More on the Transport Layer

CS158a

Chris Pollett

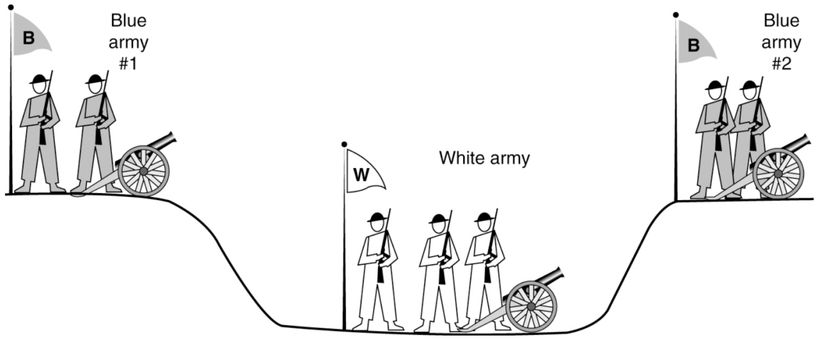
May 2, 2007.

Outline

- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery
- UDP
- TCP

Connection Release

- Releasing a connections also has some wrinkles to it.
- For example, if asymmetric release is being done, data can be lost if it was sent by the non-disconnect sender after the DISCONNECT was sent but before it arrived.
- This is solved by using a symmetric set-up.
- Unfortunately, the symmetric setting can fall victim to the **two-army problem**. If two blue armies attack separately will be destroyed, if attack together will win. Armies must send messengers on foot to communicate so some messages might be lost in the valley. How can they synchronize their attack? For connections, neither side will disconnect unless the other side does too.



More on Connection Release

- Turns out you can’t solve the the two-army problem with a three-way handshake or similar protocol.
- Still although not perfect, three-way handshakes are typically used: DISCONNECT REQUEST +start timer, DR ACK +start time, ACK back and release connection.
- If the scheme completes then connection release occurs, if no ACK then HOST 1 resends DR after timeout. If Host 2 times out before the ACK of the ACK come back and before a second DR then it releases connection. It assumes Host 1 got the DR ACK since Host 1 didn’t resend a DR. Thus, Host 2 assumes that the ACK of the ACK was lost.

Flow Control and Buffering

- Like the data link flow control is done using a sliding window protocol.
- Like the data link layer, frames are buffered at both the sender and receiver.
- In the transport layer one can have multiple connections, so the receiver has a choice to have separate buffers for each connection or to have all connections use the same buffer.
- Depending on whether all the TPDUs are the same size one can also choose between having fixed sized buffers or variable sized buffers, much as we did with the leaky bucket algorithm in the network layer.

Trade-off Between Source and Destination Buffering

- For low bandwidth bursty traffic (for example, a terminal session) it is better to buffer at the sender since the sender cannot be sure the receiver will be able to allocate a buffer at the random time the sender sends.
- On the other hand, for high bandwidth smooth traffic like file transfer, it makes sense to pre-allocate a window of buffers on the receiver before sending so that transfer can go as fast as possible.
- Traffics patterns change dynamically at the transport level and so should buffer capacity.
- To handle this, it is useful at the transport layer for the sender and receiver to be able to send to the other side buffer sizing requests.
- Initially, the sender requests a certain number of buffers based on its perceived needs. The receiver then grants as many of these as it can afford. Then every time the sender sends, it must decrement its allocation, finally stopping if it goes to 0. The receiver separately piggybacks both ACKs as well as messages saying a buffer has been allocated for the channel onto the reverse traffic.
- This scheme essentially means one has a variable sized window. The carrying capacity of such a set-up depends on the subnets capacity.
- Belsnes (1975) proposed an algorithm which adjusts the window size to match the network capacity. The sender's window is periodically adjusted to be the current estimate of c*r where c is the number of TPDUs/sec the network can handle and r is the roundtrip time for the TPDU and its ACK.

Multiplexing

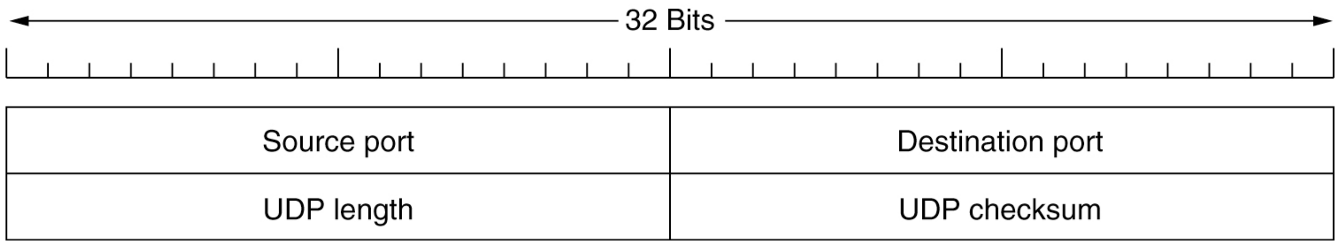
- There are two main types of multiplexing that occur in the transport layer:
  - **Upward multiplexing** -- this occurs when a TPDU arrives at the network address of its final destination and must be handed off to a particular process running there.
  - **Downward multiplexing** -- this might occur if the subnet uses virtual circuits and each has a fixed capacity. Several network connections between source and destination might be opened then and TPDUs might be split among these in a round-robin fashion. For example, in ISDN one has two 64 kbps lines, using downward multiplexing one can get an effective bandwith of 128kbps.

Crash Recovery

- If hosts and routers are subject to crashes then it is important to have schemes for recovering.
- For datagram services transport entities expect lost TPDUs so this is less of an issue that with connection oriented services.
- For the latter, one needs to re-establish a virtual circuit between the two end points and then use a querying mechanism to determine which TPDUs arrived correctly and which need to be re-sent.

UDP

- The internet has two main protocols in the transport layer: a connectionless protocol (UDP) and a connection-oriented protocol TCP).
- UDP stands for User Datagram Protocol has is described in RFC 768.
- UDP transmits **segments** consisting of an 8 byte header (see below) followed by a payload.
- UDP does not do flow control, error correction or retransmission. (This is up to the application that uses it).
- UDP is used by DNS, for remote procedure calls, where you have a simple request followed by a reply which answer some calculation. So if one doesn't get an answer one just asks again.
- Another use is RTP (Real-time transport protocol) used by Real to stream video over the web.



TCP

- TCP stands for **transmission control protocol**.
- It was designed to provide a reliable end-to-end byte stream over an unreliable internetwork.
- It is defined in RFC 793 and later revise in RFC 1122, RFC 1323.
- Each machine has a TCP entity which manages TCP streams to the IP Layer.
- The TCP entity accepts user data streams from local processes and breaks them up into pieces between 64K and 1460 bytes.
- To use TCP, the sender and receiver create end-points called sockets as we already discussed.
- Each socket has a socket number consisting of a IP address followed by a 16 bit number called a **port**. The port is the TCP name for a TSAP.
- Some well known ports are given on the next slide:

Well-known TCP ports

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

- Hel
- Dfjkfg ***fdkjfdg***
- fkdshjghdf$_{\text{fdjhgfjsdgf=fjhjdshf}}$fdhjs