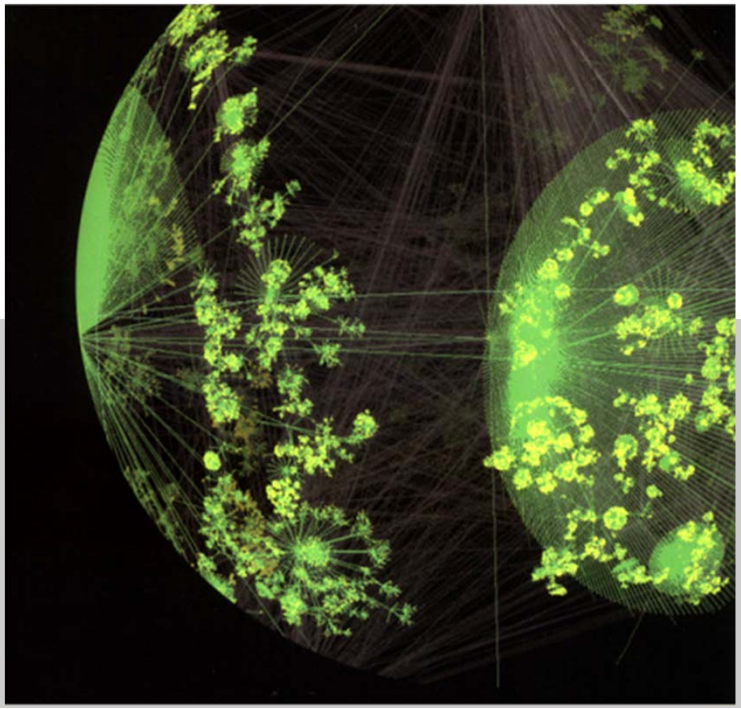


Chapter 5
UDP and Its Applications

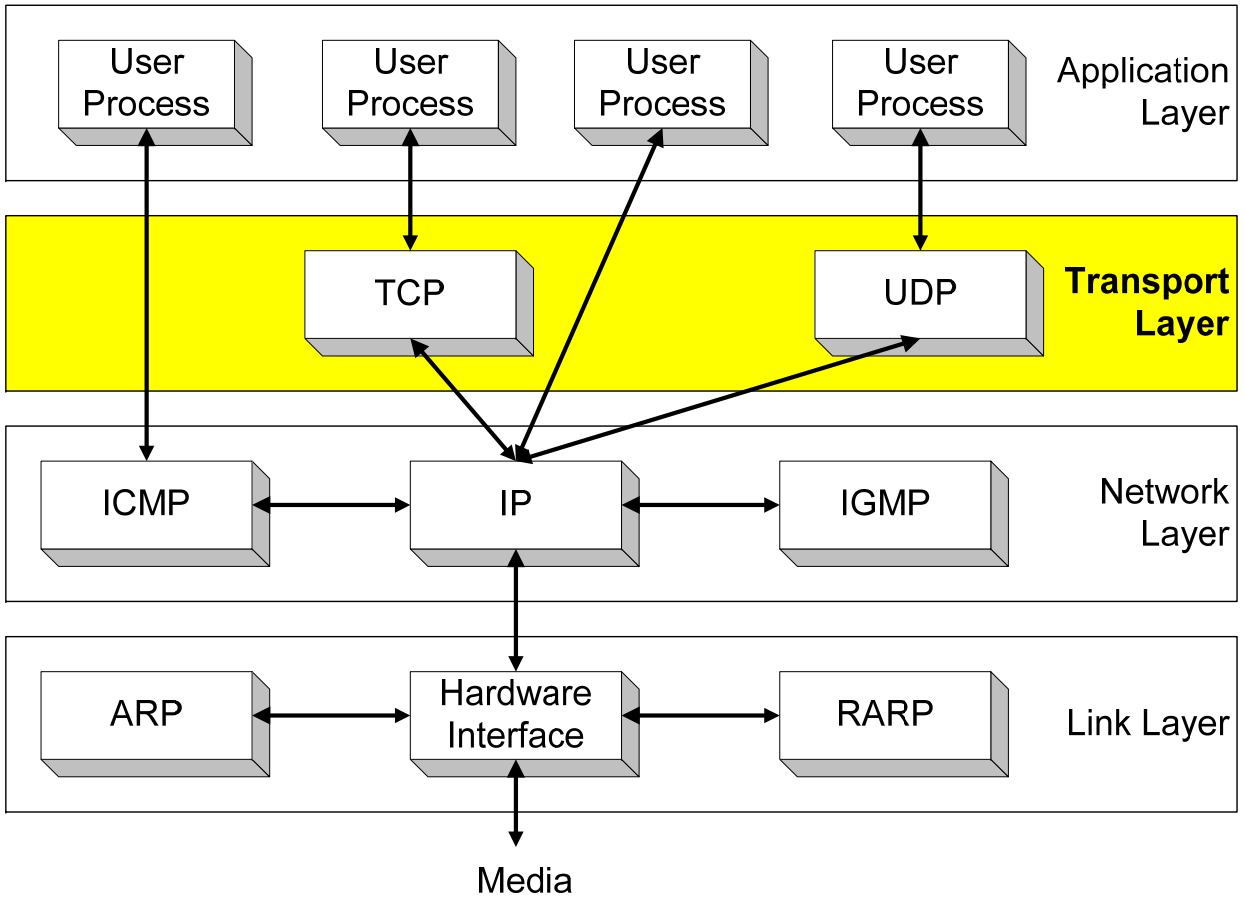
TCP/IP Essentials

A Lab-Based Approach

Spring 2017

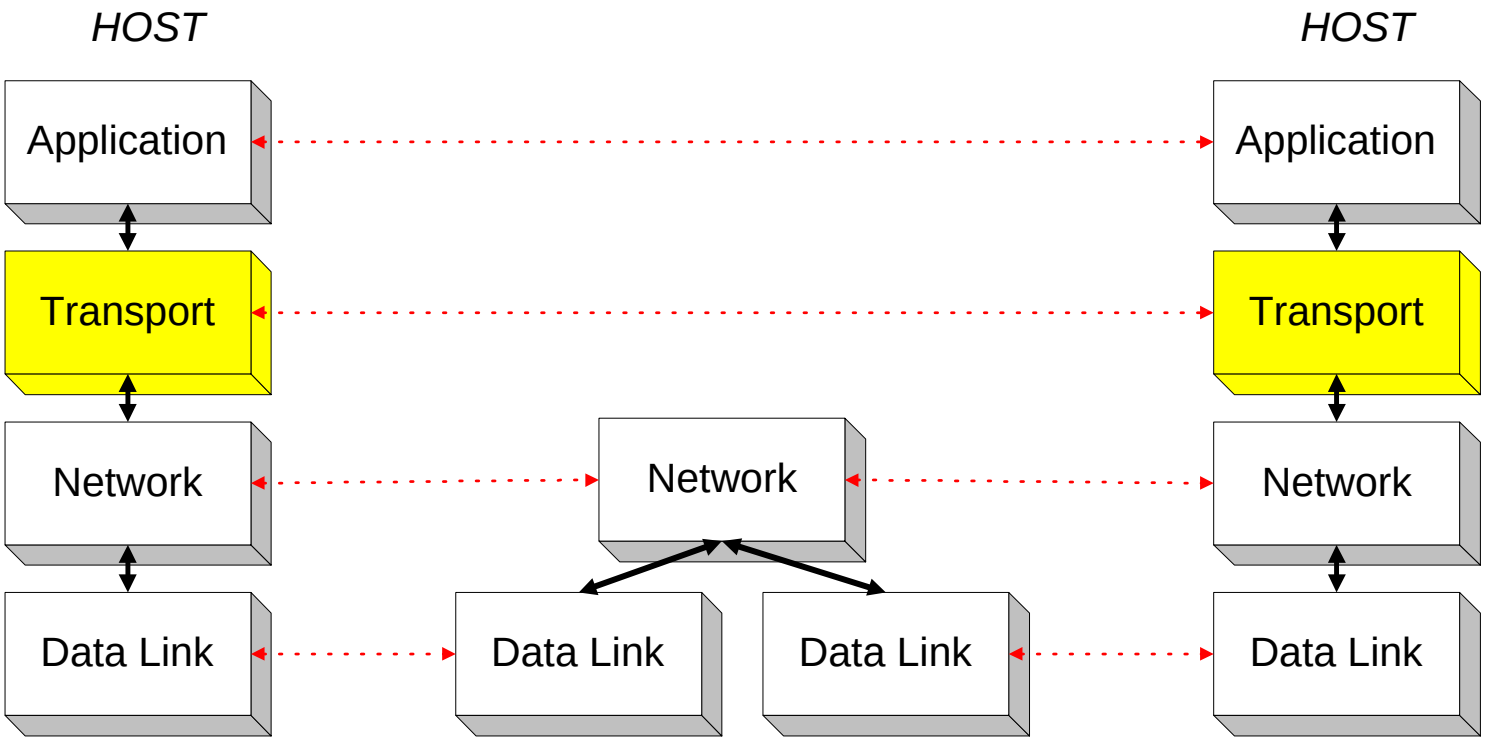
Orientation

The transport layer



Transport Layer Protocols

- Transport layer protocols are end-to-end protocols
- They are only implemented at the hosts



UDP and TCP

The Internet supports two transport protocols:

UDP – User Datagram Protocol

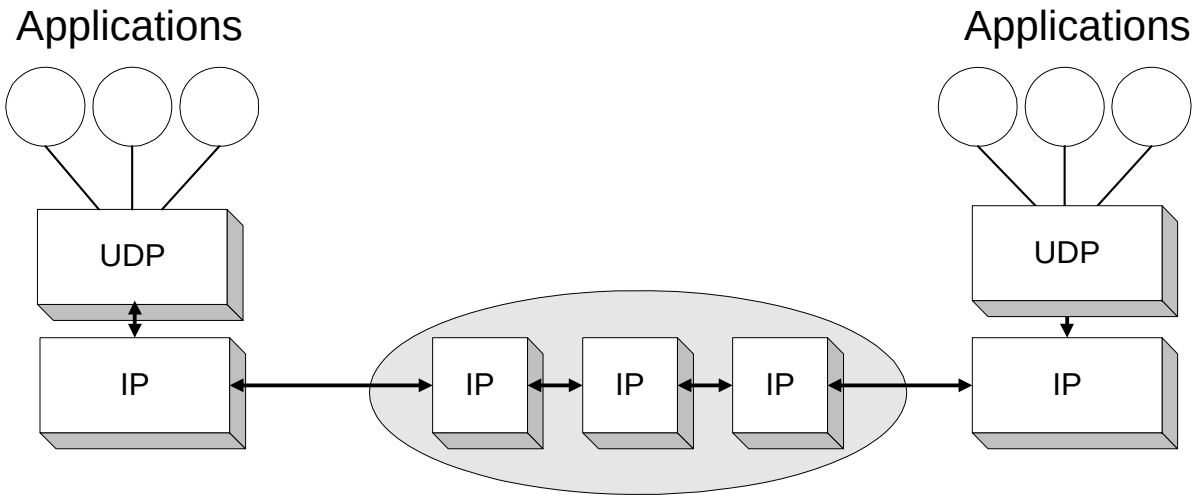
- Datagram oriented
- Unreliable, connectionless
- Simple
- Unicast and multicast
- Commonly used for network control signaling services
  - Network management (SNMP), routing (RIP), naming (DNS), etc.
- Useful for increasing number of applications, e.g., multimedia applications

TCP – Transmission Control Protocol

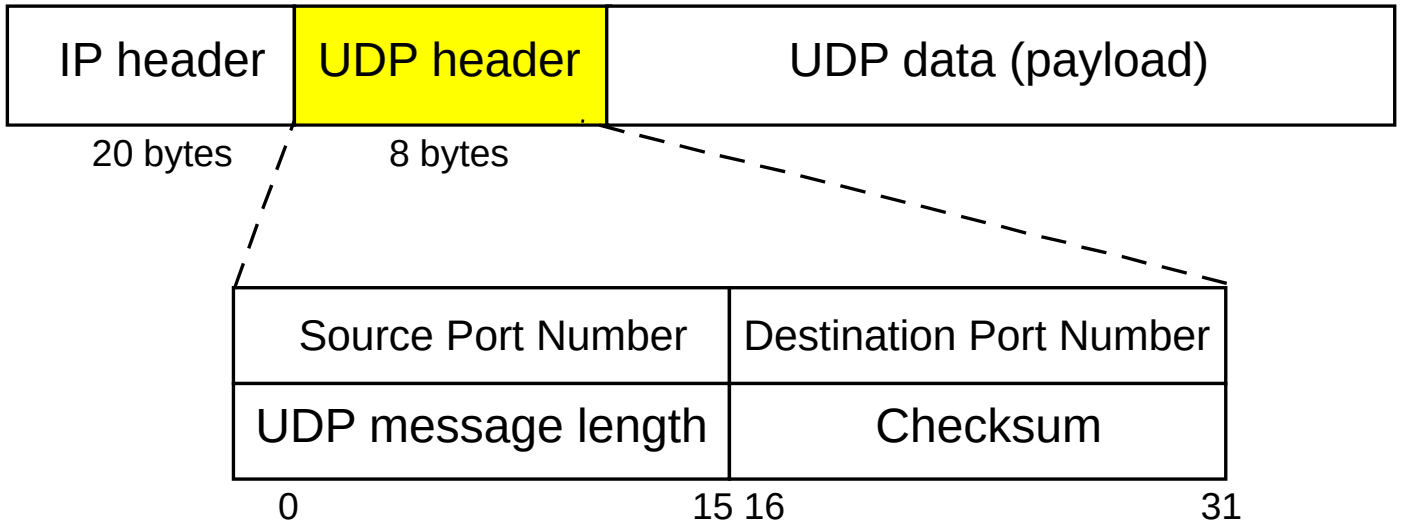
- Stream oriented
- Reliable, connection-oriented
- Complex
- Unicast only
- Currently used by most Internet applications:
  - Web (HTTP), email (SMTP), file transfer (FTP), terminal (telnet), etc.

UDP - User Datagram Protocol

- UDP supports unreliable transmissions of datagrams
- UDP merely extends the host-to-to-host delivery service of IP datagram to an application-to-application service
- The only thing that UDP adds is multiplexing and demultiplexing



UDP Format



- Port Numbers identify sending and receiving applications (processes). The maximum value for a port number is 2^{16}-1= 65,535
- Message Length is between 8 bytes (i.e., data field can be empty) and 65,535 bytes (length of UDP header and data in bytes)
- Checksum is for UDP header and UDP data

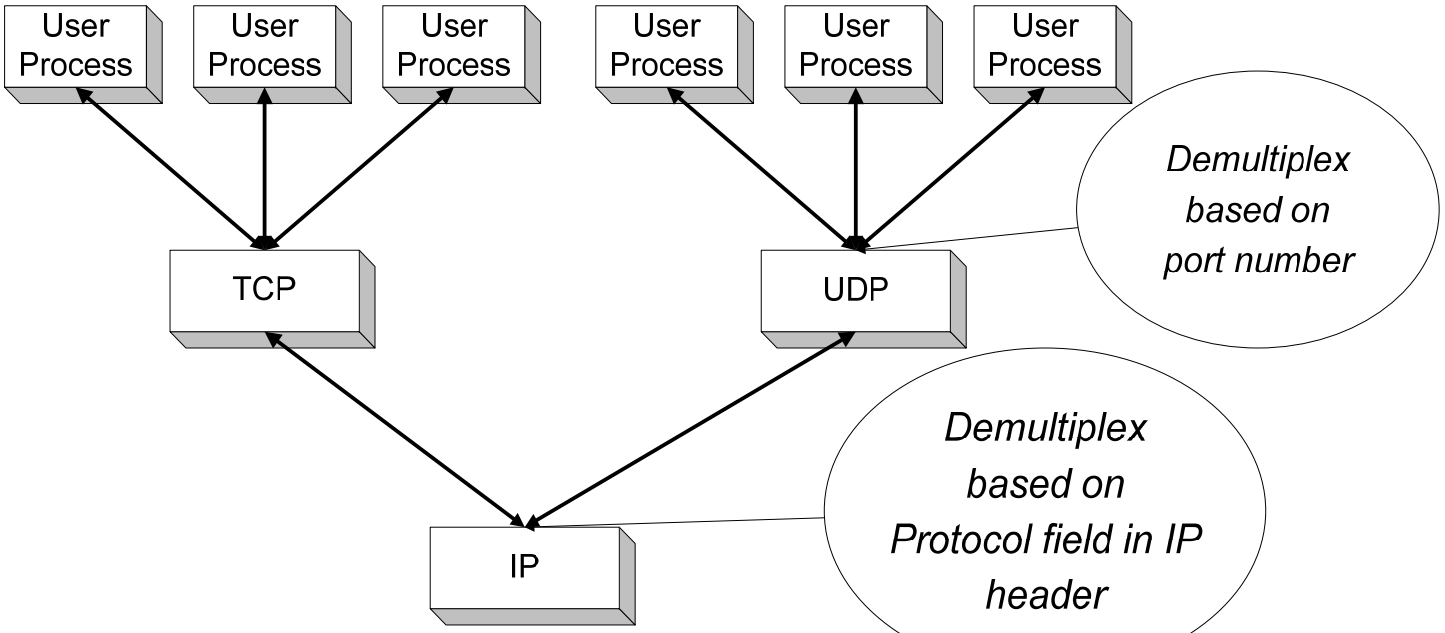
UDP Checksum

- Optional
  - set all 0's if not calculated
  - A calculated checksum can never be all 0's.
- Computed using the UDP header, UDP data and a pseudo-header as below.
- All fields of pseudo-header are available in UDP layer

32-bit Source IP Address		
32-bit Destination IP Address		
0x00	8-bit Protocol (0x17)	16-bit UDP Length (bytes)

Port Numbers

- UDP (and TCP) use port numbers to identify applications
- A globally unique flow of host application can be identified by a 5-tuple **<Src. IP, Dst IP, Src. Port, Dst. Port, Protocol No.>**
- There are 65,535 UDP ports available per host
  - Dynamic/private , used by clients, randomly picked, >49,151 (per IANA)
  - Registered, used by ordinary user processes, 1024 – 49,151
  - Well-known, used by servers, fixed, 1~1023



Ephemeral Port Range

Ephemeral Port: short-lived port used as transport protocol port automatically allocated from a predefined range by the TCP/IP stack software.

- The IANA suggests 49,152 to 65,535 as "dynamic and/or private ports.“
- The BSD uses ports 1,024 through 4,999 as ephemeral ports
- Many Linux kernels use 32,768 to 61,000 specified in */proc/sys/net/ipv4/ip_local_port_range*
- MS Windows OS’ through Server 2003 use the range 1,025 to 5,000 as ephemeral ports; use the IANA range since Windows Vista and Server 2008
- FreeBSD uses the IANA port range since release 4.6.

About Port 0

- Port 0 is a reserved port by IANA
- Many OS' allows a source port of 0 from a high layer application for connecting to a remote host. The OS automatically reassigns an ephemeral port when encapsulating the application data to a transport segment
- No traffic should flow over Internet using port 0 although different OS' have different ways of handling traffic using port 0
- The external behavior of handling port 0 in different OS's can be "fingerprinted" with a set of tests to send TCP or UDP packet from source port 0 to different destination ports
- It is highly recommend that one should block any traffic using this port at your firewall so no program should be listening on port 0 and no program should connect from port 0

Maximum Transmission Unit (MTU)

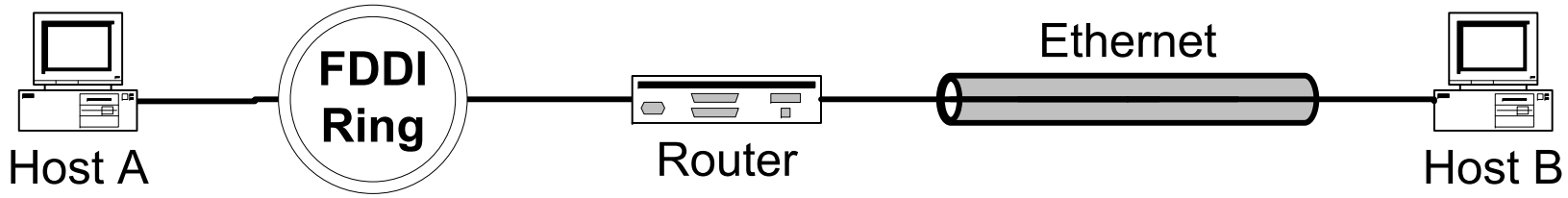
- The frame size limit of the data link protocol specifies a limit on the size of the IP datagram that can be encapsulated by the protocol.
- This limit is called Maximum Transmission Unit (MTU)
- MTUs various for different data link protocols:

Ethernet: 1500	FDDI: 4352
802.2/802.3: 1492	ATM AAL5: 9180
802.5: 4464	PPP: 296 (low delay)

- What if the size of an IP datagram exceeds the MTU?
  - IP datagram is fragmented into smaller units.
- What if the route contains networks with different MTUs?
  - The smallest MTU of any data link is used as the Path MTU.

IP Fragmentation

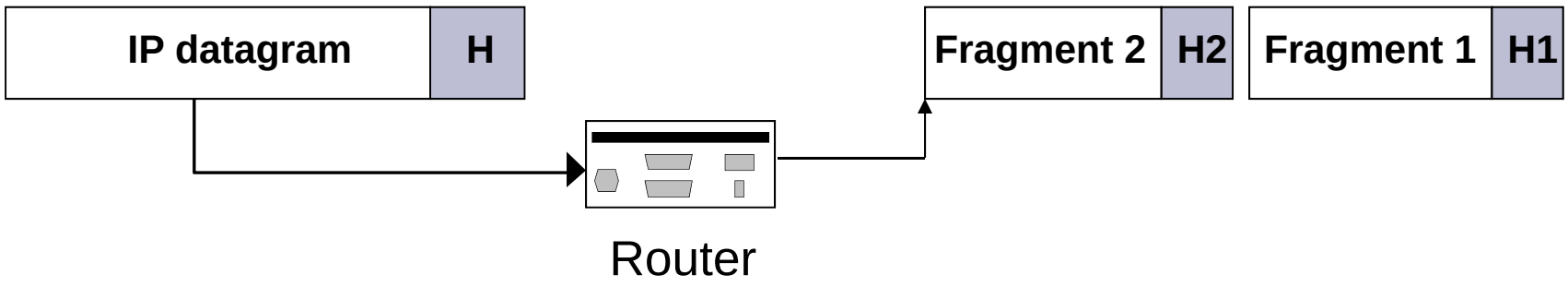
- Host A sends a large IP datagram to Host B.
- How does the intermediate router handle this?
  - IP router splits the datagram into several fragments.
  - Fragmentation requires that the data portion of every fragment except the last be a multiple of 8-bytes.



MTUs: FDDI: 4352 *(4323 bytes for IP data)* Ethernet: 1500 *(1480 bytes for IP data)*

Where is Fragmentation done?

- Fragmentation can be done at the sender and at intermediate routers.
- The same datagram can be fragmented several times.
- Reassembly of original datagram is only done at destination hosts.



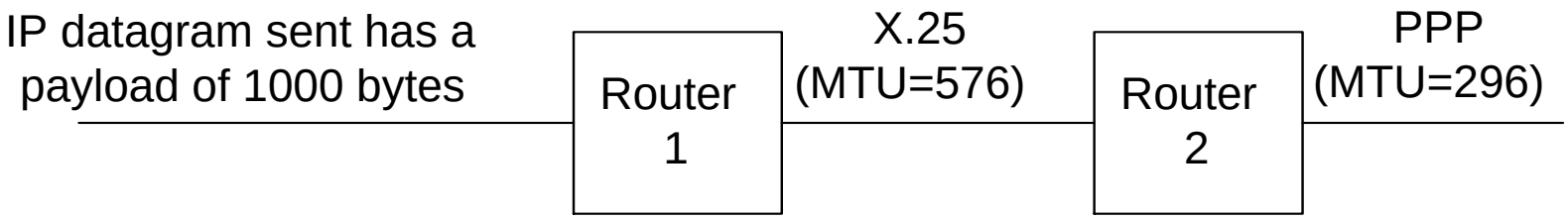
What's involved in Fragmentation?

The following fields in the IP header are involved:

<table>
<tr><td>Version</td><td>Header Length</td><td>Type of Service (TOS)</td><td colspan="4">Total Length (bytes)</td></tr>
<tr><td colspan="3">Identification</td><td>0</td><td>DF</td><td>MF</td><td>Fragment Offset (8-bytes units)</td></tr>
<tr><td colspan="2">Time-To-Live (TTL)</td><td>Protocol Type</td><td colspan="4">Header Checksum (16 bits)</td></tr>
<tr><td colspan="7">… …</td></tr>
</table>

- Identification is the same in all fragments.
- Flags field contains
  - a reserved bit, must be zero,
  - a Don't Fragment (DF) bit that can be set, and
  - a More Fragments (MF) bit.
- Fragment Offset contains the offset (in 8-byte units) of current fragment in the original datagram.
- Total Length is changed to be the size of the fragment.

Fragmentation through Multiple Links

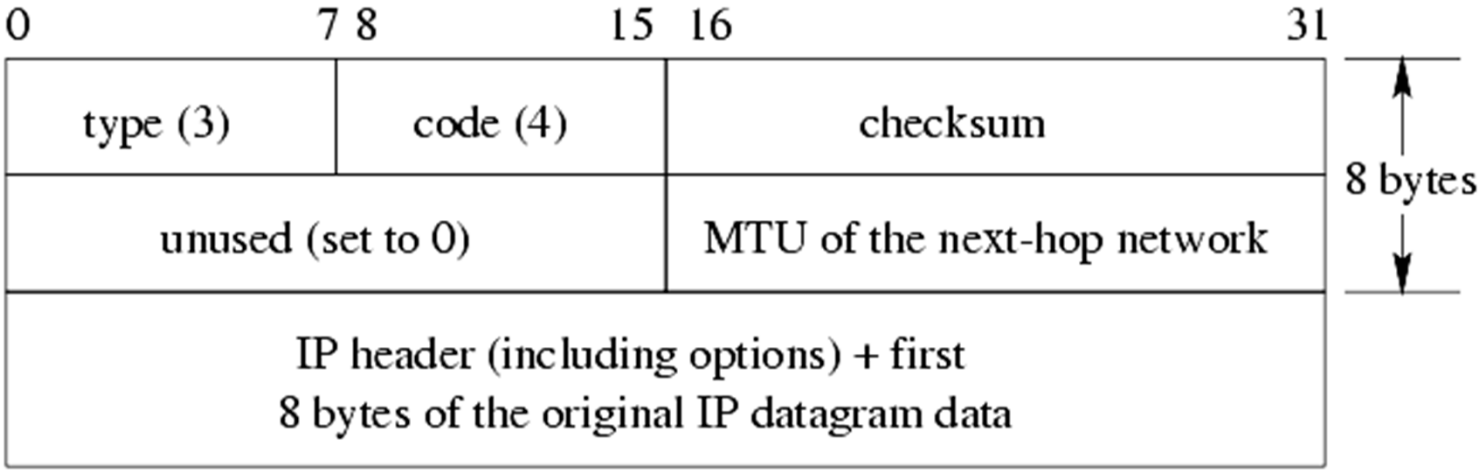


- The ID field stays the same for all fragments of a datagram sent by a sender to allow for reassemble
- The fragment offset is relative to the datagram sent by the sender.
- Two fragments created on X.25 link (offsets 0, 69)
  - 576 – 20 (IP header) = 556; 552 divides by 8 as 69.
  - First fragment: Offset 0, bytes 1~552; second fragment: Offset 69, bytes 553~1000
- Each fragment is fragmented further on the PPP link
  - ID stays the same on all fragments
  - Fragment offset on the second set of fragments is relative to the original (0, 34, 68, 69, 103)
    - > 296-20=276; 272/8 = 34

If the *Don't Fragment* flag is set…

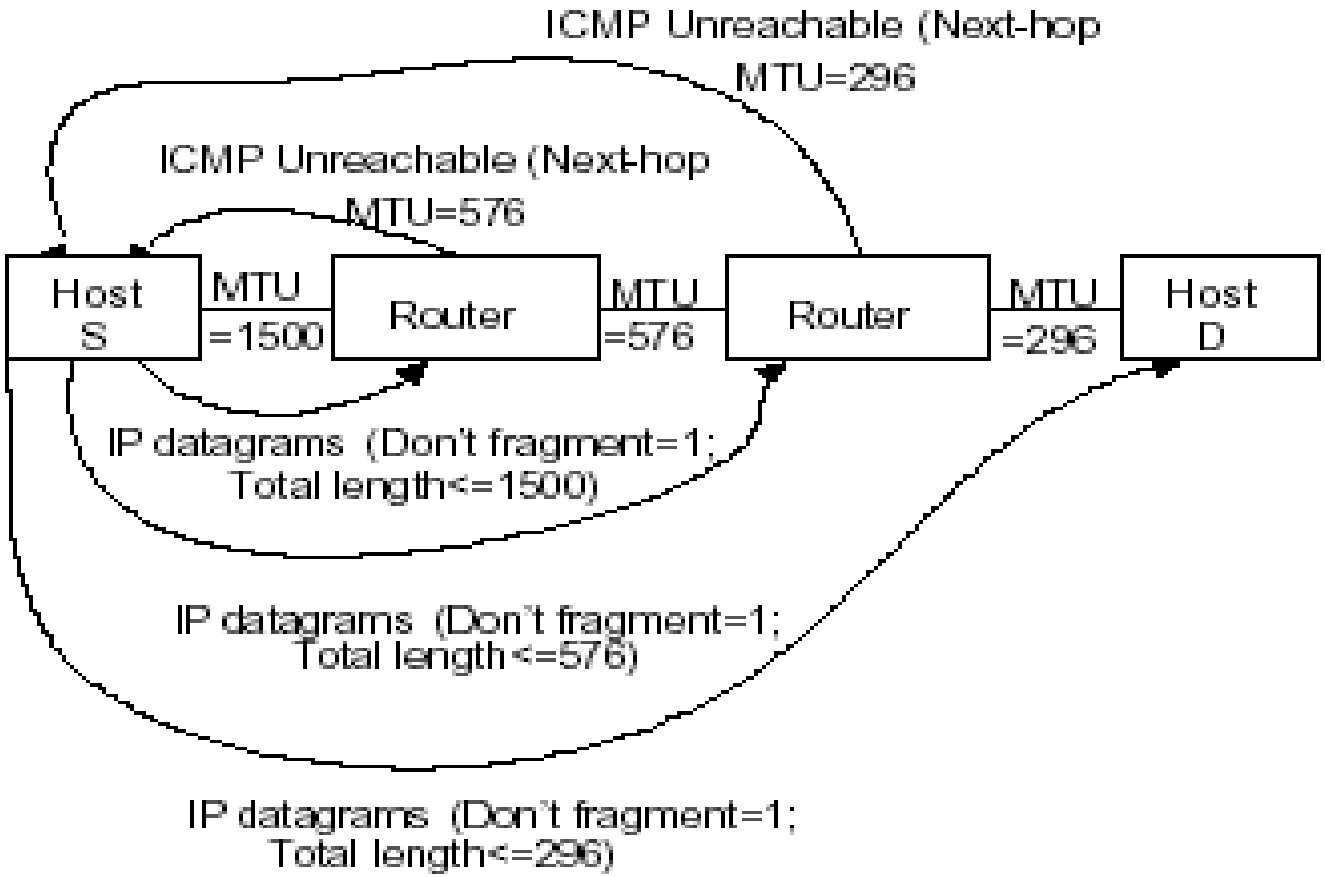
- If fragmentation is needed, and the Don't Fragment flag is set, The router drops the datagram and sends an ICMP unreachable error message to the source.
- This can be used in Path MTU Discovery to find the smallest MTU along a path.

The format of an ICMP unreachable error message:



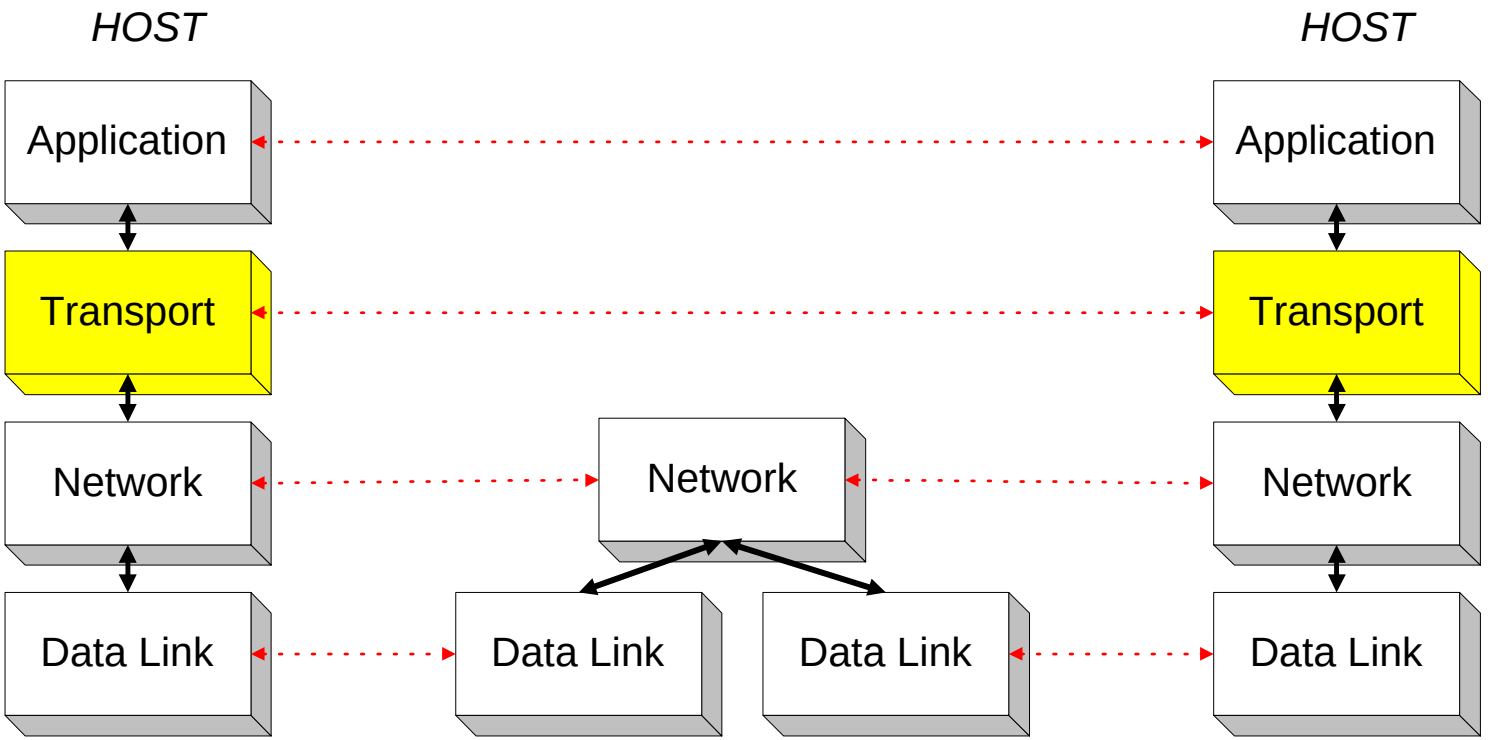
Path MTU Discovery

A host sends a set of IP datagrams with various lengths and the “don’t fragment” bit set



Transport Layer Protocols

- Transport layer protocols are end-to-end protocols
- They are only implemented at the hosts

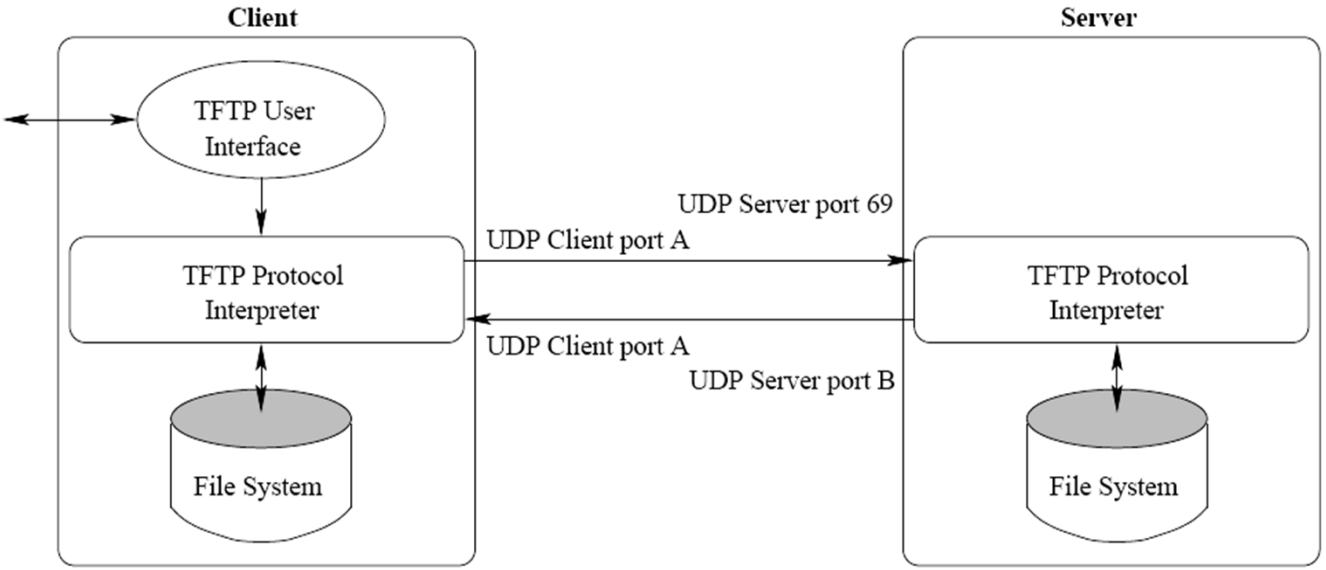


Client-Server Architecture

- Most network application are implemented using a client-server architecture.
- A server provides network service to the clients.
- Servers use well-known port numbers and run these services all the time.
- A client uses a dynamic port number and terminates after the service.
- If a client requests a service on a port number not associated with the server,
  - In UDP, an ICMP port unreachable error is returned to the client;
  - In TCP, the TCP connection is reset.

Trivial File Transfer Protocol (TFTP)

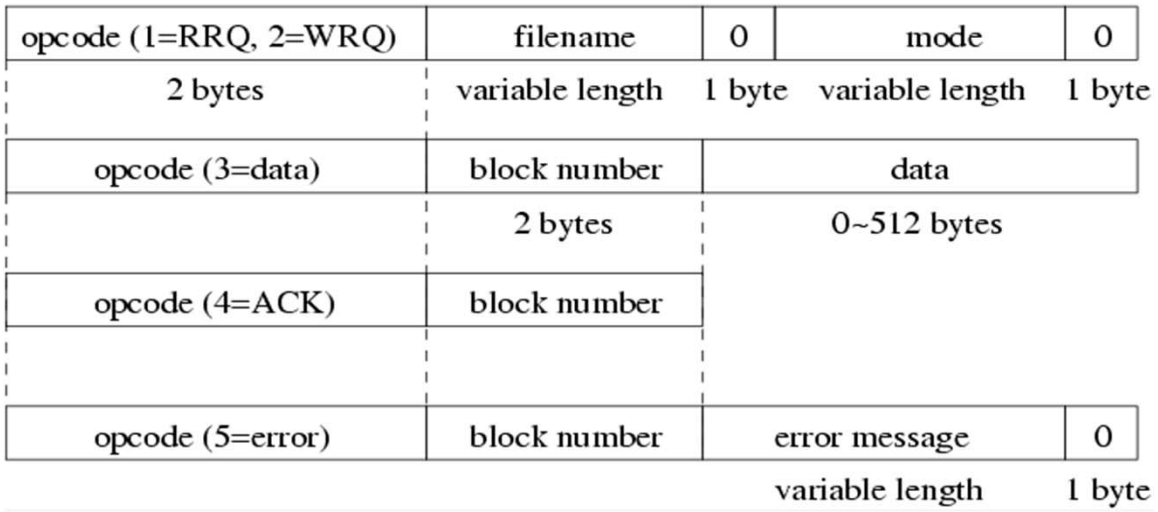
- TFTP uses UDP ~ connectionless and unreliable
  - For small infrequent file transfers
  - Throughput is not a major concern
- TFTP uses a stop-and-wait flow window control algorithm
  - Stop for ACK before sending the next data packet
  - A lost packet causes timeout and retransmission
- Designed for diskless systems to download configuration files during bootstrapping



TFTP Packet Format

A typical TFTP session:

1. A client sends a RRQ with a specific filename to a server on UDP port 69
2. If the requested file exists, the server responds with a data packet of length 512 bytes starting with block number 1
3. The client sends an ACK for block number 1
4. The server sends the next data packet with the block number 2
5. The client sends an ACK for block number 2
6. The above two steps continue until the last data block that is shorter than 512 bytes is sent and ACKed



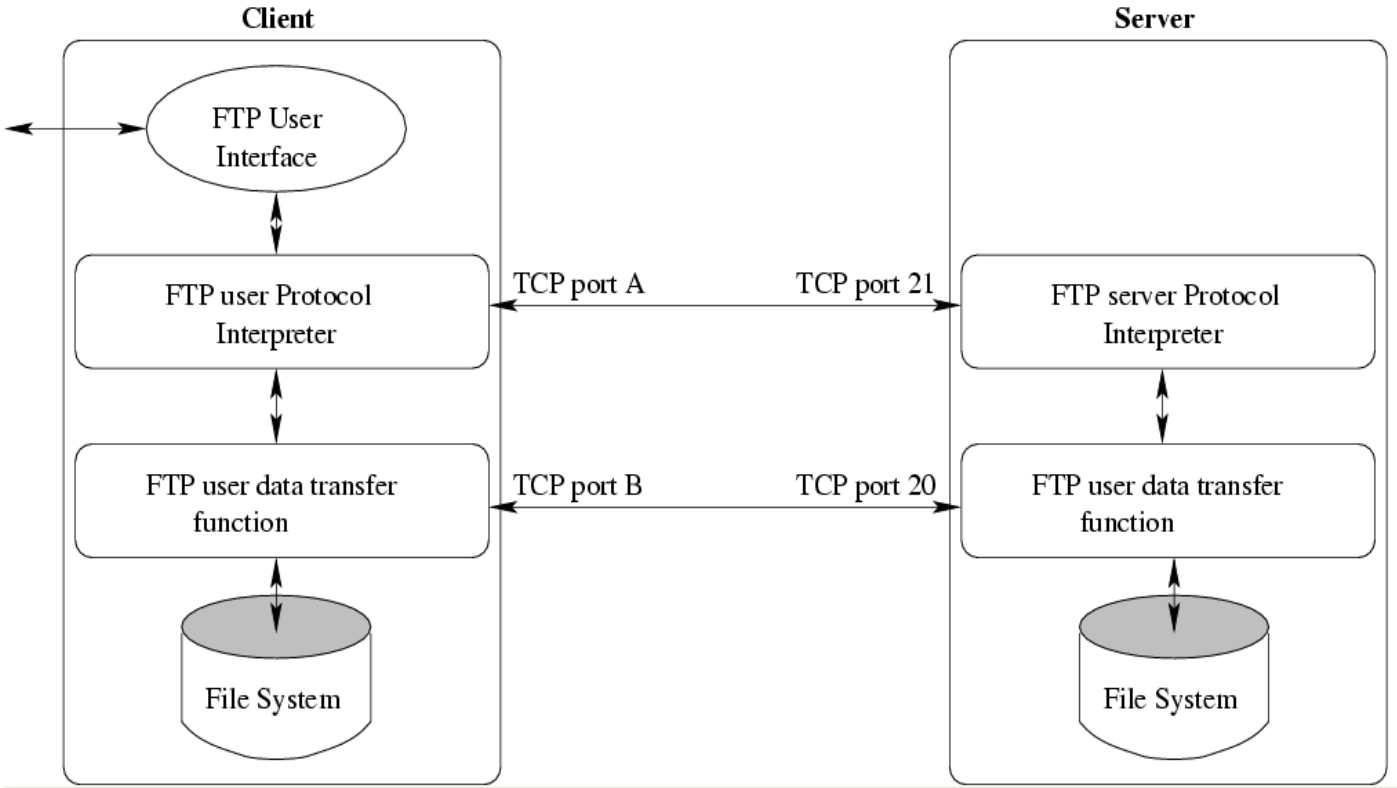
RRQ: Read Request

WRQ: Write Request

File Transfer Protocol (FTP)

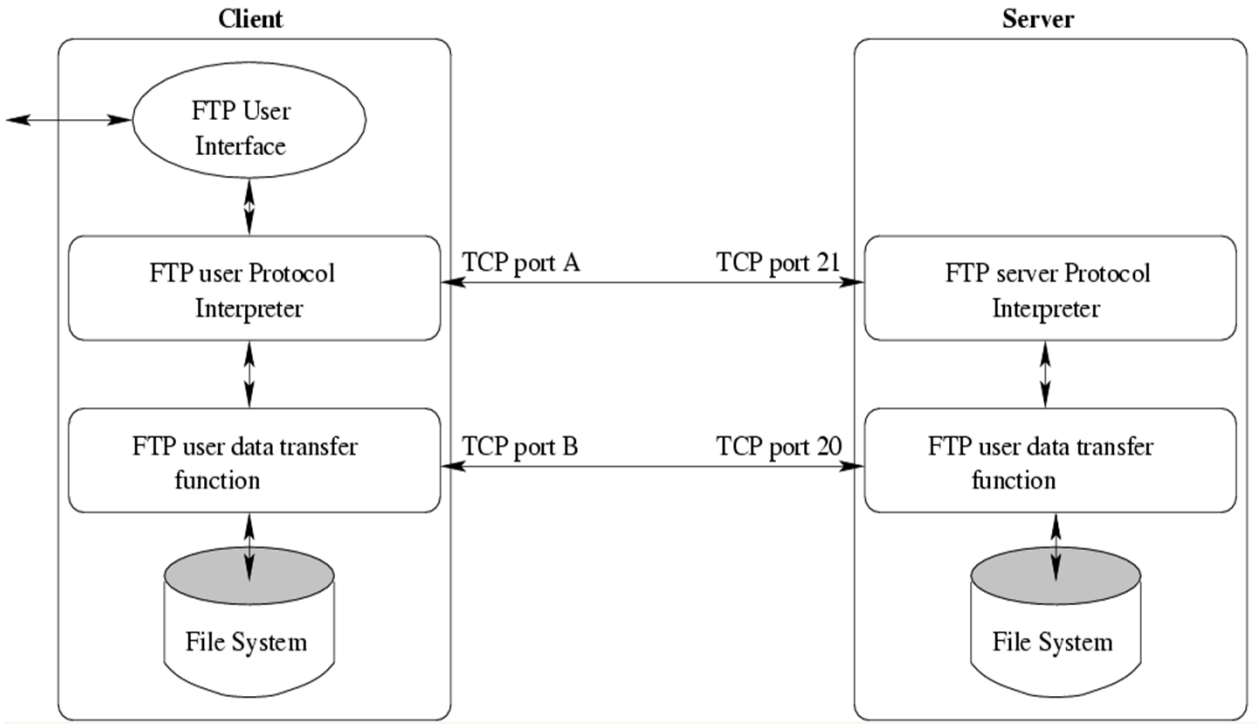
FTP uses two TCP connections

- Control connection: well-known port number at the server = 21
- Data connection: well-known port number at the server = 20



FTP Connection Management

- Control connection, opened by a client, stays up for the duration of the client-server connection.
- Creation of data connection is under the control of a client.
- Client chooses an dynamic port number on the client host for its end of the data connection.
- Client sends the PORT command to the server across the control connection.
- Server receives the port number and issues an active open to that port on the client host. The server uses port 20 at its end for the data connection.
- Multiple FTP sessions from one or more clients to the same FTP server.



FTP Commands

Command	Description
LIST *field*	list files or directories
PASS *password*	password on server
PORT *n1*,*n2*,*n3*,*n4*,*n5*,*n6*	client IP address (*n1*.*n2*.*n3*.*n4*) and port (*n5*x256+*n6*)
QUIT	logoff from server
RETR *filename*	retrieve (get) a file
STOR *filename*	store (put) a file
TYPE *type*	specify file type: A for ASSCII, I for image
USER *username*	username on server

FTP Replies

Typical FTP replies

- 125 Data connection already open; transfer starting
- 200 Command OK
- 331 Username OK, password required
- 425 Can’t open data connection
- 452 Error writing file
- 500 Syntax error (unrecognized command)
- 501 Syntax error (invalid arguments)

File Transfer: FTP vs. TFTP

FTP

- Complex but reliable file transfer use TCP
- Specified in RFC 959, well-known port 21 (control) and 20 (data)
- Data retransmission carried in lower layer by TCP
- Used for general purpose, high throughput applications
- Security feature provided
  - Username and password checking
  - Data transfer may fail when address translation/firewall implemented with random port passing

TFTP

- Simple and quick file transfer over UDP
- Specified in RFC 1350, well-known UDP port 69 (for originating request to server)
- Both ends use a timeout retransmission to resend a block of data
- Often used to
  - Load into a batch file for multiple hosts
  - Bootstrap diskless systems
- No username and password checking; constitutes a security hole.

Backup Slides

TCP Overview

- Transport layer protocol
- Provides connection-oriented, reliable service to applications, such as HTTP, email, FTP, telnet.
- Only support unicast.
- Features:
  - Error control
  - Flow control
  - Congestion control

TCP Connection

- Source and destination port numbers identify the sending and receiving application processes, respectively.
- Socket: the combination of and IP address and a port number.
- A TCP connection is uniquely identified by the two end sockets.

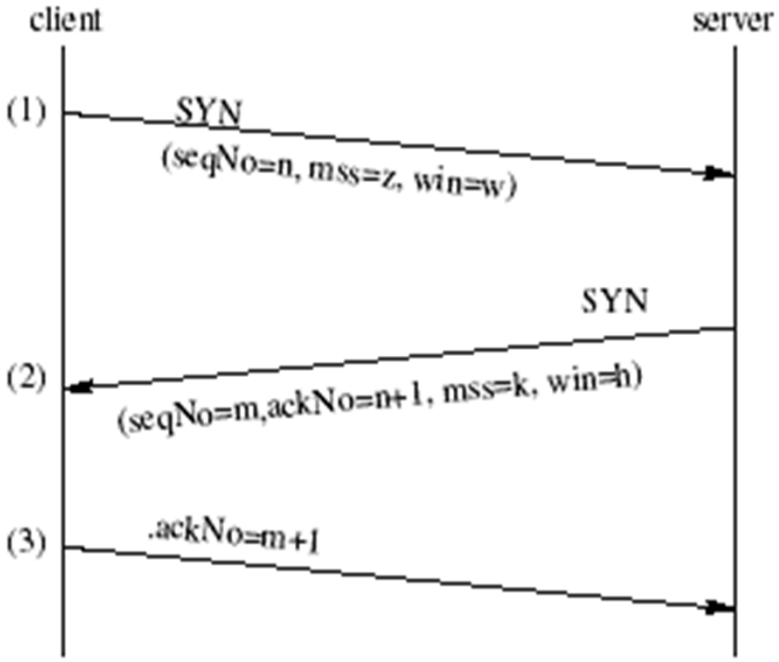
TCP Connection Management

- TCP connection establishment: two end TCP modules
  - allocate required resources for the connection, and
  - Negotiate the value of the parameter uses, such as
    - > Maximum Segment Size (MSS) , given by the receiver side (a.k.a. destination)
    - > Receiving buffer size (i.e. advertised window, WIN), given by the receiver
    - > Initial sequence number (ISN), specified by the sender side (a.k.a. source)
- TCP connection termination

TCP Connection Establishment

Three-way Handshake

- An end host initiates a TCP connection by sending a packet with
  - ISN, say *n*, in the sequence number field,
  - An empty payload field,
  - MSS,
  - TCP receiving window size, and
  - SYN flag bit is set.
- The other end replies a SYN packet with
  - ACK=*n+1*
  - Its own ISN, say *m*
  - Its own MSS, and
  - Its ownTCP receiving window size
- The initiating host sends an acknowledgement: ACK=*m+1*



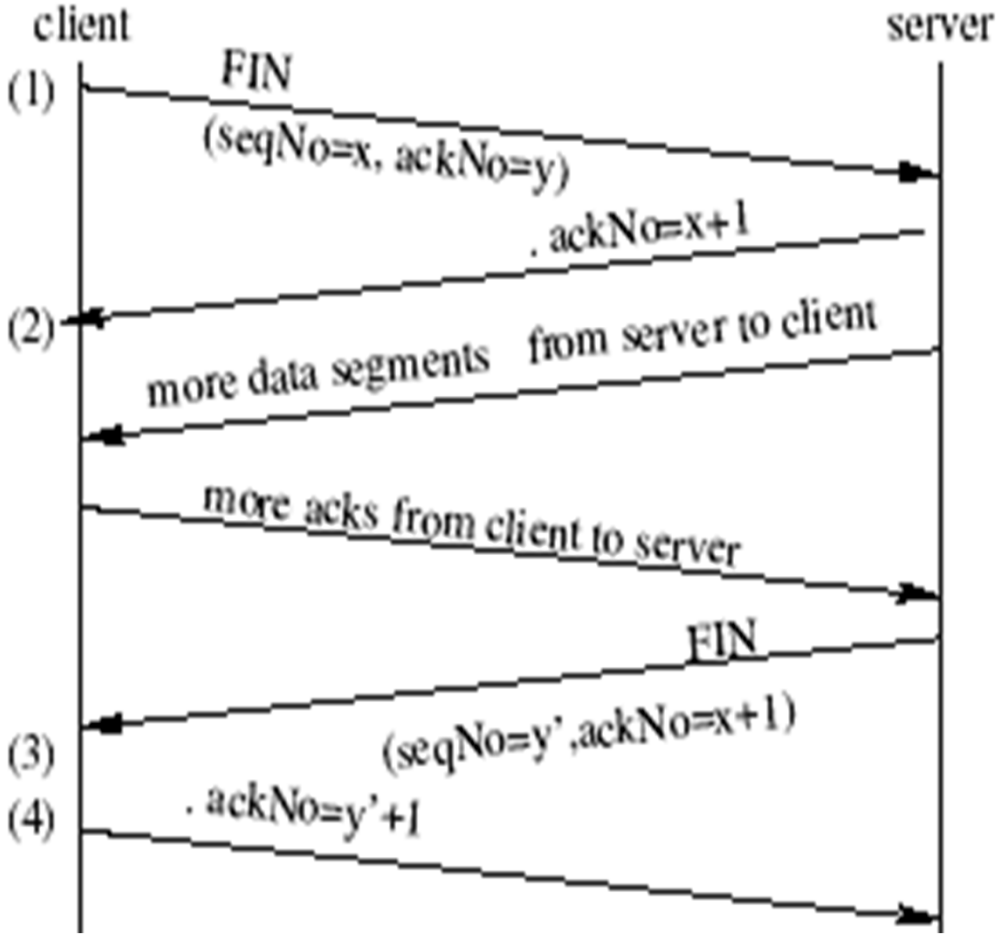
TCP Connection Termination

- A TCP connection is full duplex.
- Each end of the connection has to shut down its one-way data flow towards the other end.
- After termination performed, the connection must stay in the TIME_WAIT state for twice the *Maximum Segment Life (MSL)* to wait for delayed segments.
- If an unrecoverable error is detected, either end can close the TCP connection by sending a RST segment.

TCP Connection Termination (cont'd)

Four-way handshake

- TCP Half-Close
  - One end TCP sends a packet with the FIN flag set.
  - The other end acknowledges the FIN segment.
  - The data flow in the opposite direction still works.
- Do another Half-Close in the opposite direction.



TCP Data Flow

- TCP provides a byte-stream connection to the application layer.
- The sender TCP module
  - Receives a byte stream from the application and puts the bytes in a sending buffer.
  - Extracts the bytes from the sending buffer and sends to the lower network layer in blocks (TCP segments).
- The receiver TCP module
  - Uses a receiving buffer to store and reorder received TCP segments.
  - Restores a byte stream from the receiving buffer and sends to the application process.