Disclaimer : These notes are jottings which I prepared to guide me through the lecturing and remind me of which points are important.
They were never intended to be coherent handouts for student distribution.

Peter Fenwick March 2006

Notes §1.4.3 Continuous RQ

Figure 1.26

- Requires Duplex transmission
- Each ACK is delayed by processing delays, queue delays, etc
- V(S) has number of *next frame to send*
- V(R) has number of the *expected frame* (some protocols have the last received frame – watch this!)
- A frame is removed from the *retransmission* list as soon as it is acknowledged.
- A frame is removed from the *receive* list as soon as it has been received in sequence
- V(S) and V(R) step on, without limit, as transmission proceeds.
- A timeout is an essential part of the reception protocol.

Figure 1.27 Corrupted I-frame (N+1)

- ACK(N) is still sent while I(N+1) is being received
- Frame I(N+1) is ignored. (This is the most general case, if intermediate nodes detect an error or overload and discard the frame. If the bad frame is recognised by its destination, a NAK may be sent immediately, but only with point-to-point transmission.)
- When the unexpected frame I(N+2) is received it triggers a NAK(N+1) to signal the I (N+1) is missing, but frame I(N+2) is accepted and queued. (*Retransmission* state)
- P continues to send frames until it receives the NAK(N+1); it then resends I(N+1) and waits for the ACK(N+1). There may be many unacknowledged frames in transit over a long fast link.
- When I(N+1) is received correctly, the acknowledgment ACK(N+1) acknowledges all received frames, including I(N+4).
- P will then send I(N+5), to be acknowledged by ACK(N+6)

Figure 1.27 Corrupted ACK-frame

- The sender (P) does not know whether the I-frame has failed or the ACK has failed, but

must resend I(N).

- S now receives a *second* copy of I(N), which it recognises as a duplicate from its receive sequence variable V(S).
- But S must still acknowledge I(N) by sending (a second) ACK(N) so that P knows that the message has been received.

- **Figure 1.28 Go-back-N**
- Now, if the sender receives an out-of-sequence frame it just discards it and sends a NAK for the expected frame. Discarding continues until the expected frame is received.
- The sender repeats the first unacknowledged frame, waits for the ACK and then continues, repeating the frames sent but not acknowledged.
- Note that a timeout may force a NAK if there is no traffic for a while.

Figure 1.29 Flow Control

- The presence of some unacknowledged frames is usually described by defining a *sliding window*, both *send window* and *receive window*.
- The send window advances along the sequence of frames, dividing it into 3 groups separated by the Upper Window Edge (*UWE*) and Lower Window Edge (*LWE*).
- Frames with $N < LWE$ have been both sent and acknowledged and are of no further concern to the sender.
- Frames within the window $LWE <= N <= UWE$ have been sent but not acknowledged; they must be kept in the transmit buffer.
- Frames with $N > UWE$ are waiting to be sent and still unknown to the sender.
- We also need a receive window. For IDLE-RQ and Go-Back-N it needs only one message buffer because messages are acknowledged in order and there are never gaps in the sequence.
- Selective Repeat needs a receive window equal in size to the sliding window to handle out-of-sequence reception. The collection of message buffers is often called a "reassembly buffer".

Figure 1.30 Sequence Numbers

- We do not need an unlimited sequence number (although TCP effectively uses one).
- If we can send *X* frames in the time to send a frame and get a reply, it is enough to take the sequence number modulo *X*. Thus HDLC, to be studied in a later section, uses a 3-bit

number (modulo 8), extendable to 7 bits (modulo 127) at circuit establishment.

- Idle RQ needs only 2 frame identifiers
- With a window size *K*, Go-Back-N requires a maximum of *K*+1 separate sequence numbers.
- With a window size *K*, Selective Repeat requires a maximum of 2*K*+1 separate sequence numbers.

General comments

- Frames may be lost because of transmission errors or because IP discards them because of congestion.
- Full checking and acknowledgment as described here has migrated up the protocol stack as technology advanced
- Early protocols such as SDLC/HDLC were designed for slow and unreliable links (say 9600 bps and error prob 10^{-5}.) This mandated short frames (say 1000 bits, 128 bytes giving say 1 error in 100 frames or 10 seconds)
- Later protocols (LANs and fibre-optic trunks) have error probabilities more like 10^{-10}. Full error recovery at the lower levels is just too much wasted effort, say 1 error in 1 million frames and the ACK/NAK overheads are just not justified.
- So if a frame has an error, just throw it away.
- Full acknowledgment is always provided at a high level (such as TCP) and that can recover from errors. We don't need error recovery at lower levels.
- With LANs we can't have an ACK/NAK system anyway because many nodes may send and listen to a LAN. An error may be in an address (we don't know).
  – In the destination address, who should reply?
- – in the destination address, who should be sent the NAK?
- So a LAN node can only ignore any bad message that it receives.

Notes §1.4.6 Layered Architecture, Fig 1.31

- The layering allows different functions to be separated; as long as the interfaces are well-defined the implementations do not matter.
- Transmission goes *down the stack*; reception goes *up the stack*
- The source has a timer, armed when a frame is sent, to detect "no reply".
- Communication between layers is through queues (FIFO buffers)

Notes §1.4.7 Protocol Specification, Fig 1.32 ff

- Model a protocol as a Finite State Machine or Automaton, needing
    1. A collection of *states*, describing a general situation and each waiting on one or more events
    2. *Events*, usually from some external action such as a message arrival, which may cause actions dependent on the current state.
    3. *Actions*, initiated from a state in response to an event, such as sending a message or moving to another state.
    4. The whole is supported by *predicates*, essentially global variables, that carry information between states and from earlier to later times.

Figure 1.33

- The IDLE RQ primary waits in an IDLE state.
- If it receives an ACK or a NAK this is an error, increment an error count and return to IDLE.
- If it receives an LDATAreq then send a frame, start timer and increment Vs. Go to WTACK state.
- In WTACK, resend the frame if either timer expired or NAK received; restart timer and increment retransmission count.
- In WTACK, if ACK received, stop the timer and reset retransmission count; return to IDLE state.

Figure 1.34

- Trivial really –
    - If IRCVD with error, send NAK , return to Wait for I-Frame
    - If IRCVD no error, signal data received, send NAK, wait for next I-Frame

Figure 1.35

- The middle, between the source and destination blocks, may be a large network with many links, routers and switches. All that is necessary is that it delivers messages accurately.
- Time increases (gets later) as we go down the diagram.
- We start with “Connection-oriented” actions –

- L_CONNECT.request –
    - SETUP request is sent over network; destination receives L_CONNECT indication.
    - UnnumberedAcknowledgement is returned and appears at Source as L_CONNECT.confirm
    - The link layers set up variables to maintain the connection.
- L_DATA.request –
    - Data is sent as an I-frame; it appears as an L_DATA.indication at destination. An ACK is returned and absorbed by the Source Link Layer – user is NOT notified.
    - L_DISCONNECT.request is similar to L_CONNECT.request in message exchange, but of course the variables that establish the connection are reset.
-
    - Connectionless mode –
        - L_UNITDATA.request is just sent in the hopes that it will arrive and appear as L_UNITDATA.indication at the destination. What could be simplerÉÉ
- User connections (TCP) tend to be connection oriented, but lower levels (IP etc) are now usually best-effort.

Figure 1.36 HDLC (High-Level DataLink Control)

- Devised in 1970's by IBM as SDLC (Synchronous DataLink Control) and adopted as a standard, but renamed HDLC.
- Allows a sliding window, 3 bit (optional 7 bit) sequence numbers.
- Frames are normally about 128 octets (1024 bits) – quite small.
- Note use of Flags (01111110) and bit stuffing, with a 0 forced after *every* consecutive 5 1s in data (011111*x* ... → 0111110*x*...)
- Information and Supervisory Frames include a mysterious P/F (Poll/Final) bit, which really means "reply if P/F = 1".
    - Long data messages are broken into several Information Frames; the last has P/F=1 (F=1 final frame) and all others have P/F=0 (F=0 intermediate frame)
    - Supervisory frames use PF=1 (Poll=1) to require a response.
- Most message exchanges use **I-frames** for data and **Supervisory frames** to coordinate –
    - RR Receive Ready – frame received – positive ACK, may be sent regularly

to announce status
    - RNR positive ACK, but buffer full etc – wait until RR is sent
    - REJ Reject frame N(R) and all following ones (simple receiver cannot handle out-of-sequence frames)
    - SREJ Selective Reject – reject only frame N(R) (receiver can handle frame reassembly)
- **Unnumbered supervisory frames** set operational mode (commands to Set Normal Response Mode, set Asynchronous Balanced Mode) UA (unnumbered acknowledgment), DISC (disconnect) and bad control frame (FRMR – frame reject)

- Some vestiges of HDLC remain in the LLC layer signalling for LANs. The "address" indicates the protocol type and data is transferred as unnumbered information — BEWARE the bits here are LSB to the left and a UI control octet becomes 0x03 for most other work!

Figure 1.38 Protocol Stacks

- Each layer provides service primitives for the layer above and uses the service primitives of the layer below.
- Logically each layer communicates with its "peer" or equal layer at the other end of the link.
- Physically communication is by messages being sent down the transmission stack and being passed up the reception stack.
- As a message is passed down a transmission stack it will get layer-specific headers added to it. Sometimes messages will be split (IP fragmentation) and very occasionally may be combined to send many short messages as one longer one to reduce transmission overheads.
- As a message is passed up a reception stack headers will be stripped off at each stage and used to control processing at that level; the remnant will be passed up.

Protocols

Section § 2.1 Telephones

- Read general description §2.1 – 2.2.1
- Fig 2.5 discuss modulation – ASK, FSK, PSK
- Fig 2.6 Constellation diagrams

- Fig 2.8 DB-25 connections – shield, TxData, RxData, RTS, CTS, DSR, CD and DTR. Mention TxClk, RxClk, ignore most other signals
- Fig 2.9 Emphasise that this is a protocol, just like the message protocols.
- Section 2.2.3 Ignore ISDN
- Fig 2.15 Briefly describe T1 1.544 Mbps frame format
- Fig 2.16 Ditto ISDN (E1) 2.048 Mbps format
- Fig 2.19 Should know STM-1 and STM-4 (OC-3 and OC-12, 155.52 Mbps and 622.08 Mbps).

Section § 2.5 Broadband Modems & ADSL

- Ignore ISDN stuff, read through ADSL introduction
- Fig 2.27 different connection alternatives and data rates
- Fig 2.29 briefly discuss modulation methods – multiple carriers graded bit rate v carrier freq, etc.