Transport-Layer Services, Multiplexing & Demultiplexing

1. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

    - sockets are identified by a 5-tuple (source IP, source port, dest. IP, dest. port, protocol) for TCP
        - 3-way handshake: socket needs 5-tuple
    - source IP and source port put into UDP header
        - less info needed: socket needs 3-tuple as reply not necessary
        - 3-tuple identifies socket; if a reply is needed needs to be handled differently
    - this allows the two segments to be differentiated (i.e. the IP address for different hosts will differ)

2. Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

    - yes: becomes responsibility of application to handle missing/out of order packets

3. Indicate whether TCP or UDP (or both or neither) provide the following services to appli- cations:

(a) Reliable data transfer between processes: TCP
(b) Minimum data transmission rate between processes: Neither: Controlled in the link layer
(c) Congestion-controlled data transfer between processes: TCP, sliding window
(d) A guarantee that data will be delivered within a specified amount of time: Neither; e.g. if physical connection is broken
(e) Preserve application-level message boundaries. That is, when a sender sends a group of bytes into a socket via a single send operation, that group of bytes will be delivered as a group in a single receive operation at the receiving application: UDP
(f) Guaranteed in-order delivery of data to the receiver: TCP (at application layer this is true)

4. Why does UDP exist? Would it not have been enough to just let the user processes send raw IP packets?

    - if you want to create multi-cast application-layer protocols
    - application needs finer-grained control over what data is sent and when
    - if application benefits from lower latency (no round trip delay) and can tolerate some data loss e.g. real time applications, video streaming

  - e.g. SNMP needs to work when network is under stress; congestion control may make this difficult
  - multiplexing/demultiplexing: ports

5. Both UDP and TCP use port numbers to identify the destination entity when delivering a message. Give two reasons for why these protocols invented a new abstract ID (port numbers), instead of using process IDs, which already existed when these protocols were designed?

- This would couple ports more tightly to operating system: process IDs may have already had their own protocols that differed from machine to machine. Wouldn't have been possible to implement a universal system
- well known port numbers established
- May expose information about underlying processes: poor for security (not really)
- Process IDs may change if a new process instance is spun up, but it would continue to listen through the same port number

6. What are the guarantees that a reliable data transfer must provide?

- guaranteed data integrity: both in terms of data order and absence of corrupted bits
- guaranteed order and delivery: through handling lost and corrupted packet redelivery

7. A process on host 1 has been assigned port p, and a process on host 2 has been assigned port q.

(a) Is it possible for there to be two or more TCP connections between these two ports at the same time?

  - No: the 5-tuple would be the same; single socket bound to a port

(b) Is it possible for there to be more than one TCP connection on port p of host 1 at a time?

  - Yes: 5-tuple will be different; e.g. Web servers