The Transport Layer

CS158a

Chris Pollett

Apr 30, 2007.

Outline

- Transport Layer Terminology
- Transport Protocols
  - Addressing
  - Connection Establishment
  - Connection Release

Transport Layer Terminology

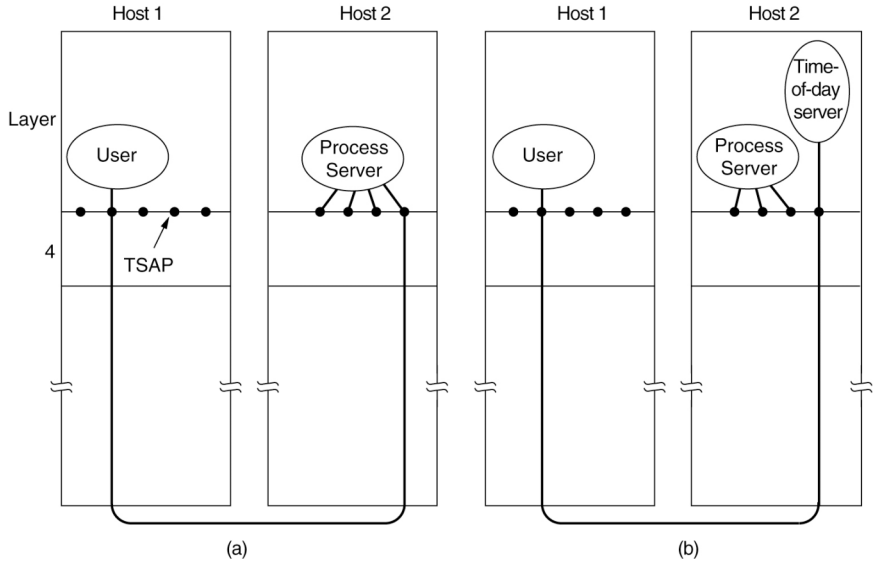
- Last day we gave a socket programming example in JAVA (book has Berkeley sockets in C example) and send the primitives the transport layer provides to the application layer are: LISTEN, CONNECT, SEND, RECEIVE, DISCONNECT.
- We will call the messages sent from transport entity to transport entities (**Transport Protocol Data Units) TPDUs**.
- So in our set-up:
  - A server might LISTEN at a port
  - Then a client CONNECT's by sending a CONNECTION REQUEST TPDU.
  - The server unblocks and sends a CONNECTION ACCEPTED TPDU.
  - At which point the two sides can SEND and RECEIVE TPDUs.
  - There are two ways to disconnect: **symmetric** and **assymetric**. In asymmetric either entity sends a DISCONNECT TPDU and both parties disconnect. In a symmetric disconnect if one party says DISCONNECT, it means they will no longer send but might still receive.

Transport Protocols

- The transport layer and the data link layer have many similarities: both deal with error control, sequencing, and flow control.
- There are some important differences however:
  - In the transport layer specific addressing of destinations is required.
  - The process of establishing a connection is more complicated in the transport layer.
  - Further, in the transport layer, packets may be delayed for a long time but rather than be permanently lost show up instead at an inopportune moment.
  - Flow control has to be more sophisticated in the transport layer as one might have many more connections.

Addressing

- The method normally used to determine where to send messages is to define transport addresses to which processes can listen for requests.
- These are called **ports** (on internet), AAL-SAPs (for ATM), or generically **TSAPs (Transport Service Access Points)**.
- We will call network layer address NSAPs.
- The reason one needs separate TSAPs and NSAPs, is that typically an NSAP only specifies a machine, not a process (aka server on that machine).



More on Addressing

- For some well known services (Web Server) it might be useful to have a fixed service listening on that port all the time.
- Typically, this is not done for most services. Instead, one has **process server (**for example inetd**)** that acts as a proxy for less heavily used servers. It listens to a set of ports.
- When a CONNECT is done, if no other server is waiting for the request, the request goes to the process serer which spawns the requested server, and gives it the desired connection.
- There are some situations where this is impractical (fileservers), in which case a **name server/directory server** approach (DNS) is used instead.
- Here the connector send a request to the name server and the name server responds with the TSAP of the desired server.

Connection Establishment

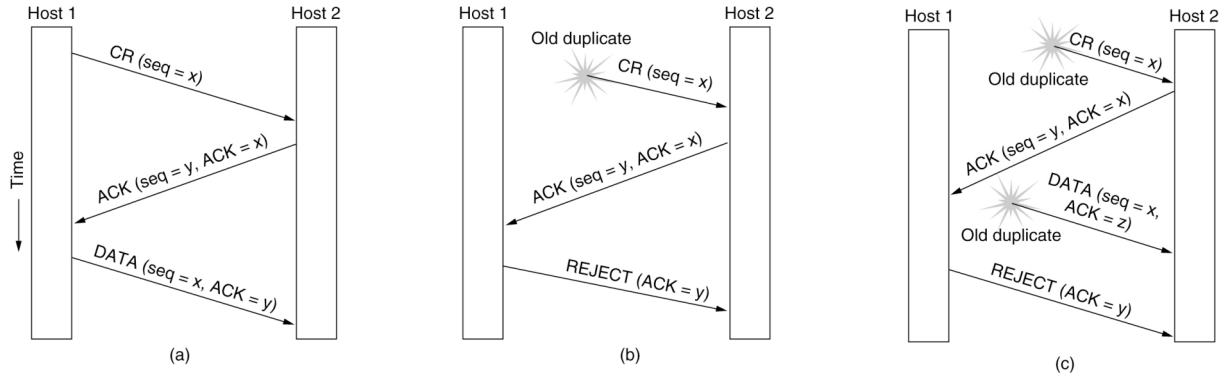
- In its simplest form this is done by sending a CONNECTION REQUEST TPDU which is answered with a CONNECTION ACCEPTED.
- However, problems can occur if the packets get delayed and so are re-sent. I.e., one gets duplicated packet issues.
- To solve this problem packets are given a fixed lifetime according to one of the following techniques:
  1. Restricted subnet design (any technique that prevents looping)
  2. Putting a hop counter in each packet
  3. Timestamping each packet
- Using one of the above schemes, let T be the length time for a packet and all of its acknowledgements to become dead.
- Tomlinson (1975) and Sunshine and Dalah (1978) give algorithms for establishing connections safely under the assumption that packets have a fixed lifetime. The setp-up is described on the next slide.

More on Connection Establishment

- Each host has a time-of-day clock, which is assumed to take the form of a binary counter which is incremented at uniform intervals.
- There are more bits in this counter than the number of bits in sequence numbers, and this clock never goes down.
- When a connection is established the low order k bits of the counter are used as the initial sequence number.
- The space of numbers is assumed large enough that by the time they wrap, old TPDUs with the same number are long since gone.
- Once the transport entities have agreed on the initial sequence number, and sliding window protocol can be used for flow control.
- If a host crashes one could wait T before letting the host send again.
- As T might be large, this is not usually done. Instead, we prevent the last sequence numbers from before the crash being used as initial sequence numbers for a time T. (Within T called **forbidden region**). We also require at most one TPDU per time tick to prevent a host from sending to many TPDUs and "catching up" with pre-crash TPDUs. One also has to check if one send to slow.
- What is left is to describe how to agree on the initial sequence number:

Three-Way Handshake

- Host 1 chooses a sequence number, sends a CONNECTION REQUEST TPDU to Host 2
- Host 2 replies with an ACK TPDU and its own initial sequence number.
- Finally, host 1 acknowledges host2's choice of initial sequence number in the first TPDU it sends.
- How this solves the problems of stray TPDUs in different situations is illustrated below:



Experiment in Class

- I showed how to telnet into a webserver on port 80 and download a web-page.