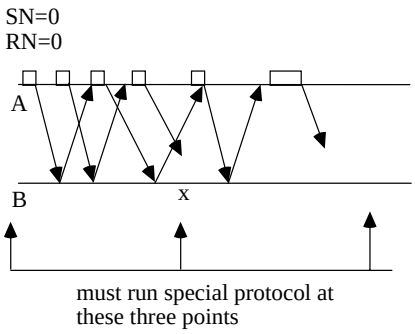
Lecture 7 - BG 2.7

So far, we just considered data transfer:



What if:

- B crashes at time x?
- The link crashes at time x?
- A crashes at time x?

Also:

- What should A (or B) do when the higher layer closes the connection?

Finally:

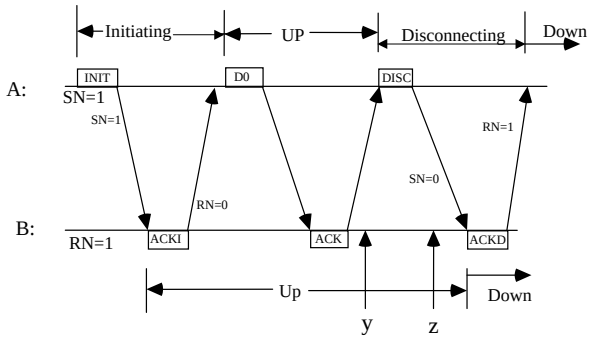
- What correctness criterion should an initialization/termination algorithm meet?

Correctness: delivered = prefix(sent)

Solution:

- Each node will partition its time axis into 2 or 4 periods
  - -- Initiating (Master only)
  - -- Up
  - -- Disconnecting (Master only)
  - -- Down
- Requests to initiate and disconnect come from higher protocol layer.
- We will use a protocol separate from the data transfer for initiation and disconnection
  - -- It uses stop & wait
  - -- It has 4 messages
    - --- INIT (SN = 1)
    - --- ACKI (RN = 0)
    - --- DISC (SN = 0)
    - --- ACKD (RN = 1)
  - -- The SN, RN's are different than those used in data transfer

- 1st protocol: Master/Slave
  Node A always decides when to initiate or disconnect.
- Again assume FCFS delivery (may be violated at layer 4)



- What happens if B or link crashes?
  - at Y?
    - -- A's go back n or selective repeat will retry to send data D_i until higher layer at A issues disconnect
  - at Z?
    - -- A just keeps sending DISC until link or B comes up, after which B will send ACKD & all continues as usual

- When does A send DISC?
  - When higher protocol layer decides to disconnect
  - In case of multiaccess link with polling
- Can any A->B data still be in transit when B goes from up to down?
  - No, due to FCFS delivery order.
- DISC has one other use, at DLC layer
  - To change which pair communicates in a multiaccess medium (e.g., polling)

2.7.3 - Balanced link initialization protocol (see Fig. 2.45 in [BG])

- Either endpoint can send INIT; both can overlap

- Solution: Always piggyback ACKI/ACKD on any INIT, DISC. In addition, send ACKI, ACKD as necessary.

- When node receives ACKI/ACKD in same frame as INIT/DISC, act on ACK first.

- "3 way handshake" -> used in transport protocols, but with out-of-order packet delivery.

- Sometimes 4-way occurs - see right end of Fig. 2.45 (BG).

- A node considers link up (down) if it is up (down) according to both A->B master/slave protocol and the B->A protocol.

- Correctness (safety property): Both sides reach the same state (either up or down).

2.7.4 Link initialization in the presence of Node Failures

Suppose a node does not retain state information when it crashes.

- Sometimes a node A sends data, crashes, comes up, sends unrelated data, and confuses ACKS for the earlier data as ACKS for the resent data. This can lead to data that is never reliable.

- See (BG) Fig. 2.46 (I believe "RN 0" should be "RN 1") In Fig. 2.4.6, D'0 is unrelated to D0. (Do belongs to a new, unrelated connection.)

- This happens if the time between INITS is not long enough to guarantee that any data & data-ack in transit are delivered.

- There is *no* protocol that can solve the problem, assuming link delay is unbounded.

Solutions:

1) Provide small non-volatile memory (1 bit)
2) Use timeout > max propagation delay between INITs
3) Use pseudo-random number in SN field (still no guarantee)