Computer Networks (Sheet #2)

Marius Gavrilescu

1. We need enough sequence numbers to distinguish between resending a whole window of frames with lost acks, and sending a whole window of new frames. With a 3-bit sequence number, we have 8 sequence numbers. As the send window size is 7, the receive window size can be at most 1.

2. If the channel is half-duplex, receiver can use back pressure to implement flow control. When the receiver falls behind, it jams the channel until it finishes processing the pending data.

   If the receiver has no way communicate with the sender, then the sender can only prevent data overruns by sending data slowly enough. This is only feasible if the sender has some idea of the speed the receiver can process data at.

   Buffers only help for bursts of data. If the average data rate is faster than the receiver can process, then any buffer would fill up after a while, rendering it useless.

3. When latency is key (audio/video communication, market data feeds) open-loop protocols could be superior. They are also useful for applications where reliability is not needed, such as a clock constantly receiving updates from the network (where any update overrides older ones).

4. This is done for error correction: if every bit is transmitted three times, the value that is seen at least two times is taken to be the bit's actual value. This means that for a bit to be corrupted two of the three values sent need to be flipped. Hamming distance between any two codewords is 3, so this code can detect at most 2 errors and correct at most 1 error.

   As bluetooth is used (among other things) for real-time transport of audio, the use of forward error correction is suitable here.

5. Speed is $2 * 10^8 m/s$, so the slot time for a 10km long network is $\frac{2}{2 \cdot 10^4} s = 0.1ms$. The minimum size of frame is the data rate multiplied by the slot time, so $1024, 10240, 104857$ bits.

   Overhead (not including slot) is $56 + 8 + 96 = 160$ bit times.

   Efficiencies are:
   10Mbps: $\frac{128}{1024+160} = 0.11$, $\frac{2048}{2048+160} = 0.93$, $\frac{32768}{32768+160} = 0.995$
   100Mbps: $\frac{128}{10240+160} = 0.01$, $\frac{2048}{10240+160} = 0.20$, $\frac{32768}{32768+160} = 0.995$
   1Gbps: $\frac{128}{104857+160} = 0.001$, $\frac{2048}{104857+160} = 0.02$, $\frac{32768}{104857+160} = 0.31$.

   To achieve the same efficiency we can use at most 1km at 100Mbps and 100m at 1Gbps.

6.
```
# Initial state
B 1 A
C 2 B
D 2 B
E 3 D

# B discovers A is down
B inf A

# B updates C
C inf B

# E updates C
C 4   E

# C updates B
B 5   C

# B updates D
D 6   B

# D updates E
E 7   D

# E updates C
C 8   E

# C updates B
B 9   C

# B updates D
D 10  B

# Final state
B 9  C
C 8  E
D 10 B
E 7  D
```

This is the count-to-infinity problem. Instead we would want to propagate the infinite-length route to every router. Split-horizon route advertisment would fix this, as E would not tell C about its route to A through C (and the rest of the updates would also avoid creating loops).

7. D knows about its links to A and E. Then A tells it about the AC and AD links, and E tells it about the EC, ED and EB links. Then C and B tell it about the CA, CE, CF, ED, EC, EB links. Finally F tells it about the FC link.

   With the complete set of link-state advertisments, D checks every link is reported from both sides, and then runs a variant of Dijkstra's algorithm to find the shortest paths to every other node. It then fills its routing table as follows:

   ```
   node via
   --------
   A    E
   E    E
   C    E
   B    E
   F    E
   ```

8. Not always. Take this graph:

   ```
   S---B--F
   |  /   |
   | /    |
   |/     |
   A------C
   ```

   Two routes from S to F are S, B, F and S, A, C, F. But if Dijkstra finds S, A, B, F as the shortest route, the graph will no longer be connected after removing these edges.