CSIS 4222

Ch 22: Datagram forwarding

Ch 25: UDP

Ch 26: TCP

TCP/IP Design Goals

- Interconnect multiple networks in a seamless way
- Be able to survive a partial loss of subnet hardware
- Have flexibility to handle the requirements of diverse applications

TCP/IP Reference Model

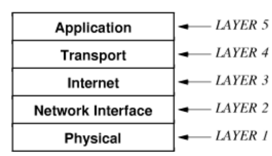
- The protocol used for internetworking
- Based on a view of data communication as

  Processes - Fundamental entities that communicate

  Hosts - Execute processes

  Networks - Connection between hosts

TCP/IP Protocol Stack



TCP/IP Reference Model

Transfer of data to a process requires

- Getting data to the right host where the process resides
- Getting data to the right process on the host

Protocols provide mechanisms to distinguish between

- Multiple computers on a network (Layer 3)
- Multiple applications on a computer (Layer 4)
- Multiple copies of a single application on a computer (Layer 4)

Internet Packets

- Created and understood only by software
- Called *IP datagram*
  - A self contained packet that carries sufficient information for routing from source *host* to destination *host*

The IP Datagram

Datagram size is determined by the application that sends data

- Fixed size header fields
- Payload can be up to 64K octets

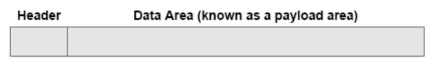


Figure 22.1 The general form of an IP datagram with a header followed by a payload.

IP Semantics

- IP is connectionless
  - Each datagram contains identity of destination
  - Each datagram sent/handled independently
- Routes can change at any time
- Motivation: accommodate all possible networks

Best-Effort Delivery

IP does not guarantee that it will handle all problems

- Datagram loss
- Corruption of data
- Delayed or out-of-order delivery
- Datagram duplication

Encapsulation Across Multiple Hops

- Each router in the path from the source to the destination:
  - *Unencapsulates* incoming datagram from frame
  - Processes datagram (determines next hop)
  - *Encapsulates* datagram in outgoing frame
- Datagram may be encapsulated in a different hardware format at each hop
  - Datagram survives entire trip across Internet
  - Frame only survives one hop

Maximum Transmission Unit (MTU)

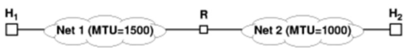
- Every hardware technology specification defines the maximum size of the frame data area
- Any datagram encapsulated in a hardware frame must be smaller than the MTU for that hardware

MTU and Heterogeneous Networks

An internet *may* have networks with different MTUs

Host 1
- Creates datagram for Host *2*
- Uses datagram size of 1500 octets
- Transmits datagram across Net 1

Router R
- Receives datagram in Net 1
- Must send datagram through Net *2*
- Uses *fragmentation* because of smaller MTU

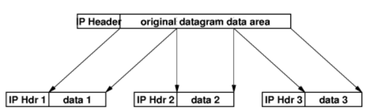


Datagram Fragmentation

- Performed by routers
- Divides datagram into pieces (*fragments)*
- Fragments sent separately
- Destination host *reassembles* fragments

Datagram Fragmentation

Each fragment's IP datagram header
- Identifies original datagram (IDENT)
- Indicate where fragment fits (FRAG OFFSET)



Multiple Fragmentation

- Occurs when fragment is too large for network MTU
- Suppose MTUs along internet path are

  $1500 \rightarrow 1500 \rightarrow 1000 \rightarrow 1500 \rightarrow 575 \rightarrow 1500$
- Fragmentation must occur twice

Fragment Loss

Receiver
- Collects incoming fragments
- Reassembles when all fragments arrive
- Does not know identity of router that did fragmentation
- Cannot request missing pieces

- Consequence: Loss of one fragment means loss of entire datagram

Fragment Loss

How does destination identify lost fragment?
- Sets timer with each fragment
- If timer expires before all fragments arrive, fragment assumed lost
- Datagram dropped

IP vs. Transport

IP provides computer-to-computer communication

- Unreliable datagram service from *machine-to-machine* (no acknowledgement from receiver)

Transport protocols provide application-to-application communication

- Needs extended addressing mechanism to identify applications

Transport Protocol Functionality

- Identify sending and receiving *applications*
- Optionally provides
  - Reliability
  - Flow control
  - Congestion control
- Two main transport protocols
  - *Transmission Control Protocol (TCP)*
  - *User Datagram Protocol (UDP)*

User Datagram Protocol (UDP)

Provides unreliable transfer of independent messages that requires minimal

- Overhead, Computation, Communication

Same best-effort delivery as IP

UDP Details

- Connectionless service paradigm
- Messages encapsulated in IP datagrams
- UDP header identifies
  - Sending application
  - Receiving application
  - Message length
  - Checksum

UDP Encapsulation

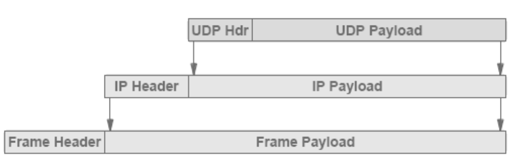


Figure 25.3 The encapsulation of a UDP message in an IP datagram.

© 2009 Pearson Education Inc., Upper Saddle River, NJ. All rights reserved.

UDP is Connectionless

- An application using UDP does not need to pre-establish communication before sending data
  - can generate and send data at any time
- UDP does not maintain state
- UDP does not use control messages
  - communication consists only of the data messages themselves
- UDP has very low overhead

Message-Oriented Interface

- Each time an application requests that UDP send data
  - UDP does not divide a message into multiple packets
  - UDP does not combine messages for delivery
- Consequences
  - Positive: each message will be exactly the same as was transmitted
  - Negative: each UDP message must fit into a single IP datagram

Message-Oriented Interface

- IP datagram size forms an absolute limit on the size of UDP message
- If an application sends extremely small messages
  - datagrams will have a large ratio of header octets to data octets
- If an application sends extremely large messages
  - datagrams may be larger than the network MTU and will be fragmented by IP

Identifying an Application

Multiplexing/demultiplexing
- Multiple streams of data are sent over a single connection
- Must identify sender and receiver unambiguously
- Each application is assigned a unique integer (*protocol port number*)

Protocol Ports

Server
- Always uses same port number

(Generally uses lower port numbers)

Client
- Obtains unused port from protocol software

(Generally uses higher port numbers)

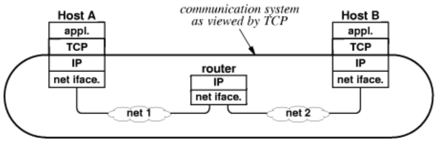
Protocol Port Example

- Domain name server application uses port 53
- Application using DNS obtains port 1045
- UDP datagram sent from application to DNS server has
  - Source port number 1045
  - Destination port number 53
- When DNS server replies, UDP datagram has
  - Source port number 53
  - Destination port number 1045

TCP Service

- Connection Oriented
  - An application must first request a connection to a destination
- Stream Interface
  - Application sends a continuous sequence of octets (data not grouped into messages)
- Complete Reliability
  - TCP guarantees that the data sent across a connection will be delivered completely and in order

TCP on one host uses IP to communicate with TCP on another host



A Contradiction?

- IP offers best-effort (unreliable) delivery
- TCP uses IP
- TCP provides completely reliable transfer
- How is this possible?

Reliable Data Transmission

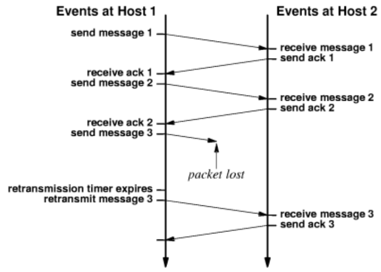
Positive acknowledgment

- Receiver returns short message (ACK) when data arrives

Retransmission

- Sender starts timer whenever data is transmitted
- If timer expires before acknowledgment arrives, sender *retransmits* the same data

Retransmission



TCP Waiting Time

Time for acknowledgment to arrive depends on

- Distance to destination
- Current traffic conditions
- Traffic conditions change rapidly

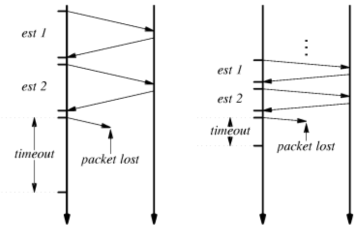
The Key to TCP's Success: Adaptive Retransmission

- Keep estimate of round trip time (RTT) on each connection
- Use current estimate to set retransmission timer

Estimating RTT

- For each segment, its RTT is the amount of time
  - from when the segment is sent (passed to IP)
  - until an ACK is received
- Can vary from segment to segment
- Maintain an estimated (*weighted average*) RTT

Adaptive Retransmission



Adaptive Retransmission

As it sends data packets and receives ACKs

- TCP generates a sequence of round-trip estimates
- It uses a statistical function to produce a weighted average
- TCP keeps an estimate of the variance
- It uses a linear combination of the estimated mean and variance to compute estimated time:

$$\text{Timeout} = \text{EstRTT} + n \times \text{variance}$$

Duplicates and Out-of-Order Delivery

- The sender attaches a sequence number to each packet
- The receiver stores the sequence number of the last packet received in order and a list of additional packets that arrived out of order
- The receiver examines the sequence number to determine how the packet should be handled
- If the packet has already been delivered or the sequence number matches one of the packets waiting on the list the software discards the new copy

Replay

- Very long delays can lead to replay errors
- A packet from an earlier communication might be accepted and the correct packet discarded as a duplicate
- To prevent replays, protocols mark each session with a unique ID and require this ID in each packet
- The protocol discards any arriving packet that contains an incorrect ID

Flow Control

- Necessary to prevent a fast computer from sending so much data that it overruns a slower receiver
- Simplest form of flow control is stop-and-go
  - a sender waits after transmitting each packet
  - when the receiver is ready for another packet, it sends a control message (like ACK)
  - results in very low throughput
- A better flow control technique is sliding window

Flow Control to Data Overrun

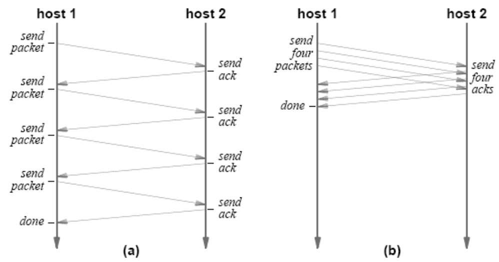


Figure 26.3 Comparison of transmission using (a) stop-and-go, and (b) sliding window.

© 2009 Pearson Education Inc., Upper Saddle River, NJ. All rights reserved.

TCP Flow Control

Receiver

- Advertises available buffer space (window)
  - Each acknowledgment carries new window information
- Interpretation: I have received octets up through *X*, and can take *Y* more octets

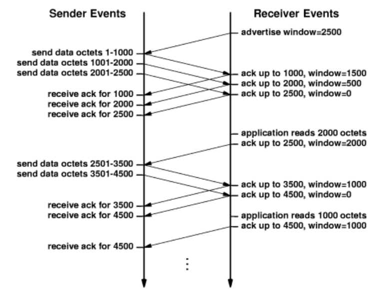
Sender

- Can send up to entire window size before ACK arrives

Buffers, Flow Control, and Windows

- Window is the buffer space available at any time A TCP window is measured in octets
- When a connection is established each end of the connection allocates a buffer to hold incoming data and sends the size of the buffer to the other end
- As data arrives receiving TCP sends ACKs, which specify the remaining buffer size

Window Advertisement



Congestion Control

- Excessive traffic can cause packet loss
  - Transport protocols respond with retransmission
  - Excessive retransmission can cause *congestion collapse*
- TCP interprets packet loss as an indicator of congestion
- Sender uses TCP *congestion control* and slows transmission of packets
  - Sends single packet
  - If acknowledgment returns without loss, sends two packets
  - When TCP sends one-half window size, rate of increase slows
- *Many* variations exist

Techniques to Avoid Congestion

- Using delay and loss to estimate congestion is reasonable in the Internet because:
  - Modern network hardware is very reliable
  - Most delay and loss is from congestion
- The appropriate response to congestion
  - Reducing the rate at which packets are being transmitted
  - Sliding window protocols can achieve the effect of reducing the rate by temporarily reducing the window size

Connection Startup and Shutdown

- Connection startup
  - Must be reliable
- Connection shutdown
  - Must be graceful (guarantee delivery of all data after endpoint shutdown)
- This is difficult!
  - Segments can be lost, duplicated, delayed, delivered out of order
  - Either side can crash
  - Either side can reboot

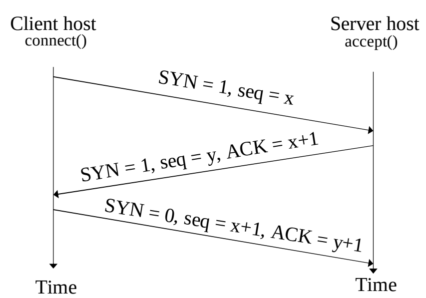
Three-way Handshake

Technique used by TCP for reliable connection establishment and termination

Example: Client initiates connection to server:

1. Client sends special TCP segment with SYN = 1, and a random initial sequence number
2. Server allocates TCP buffers and sends a connection-granted segment, with SYN = 1, ACK = client seqNum + 1, and a random sequence number
3. Client allocates TCP buffers and sends acknowledgement to server

3-way Handshake - Connection



3-way Handshake - Shutdown