Data Link Layer Protocols

The data link layer provides service to the Network Layer above it:

- The network layer is interested in getting messages to the corresponding network layer module on an adjacent machine.
- The remote Network Layer peer should receive the identical message generated by the sender (e.g., if the data link layer adds control information, the header information must be removed before the message is passed to the Network Layer).
- The Network Layer wants to be sure that all messages it sends, will be delivered correctly (e.g., none lost, no corruption). Note that arbitrary errors may result in the loss of both data and control frames.
- The Network Layer wants messages to be delivered to the remote peer in the exact same order as they are sent.

Note: It is not always clear that we really want our data link layer protocol to provide this type of service. What if we run real-time applications across the link?

Nonetheless, the ISO reference model suggests that the data link layer provide such a service, and we now examine the protocols that provide such a service.

Motivation

Look at successive data link protocols of increasing complexity to provide reliable, in order message delivery to the network layer.

Environment

Assume DLL executes as a process with routines to communicate with the Network Layer above and the Physical Layer below.

Deal with messages (packets)—bit strings

frames are the unit of transmission. Consists of data plus control bits (header information).

```
procedure wait (var event: EvType);

{wait for an event; return event type in 'event'}
```

Look at data structures in Fig. 3-8

Utopia/Unrestricted Simplex Protocol

Assumptions:

- data transmission in one direction only (simplex)
- no errors take place on the physical channel
- the sender/receiver can generate/consume an infinite amount of data.
- always ready for sending/receiving

Figure 3.9

Simplex Stop-and-Wait Protocol

Drop assumption that receiver can process incoming data infinitely fast

stop-and-wait—protocols where the sender sends one frame and then waits for acknowledgement.

In this protocol, the contents of the acknowledgement frame are unimportant.

Data transmission is one directional, but must have bidirectional line. Could have a half-duplex (one direction at a time) physical channel.

Figure 3.10

Simplex Protocol for a Noisy Channel

What if the channel is noisy and we can lose frames (checksum incorrect).

Simple approach, add a time out to the sender so it retransmits after a certain period and retransmit the frame.

Scenario of what could happen:

- A transmits frame one
- B receives A1
- B generates ACK
- ACK is lost
- A times out, retransmits
- B gets duplicate copy (sending on to network layer)

Use a *sequence number*. How many bits? 1-bit is sufficient because only concerned about two successive frames.

Positive Acknowledgement with Retransmission (PAR)—sender waits for positive acknowledgement before advancing to the next data item.

Figure 3-11.

How long should the timer be? What if too long? (inefficient)

What if too short? A problem because the ACK does not contain the sequence number of the frame which is being ACK'ed.

Scenario:

- A sends frame zero
- timeout of A
- resend frame A0
- B receives A0, ACKS
- B receives A0 again, ACKS again (does not accept)
- A gets A0 ACK, sends frame A1
- A1 gets lost
- A gets second A0 ACK, sends A2
- B gets A2 (rejects, not correct seq. number)
- will lose two packets before getting back on track (with A3,1)

One Bit Sliding Window Protocol

Two-way communication. One-way is not realistic.

Have two kinds of frames (*kind* field):

1. Data
2. Ack (sequence number of last correctly received frame)

piggybacking—add acknowledgement to data frames going in reverse direction.

For better use of bandwidth. How long to wait for outgoing data frame before sending the ACK on its own.

Example of a *sliding window* protocol. Contains a sequence number whose maximum value *MaxSeq* is $2^n - 1$.

For stop-and-wait sliding window protocol $n = 1$.

- Essentially protocol 3, except ACKs are numbered, which solves early time out problem.
- protocol works, all frames delivered in correct order
- requires little buffer space
- poor line utilization (next page)

Problem with stop and wait protocols is that sender can only have one unACKed frame outstanding.

Example:

- 1000 bit frames
- 1 Mbs channel (satellite)
- 270 ms propagation delay

Frame takes 1ms to send. With propagation delay the ACK is not seen at the sender again until time 541ms. Very poor channel utilization. Solution:

1. We can use larger frames, but the maximum size is limited by the bit error rate of the channel. The larger the frame, the higher the probability that it will become damaged during transmission.
2. Use *pipelining*: allow multiple frames to be in transmission simultaneously.

Pipelining

Sender does not wait for each frame to be ACK'ed. Rather it sends many frames with the assumption that they will arrive. Must still get back ACKs for each frame.

Example 1: Use a 3-bit sequence number (0-7). Now we can transmit 7 frames (seq. nr. 0-6) before receiving an ACK.

Example 2: What if we allow the sender to send 8 (0-7) instead of 7 (0-6) frames?

Potential problem of window sizes (receiver window size of one):
MaxSeq is 7 (0 through 7) is valid. How big can sender window be?

1. Send 0-7.
2. Receive 0-7 (one at a time) and send ACKS
3. All ACKS are lost
4. Message 0 times out and is retransmitted
5. Receiver accepts frame 0 (why?—because that is next frame) and passes it to NL.

Window Size Rule: The sender window size (number of buffers) plus the receiver window size (number of buffers) must be $<= 2^n$ where n is the number of bits in the sequence number.

Example 3: Provides more efficient use of space and works well if we follow the window size rule and do not get errors. What if an error occurs?

What if 7 frames transmitted (seq nr 0-6), and sequence number 1 has an error. Frames 2-6 will be ignored at receiver side? Sender will have to retransmit.

Two strategies for Window size:

1. Go back n—receiver's window size of one (protocol 5)
2. selective repeat—receiver's window size larger than one, generally the sender and receiver have the same window size. (protocol 6)

Tradeoff between bandwidth and data link layer buffer space.

In either case will need buffer space on the sender side. Cannot release until an ACK is received.

Use a timer for each unACK'ed frame that has been sent.

Can enable/disable network layer because no longer assume that network layer is ready (*NetworkLayerReady* event).

Protocol 6, Selective Repeat

Summarize, not details

Example 4: Look at same example using protocol 6 with sender and receiver window size each equal to 4.

Now if 4 frames transmitted (seq nr 0-3), and sequence number 1 has an error. Frames 2-3 will *not* be ignored at receiver side, but they can be buffered.

What happens when Frame 1 times out and is retransmitted? Upon reception, the receiver can pass frames 1, 2 and 3 up to the network layer and update its receiver window.

Look at Sliding Window Protocols notes.

Protocol Performance

What is the channel efficiency of a stop-and-wait protocol?

- F = frame size = $D + H$ = data + header bits
- C = channel capacity (bps)
- I = propagation delay and IMP service time (seconds)
- A = ack size (bits)

Draw picture

time between frames: $F/C + 2I + A/C$

time spent sending data: D/C

efficiency:

$$\frac{D/C}{F/C + 2I + A/C} = \frac{D}{F + 2IC + A} = \frac{D}{D + H + 2IC + A}$$

Examples of the Data Link Layer

High-level Data Link Control

Adopted as part of X.25.

- Bit oriented (uses bit stuffing and bit delimeters)
- 3-bit sequence numbers
- up to 7 unack'ed frames can be outstanding at any time (how big is the receiver's window? one)
- ACK's the "frame expected" rather than last frame received (any difference between the two? No, as long as same convention).

Basically, Go Back N protocol.

Data Link Layer in the Internet

Point-to-point lines

1. between routers over leased lines
2. dial-up to a host via a modem

Two protocols used:

SLIP—Serial Line IP

Older protocol that just adds a framing byte at end of IP packet (with appropriate character stuffing). Some problems:

1. no error detection/correction
2. supports only IP
3. each side must have its own IP address (not enough for all homes) or some versions do it dynamically.
4. no authentication
5. not an approved standard, many version exist

PPP—Point-to-Point Protocol

a Standard (RFCs 1661-1663)! Can be used for dial-up and leased router-router lines.

Provides:

1. Framing method to delineate frames. Also handles error detection.
2. Link Control Protocol (LCP) for bringing lines up, negotiation of options, bringing them down. These are distinct PPP packets.
3. Network Control Protocol (NCP) for negotiating network layer options.

See Fig. 3-28.

Similar to HDLC, but is character-oriented.

It does not provide reliable data transfer using sequence numbers and acknowledgements as the default. Reliable data transfer can be requested as an option (as part of LCP).

Data Link Layer in ATM

Transmission Convergence sublayer (refer back to ATM reference model).

Physical layer is T1, T3, SONET, FDDI.

This sublayer does header checksumming and cell reception.

Header Checksum

5-byte header consists of 4 bytes of virtual circuit and control stuff.

Checksum 4 bytes of header information and store in 5th byte. Use CRC checksum $x^8 + x^2 + x + 1$ and add a constant 01010101 bit string.

Low probability of error (likelihood of fiber) so keep it cheap to checksum. Upper layers can do payload if they like.

8-bit checksum field is called *Header Error Control (HEC).*

Transmission

May have to output dummy cells in a synchronous medium (must send cells at periodic times). Use *idle cells.* Also have *operation and maintenance (OAM)* cells. Exchange control and other information.

Cell Reception

Drop *idle cells*, pass along *OAM* cells.

Need to generate framing information for underlying technology, but no framing bits! Use a probablistic approach of matching up valid headers and checksums in a 40-bit window.

See Fig. 3-30. Have a state-transition diagram where we are looking for δ consecutive valid headers.

If a bad cell received (flipped bit) do not immediately give up on synchronization.

Important: TC sublayer must know about header information above it. Violates layering principle to make things work.