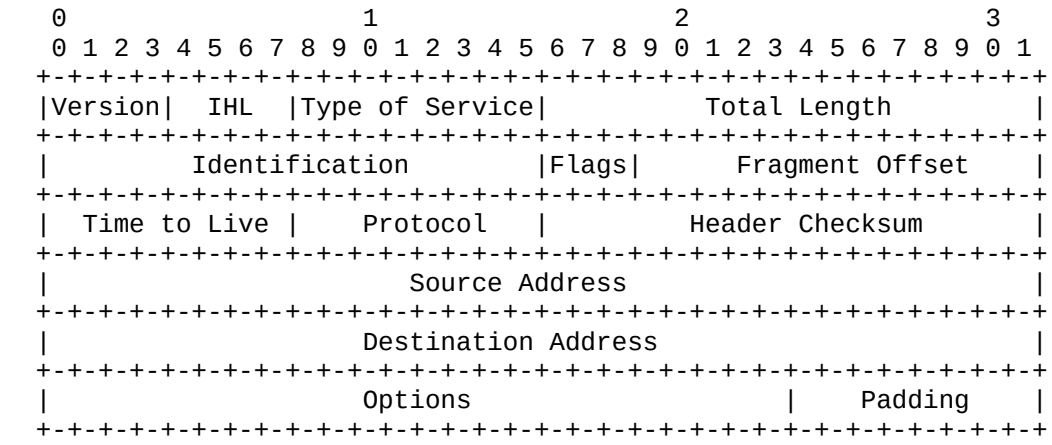
Figure 2



Gateway Protocols

Figure 3.

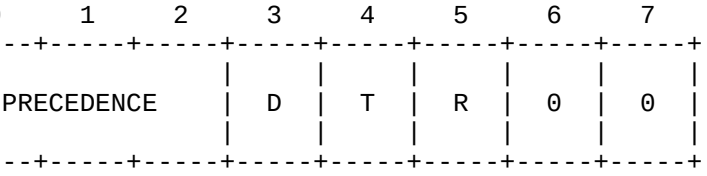
A summary of the contents of the internet header follows:



Example Internet Datagram Header

Figure 4.

```
Bits 0-2:  Precedence.
Bit    3:  0 = Normal Delay,      1 = Low Delay.
Bits   4:  0 = Normal Throughput, 1 = High Throughput.
Bits   5:  0 = Normal Relibility, 1 = High Relibility.
Bit  6-7:  Reserved for Future Use.
```



Precedence

```
111 - Network Control
110 - Internetwork Control
101 - CRITIC/ECP
100 - Flash Override
011 - Flash
010 - Immediate
001 - Priority
000 - Routine
```

7. Security Architecture for IP

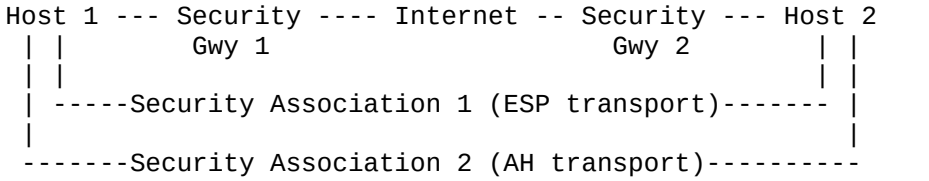
IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. IPsec can be used to protect one or more "paths" between a pair of hosts, between a pair of security gateways, or between a security gateway and a host. (The term "security gateway" is used throughout the IPsec documents to refer to an intermediate system those implements IPsec protocols. For example, a router or a firewall implementing IPsec is a security gateway.)

The set of security services that IPsec can provide includes access control, connectionless integrity, data origin authentication, rejection of replayed packets (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. Because these services are provided at the IP layer, they can be used by any higher layer protocol, e.g., TCP, UDP, ICMP, BGP, etc.

The IPsec DOI also supports negotiation of IP compression [SMPT98], motivated in part by the observation that when encryption is employed within IPsec, it prevents effective compression by lower protocol layers.

Security associations may be combined into bundles in two ways: transport adjacency and iterated tunneling.

```
o Transport adjacency refers to applying more than one
  security protocol to the same IP datagram, without invoking
  tunneling.  This approach to combining AH and ESP allows
  for only one level of combination; further nesting yields
  no added benefit (assuming use of adequately strong
  algorithms in each protocol) since the processing is
  performed at one IPsec instance at the (ultimate)
  destination.
```



o Iterated tunneling refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec

8. Chapter 1, Problem 5

Calculate the total time required transferring a 1,000-KB file in the following cases, assuming an RTT of 100 ms, a packet size of 1-KB data, and an initial 2×RTT of "handshaking" before data is sent.

(a) The bandwidth is 1.5 Mbps, and data packets can be sent continuously.

(b) The bandwidth is 1.5 Mbps, but after we finish sending each data packet we must wait one RTT before sending the next.

(c) The bandwidth is "infinite," meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

(d) The bandwidth is infinite, and during the first RTT we can send one packet (21−1), during the second RTT we can send two packets (22−1), during the third we can send four (23−1), and so on. (A justification for such an exponential increase will be given in Chapter 6.)

Answer:

(a) 1000KB/1.5 Mbps + 2×RTT = 5.53s
(b) 1000KB/1.5 Mbps + (1000KB/1KB - 1) × RTT = 105.2s
(c) 1000KB/(1KB×20)*RTT = 50s
(d) $\sum 2^n = 2^{n+1}-1$,
So, let $\sum 2^n=1000$, we get n = 9.97, n should be given by 10
Thus, Total Time spent = n ×RTT = 1s

9. Chapter 1, Problem 7

Consider a point-to-point link 2 km in length. At what bandwidth would propagation delay (at a speed of 2 m/s) equal transmit delay for 100-byte packets? What about 512-Byte packets?

Answer:

Propagation delay = 2km / (2m/s) = 1000s
Bandwidth = 100-byte / 1000s = 0.8 bps, for 100-Byte packets
Bandwidth = 512-byte / 1000s = 4.096 bps, for 512-Byte packets

10. Chapter 1, Problem 12

What differences in traffic patterns account for the fact that STDM is a cost-effective form of multiplexing for a voice telephone networks and FDM is a cost-effective form of multiplexing for television and radio networks, yet we reject both as not being cost-effective for a general-purpose computer network?

Answer:

STDM/FDM is dividing the bandwidth by time/frequency. So if the information is mainly transmitted by time, this kind of traffic pattern can divide the bandwidth by time and transmit the information. And voice telephone networks provide time-unit services, to which STDM is a cost-effective.
The similar thing happens to FDM, television and radio networks are mostly transmit frequency-related signal, so it is cost-effective for such kind of service.
Both TDM, FDM (frequency division multiplexing have the below characters

(1) applicable only to fixed numbers of flows
(2) requires precise timer (or oscillator and guard bands for FDM)
(3) resources are guaranteed

However, for the Internet or for a general-purpose computer network, different kinds of signals should be transmitted by different type of networks, it is impossible to provide precise timer or oscillator, so both traffic patterns are not cost-effective for this complex, general-purpose computer network.