1. Yes. At the socket interface, the operating system will provide the source IP to determine the origins.
2. 1. 1025 20
   2. 1025 20
   3. 20 1025
   4. 20 1025
   5. Yes.
   6. No.
3. Less delay, no connection establishment,  no retransmission, smaller segment header, no flow control and congestion control.
4. 1. $2^{32} * 8/336k = 28.4$hours
   2. $2^{32} * 8/30M = 19$ minutes
   3. $2^{32} * 8/2G = 17$ seconds
   4. Suppose a packet can stay in Internet for 1 minute at most, 32-bit sequence number is not enough for 12Gbps network. There might be two different TCP segment with same sequence number. To deal with this problem, TCP introduces a TIMESTAMP. Sequence number and Timestamp together uniquely identify a segment.
5. 1. With TCP, the application writes data to the connection send buffer and TCP will grab bytes without necessarily putting a single message in the TCP segment. TCP may put more or less than a singe message in a segment.  UDP encapsulates in a segment whatever the application gives it. If the application gives UDP an application message, this message will be the payload of the UDP segment.  Thus, with UDP, an application has more control of what data is sent in a segment.
   2. With TCP, due to flow control and congestion control, there may be significant delay from the time when an application writes data to its send buffer until when the data is given to the network layer. UDP does not have delays due to flow control and congestion control.
6. 1. 1. GBN: Initially, segments 1,2,3,4,5,6,7 are sent. Later, segments 3,4,5,6,7 are resent. So A sends 12 segments in total. B sends 1 ACK with sequence number 1, 5 ACKs with sequence number 2 and 5 ACKs with 3,2,4,5,6,7. Totally, B sends 12 ACKs.
      2. SR: 1,2,4,5,6,7 are sent once and 3 is sent twice. So A sends 8 segments in total, sequence number is 1,2,3,4,5,6,7,3. B sends 7 ACKs, with sequence number 1,2,4,5,6,7,3.
      3. TCP: A sends 8 segments, with sequence number 1,2,3,4,5,6,7,3. B sends 7ACKs, with sequence number 2,3,3,3,3,3,8.
   2. TCP. Because TCP uses fast retransmit and won't wait until timeout.
7. 1. In the intervals [1,6] and [23,26]
   2. By a triple duplicate ACK.
   3. By a timeout, for the window size dropped to 1.
   4. 32

5. 21, which is half of 42 at 16th round.
6. 14, which is half of 29 at 22th round.
7. At the 7th round. When reaching 6th round, 63 segments have been transmitted. When reaching 7th round, 96 segments have been transmitted.
8. Ssthresh will be 4, and window size will be 7.
9. Using TCP Tahoe, window size will be set to 1 at the 17th round and start a slow start. So window size will be 32 at the 21st round.

8. Process in host A will not continuously pass data to the socket because the send buffer will quickly become full as it holds only 1% of the file. Receiver won't overflow for receive buffer holds the entire file. Because there is no loss and acknowledgements are returned before timers expire, TCP congestion control does not throttle the sender.