Chapter 5: Transport Layer

Transport Layer - Why?

- Why do we need a Transport Layer?
  - With a Network Service Provider you can exchange packets between hosts (e.g. PCs), these hosts are uniquely identified by their network address (e.g. IP address).
  - As a user you may want to send and receive email, browse the web, logon to another host. So, you may want to run several programs or processes.
  - The transport layer allows for processes or applications to communicate with each other.

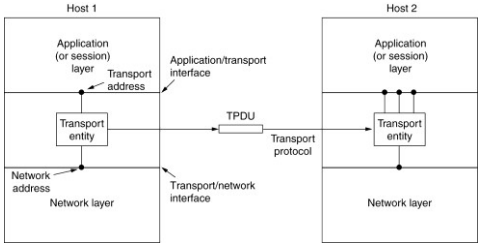
Contd..

- Networks (and the network layer) is under control of a network operator. The network service users have no control over it. So, if something goes wrong, a user can do nothing.
- The transport service is what users can add to improve the reliability of the service provided by the network.

Transport Layer Functions

- Functions that you can encounter in the Transport Layer are:
  - Error Handling
  - Flow Control
  - Multiplexing
  - Connection Set-up and Release
  - Congestion Handling
  - Segmentation and Reassembly
  - Addressing

Services Provided to the Upper Layers

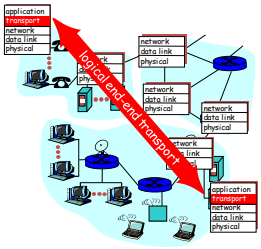


The network, transport, and application layers

- The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer.
- The software and/or hardware within the transport layer that does the work is called the transport entity.

Transport Layer Services and Protocols

- provide *logical communication* between application processes running on different hosts
- transport protocols run in end systems
  - send side: breaks application messages into segments, passes to network layer
  - receive side: reassembles segments into messages, passes to application layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



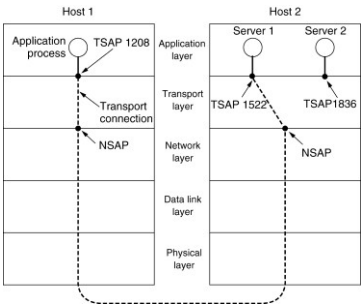
7

Internet transport services

- reliable, in-order unicast delivery (TCP)
  - connection setup
  - congestion
  - flow control
- unreliable ("best-effort"), unordered unicast or multicast delivery: UDP
- services not available:
  - real-time
  - bandwidth guarantees
  - reliable multicast

8

Addressing



TSAPs, NSAPs and transport connections

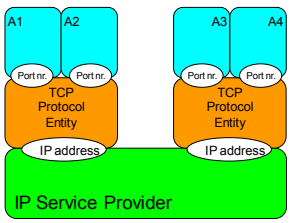
9

- Transport addresses to which processes can listen for connection requests are called ports.
- The generic term TSAP is used that means a specific endpoint (port) in the transport layer
- Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP.
- These connections run through NSAPs on each host

10

Transmission Control Protocol (TCP)

- TCP Protocol Functions:
  - Multiplexing
  - Error Handling
  - Flow Control
  - Congestion Handling
  - Connection Set-up and release
- TCP Transport Service
  - Connection Oriented (full duplex point-to-point connection between processes).
  - Reliable
  - In-sequence segment delivery



11

TCP Connections

- TCP identifies connections on the basis of endpoints:
  - IP address + port number
  - Often written as: IP-address : port-number, for instance: 130.89.17.3:80
- Two endpoints define a connection

12

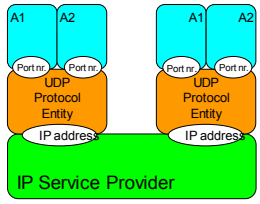
TCP Traffic

- You can see the TCP traffic statistics from and to you PC with the following command:
  - netstat -snp tcp or netstat –sp tcp
- This command also shows you the connections:
  - Local address (IP-address : port-number)
  - Foreign address (IP-address : port-number)
  - State of the connection

13

User Datagram Protocol (UDP)

- The UDP protocol functions are:
  - Multiplexing
  - Error Detection
- The UDP Service:
  - Is a connectionless service
  - Is unreliable
  - Has no in-sequence delivery guarantees



14

UDP Traffic

- You can view UDP traffic to and from your PC with the following command:
  - netstat -snp udp or netstat –sp udp

15

Why Would Anyone Use UDP?

- No delay for connection establishment
  - UDP just blasts away without any formal preliminaries
  - avoids introducing any unnecessary delays
- No connection state
  - No allocation of buffers, parameters, sequence numbers, etc.
  - easier to handle many active clients at once
- Small packet header overhead
  - UDP header is only eight-bytes long

16

Comparison of TCP and UDP

UDP - User Datagram Protocol	TCP - Transmission Control Protocol
• datagram oriented	• stream oriented
• unreliable, connectionless	• reliable, connection-oriented
• simple	• complex
• unicast and multicast	• only unicast
• No windows or ACKs	• Uses windows or ACKs
• Smaller header, less overhead	• Full header
• No Sequencing	• Sequencing
• useful only for few applications, e.g., multimedia applications	• used for most Internet applications
• network management (SNMP), routing (RIP), naming (DNS), etc.	• web (HTTP), email (SMTP), file transfer (FTP), terminal (telnet), etc.

17

Congestion Control

- **Open Loop Congestion Control:**
  - In open-loop congestion control, policies are applied to prevent congestion before it happens.
  - In these mechanisms, congestion control is handled by either the source or the destination
- **Closed-Loop Congestion Control**
  - Closed-loop congestion control mechanisms try to alleviate congestion after it happens.

18

- **Open Loop Congestion Control**
  - **Retransmission Policy**
    - If a sent packet is lost or corrupted, the packet needs to be retransmitted.
    - Retransmission in general may increase congestion in the network.
    - However, a good retransmission policy can prevent congestion.
  - **Window Policy**
    - The Selective Repeat window is better than the Go-Back-N window for congestion control.
    - In the Go-Back-N window, sends from last lost frame that may cause duplication and congestion.
    - The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

19

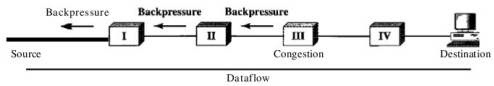
- **Acknowledgment Policy**
  - If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
  - Several approaches are used in this case.
    - A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires.
    - A receiver may decide to acknowledge only N packets at a time.
    - Sending fewer acknowledgments means imposing less load on the network.

20

- **Discarding Policy**
  - A good discarding policy by the routers may prevent congestion
    - For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.
- **Admission Policy**
  - Switches in a flow first check the resource requirement of a flow before admitting it to the network.
  - A router can deny establishing a virtual- circuit connection if there is congestion in the network or if there is a possibility of future congestion.

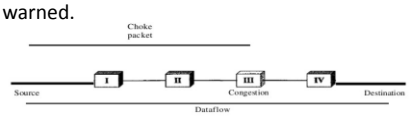
21

- **Closed-Loop Congestion Control**
  - **Back-pressure**
    - The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.
    - This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes and so on.



22

- **Choke Packet**
  - A choke packet is a packet sent by a node to the source to inform it of congestion.
  - Difference from backpressure - the warning is from one node to its upstream node, although the warning may eventually reach the source station.
  - In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has traveled are not warned.



23

- **Implicit Signaling**
  - In implicit signaling, there is no communication between the congested node or nodes and the source.
  - The source guesses that there is a congestion somewhere in the network from other symptoms.
    - For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested.
    - The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down.

24

- **Explicit Signaling**
  - In the choke packet method, a separate packet is used ; in the explicit signaling method, the signal is included in the packets that carry data.
  - **Backward Signaling**
    - A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.
  - **Forward Signaling**
    - A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

25

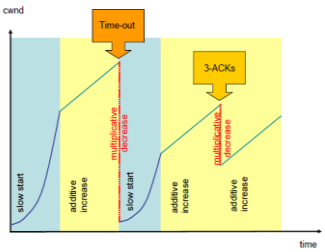
TCP congestion control: additive increase, multiplicative decrease (AIMD)

- *Approach: increase transmission rate (window size), probing* for usable bandwidth, until loss occurs.
  - additive increase: increase CongWin by 1 MSS every RTT until loss detected
  - multiplicative decrease: cut CongWin in half after loss

MSS=Maximum Segment Size

26

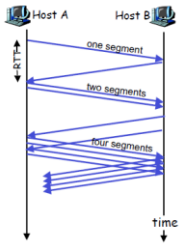
- It starts with slow start phase that exponentially increase CWND and continues to congestion avoidance threshold.
- When it reaches the congestion avoidance threshold then it additively increases CWND
- It follows multiplicative decreasing rate upon congestion detection.



27

TCP congestion control : TCP Slow Start

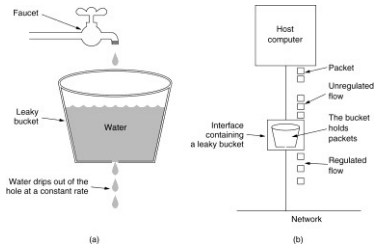
- A sender starts transmission with a slow rate
- Increases the rate as long as its rate is below a threshold.
- When the threshold is reached, the rate is decreased to avoid congestion.
- When connection begins, increase rate exponentially until first loss event:
  - double CongWin every RTT
  - done by incrementing CongWin **for** every ACK received



28

Traffic Shaping Algorithm

The Leaky Bucket Algorithm



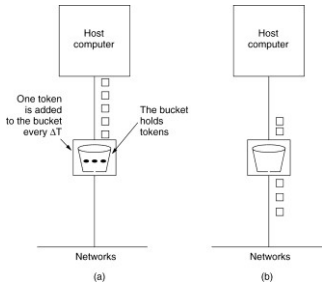
(a) A leaky bucket with water. (b) a leaky bucket with packets.

29

- Try to imagine a bucket with a small hole in the bottom, as illustrated in figure.
- No matter the rate at which water enters the bucket, the outflow is at a constant rate, *R, when there is any water in the bucket and zero when the bucket* is empty.
- Also, once the bucket is full to capacity, *any additional water entering* it spills over the sides and is lost.
- This bucket can be used to shape or police packets entering the network.
- Conceptually, each host is connected to the network by an interface containing a leaky bucket.
- If a packet arrives when the bucket is full, the packet must either be queued until enough water leaks out to hold it or be discarded.

30

The Token Bucket Algorithm



(a) Before. (b) After.

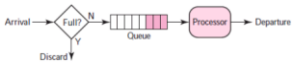
31

- A different but equivalent formulation is to imagine the network interface as a bucket that is being filled, as shown in figure.
- *Now, to send a packet we must be* able to take water, or tokens, as the contents are commonly called, out of the bucket (rather than putting water into the bucket).
- No more than a fixed number of tokens*, can accumulate in the bucket*
- *if the bucket is empty, we must* wait until more tokens arrive before we can send another packet.

32

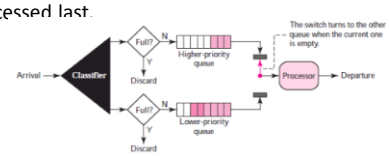
Techniques to improve QOS

- *Scheduling*
  - A good scheduling technique treats the different flows in a fair and appropriate manner.
  - Several scheduling techniques are designed to improve the quality of service. Three of them here:
    - ➢FIFO queuing,
    - ➢priority queuing, and
    - ➢weighted fair queuing
  - ➢FIFO



33

- ➢Priority queuing
  - ❖In priority queuing, packets are first assigned to a priority class.
  - ❖Each priority class has its own queue.
  - ❖The packets in the highest-priority queue are processed first.
  - ❖Packets in the lowest-priority queue are processed last.



34

- ➢Weighted fair queuing
  - ❖In this technique, the packets are still assigned to different classes and admitted to different queues.
  - ❖The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight.
  - ❖The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.

35

- *Traffic shaping*
  - ➢*Leaky Bucket and Token Bucket (Already Discussed)*
- *Resource reservation*
  - ➢A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on.
  - ➢The quality of service is improved if these resources are reserved beforehand.

36

- *Admission control*
  - Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications.
  - Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

37

Port

- Ports are conceptual "points of entry" into a host computer.
- Usually a service is associated with a port (e.g. http on port 80).
- Servers "listen on a port" for connection attempts.
- Ports provide one level of internet security.
- Each *process that wants to communicate with another process identifies itself* to the TCP/IP protocol suite by one or more ports.
- Popular applications have well-known ports
  - E.g., port 80 for Web and port 25 for e-mail
  - See http://www.iana.org/assignments/port-numbers

38

Port Contd..

- Ranges of Port Number
  - Server has a well-known port (e.g., port 80)
    - Between 0 and 1023 (requires root to use)
    - Allocated to server services by IANA
  - Registered Port
    - Between 1024 and 49151
    - Can be registered for services with the IANA
  - Client picks an unused ephemeral (i.e., temporary) port
    - Between 49152 and 65535
    - Used by client programs

39

- Some common services and port

Services	Listens on Port
HTTP	80
Telnet	23
SMTP	25
FTP Control	21
FTP Data	20
SSH	22
POP3	110
IMAP	143

40

Socket

- To the kernel, a socket is an endpoint of communication.
- To an application, a socket is a file descriptor that lets the application read/write from/to the network.
  - All Unix I/O devices, including networks, are modeled as files.
- Sockets are UNIQUELY identified by Internet address, end-to-end protocol, and port number.
- Clients and servers communicate with each by reading from and writing to socket descriptors.

41

Functions

- Define an "end- point" for communication
- Initiate and accept a connection
- Send and receive data
- Terminate a connection gracefully

Examples

- File transfer apps (FTP), Web browsers
- (HTTP), Email (SMTP/ POP3), etc…

42

Socket programming with TCP

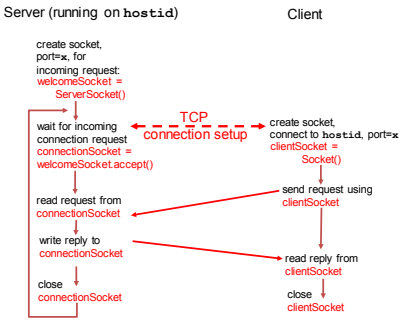
Client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

Client contacts server by:

- creating client-local TCP socket
- specifying IP address, port number of server process
- client establishes TCP connection to server

Client-server socket interaction: TCP



Socket programming *with UDP*

UDP: no "connection" between client and server

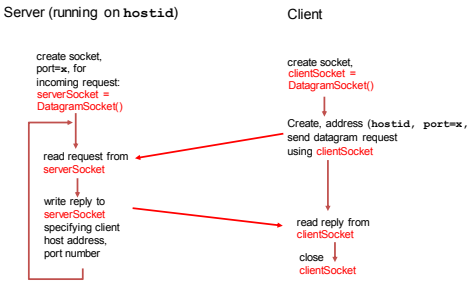
- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet to return uppercase sentence to sender

UDP: transmitted data may be received out of order, or lost

application viewpoint

UDP provides <u>unreliable</u> transfer of groups of bytes ("datagrams") between client and server

Client-server socket interaction: UDP



Thank You!!!!!