Applied Math 254: Computer Networks

Notes for Class 5: Node-to-Node Transmission Algorithms

John M. McQuillan and David C. Walden

March 1975

1. The Network Circuits

We first consider some of the important characteristics of the circuits used in the network.

1.1 Bandwidth

The bandwidth of the network circuits is likely to be their most important characteristic. It defines the traffic-carrying capacity of the network, both in the aggregate and between any given source and destination. What is less obvious is that the bandwidth (and hence the time to clock a packet out onto the line) may be the main factor determining the transit delays in the network. The minimum delay through the network depends mainly on circuit rates and lengths, and additional delays are largely accounted for by queueing delay, which is directly proportional to circuit bandwidth. These two factors lead to the general observation that the faster the network lines, the longer the packet should be, since long packets have less overhead and permit higher throughput, while the added delay due to length is less important at high circuit rates. In addition, more packet and message buffering is required when higher speed circuits are used.

1.2 Delay

The major effect of circuits with appreciable delay is that they require more buffering in the nodes to keep them fully loaded. That is, the node must maintain more packets in flight at once over a circuit with longer delay. This effect may be so large (a circuit using a satellite has a delay of a quarter of a second) as to require significantly more memory in the nodes (McQuillan 72). This memory is needed at the nodes connected to the circuit to permit sufficient packet buffering for node-to-node transmission using the circuit. The subtle point is that additional buffering is also required at all nodes in the network that may need to maintain high source-to-destination rates over network paths which include this circuit. If they are to provide maximum throughput, they need sufficient message buffering to keep the entire network path fully loaded.

1.3 Reliability

Traditionally, the telephone carriers have quoted error rates in the following manner: "No more than an average of 1 bit in 10 to the 6th bits in error." This definition is not entirely adequate for packet switching, though it may be for continuous transmission. For packet switching, the average bit error rate is less interesting than the average packet error rate (packets with one or more bits in error). For example, ten bits in error in every tenth packet is a 10%

packet error rate, while one bit in error in every packet is a 100% packet error rate, yet the two cases have the same bit error rate.

An example of an acceptable statement of error performance would be as follows:

> The circuit operates in two modes. Mode 1: no continuous sequence of packet errors longer than two seconds, with the average packet error rate less than one in a thousand. Mode 2: a continuous sequence of errors longer than two seconds with the following frequency distribution:
>
> | | | |
> |---|---|---|
> | > | 2 seconds | no more often than once per day |
> | > | 1 minute | no more often than once per week |
> | > | 15 minutes | no more often than once per month |
> | > | 1 hour | no more often than once per 3 months |
> | > | 6 hours | no more often than once per year |
> | > | 1 day | never |

While the figures above may seem too stringent in practice, the mode 1 bit error rate is actually quite lax compared to conventional standards. In any case, these are the kinds of behavior descriptions needed for intelligent design of packet-switching network error control procedures. Therefore, it is important that the carriers begin to provide such descriptions.

The packet error rate of a circuit has two main effects. First, if the rate is high enough, it can degrade the effective circuit bandwidth by forcing the retransmission of many packets. While this is basically a problem for the carrier to repair, the network nodes must recognize this condition and decide whether or not to continue to use the circuit. This is a tradeoff between reduced throughput with the circuit and increased delay and less network connectivity without it. Before the circuit can be used, it must be working in both directions for packets and for control information like routing and acknowledgments, and with a sufficiently low packet error rate.

The second effect of the error rate is present even for relatively low error rates. It is necessary to build a very good error-detection system so that the users of the network do not see errors more often than some specified extremely low frequency. That is, the network should detect enough errors so that the effective network error rate is at least an order of magnitude less than the Host error or failure rate. A usual technique here is a cyclic redundancy check on each packet. This checksum should be chosen carefully; to first order, its size does not depend on packet length*

and it should be quite large, for example 24 bits for 50-Kbs lines and 32 bits for multi-megabit lines or lines with high error rates.

---

*Assuming that the probability of packet error is proportional to the product of packet length and bit error rate, the checksum length should be proportional to the log of the product of the desired time between undetected errors, the bit error rate, and the total bandwidth of all network circuits.

2. Node-to-Node Transmission Procedures

In this section we discuss some of the issues in designing node-to-node transmission procedures that is, the packet processing algorithms. We touch on these points only briefly since many of them are simple or have been discussed previously. Note that many of these issues occur again in the discussion of source-to-destination transmission procedures.

2.1 Buffering and Pipelining

As we noted in discussing memory requirements, the amount of node-to-node packet buffering needs to equal the product of the circuit rate times the expected acknowledgment delay in order to get full line utilization. It may also be efficient to provide a small amount of additional buffering to deal with statistical fluctuations in the arrival rates, i.e., to provide queueing. These requirements imply that the nodes must do bookkeeping about multiple packets, which raises the several issues discussed next.

2.2 Error Control

We have discussed many of the aspects of node-to-node error control above: the need for a packet checksum, its size, the basis of the acknowledgment/retransmission system, the decision on whether the line is usable, and so on. These procedures are critical for network reliability, and they should therefore run smoothly in the face of any kind of node or circuit failure. Where possible, the procedures should be self-synchronizing; at least they should be free from deadlock and easy to resynchronize (McQuillan 72).

2.3 Storage Allocation and Flow Control

Storage allocation can be fairly simple for the packet processing algorithms. The sender must hold a copy of the packet until it receives an acknowledgment; the receiver can accept the packet if it is without error and there is an available buffer. The receiver should not use the last free buffer in memory, since that would cut off the flow of control information such as routing and acknowledgments. In accepting too many packets, there is also the chance of a storage-based deadlock in which two nodes are trying to send to each other and have no more room to accept packets. This is explained fully in (Kahn 71).

The above implies that the flow control procedures can also be fairly simple. The need to buffer a circuit can be expressed in a quantitative limit of a certain number of packets. Therefore, the node can apply a cut-off test per

line as its flow control throttle. More stringent rules can be used, but may be unnecessary.

2.4 Priority

The issue of priority in packet processing is quite important for network performance. First of all, the concept of two or more priority levels for packets is useful in decreasing queueing delay for important traffic. Beyond this, however, careful attention must be paid to other kinds of transmissions. Routing messages should go with the highest priority, followed by acknowledgments (which can also be piggy-backed in packets). Packet retransmissions must be sent with the next highest priority, higher than that for first transmission of packets. If this priority is not observed, retransmissions can be locked out indefinitely. The question of preemptive priority (i.e., stopping a packet in mid-transmission to start a higher priority one) is one of a direct tradeoff of bandwidth against delay since circuit bandwidth is wasted by each preemption.

2.5 Packet Size

There has been much thought given in the packet-switching community to the proper size for packets. Large packets have a lower probability of successful transmission over an error-prone telephone line (and this drives the packet size down), while overhead considerations (longer packets have a lower percentage overhead) drive packet size up. The delay-lowering effects of pipelining become more pronounced as packet size decreases, generally improving store-and-forward delay characteristics; further, decreasing packet size reduces the delay that priority packets see because they are waiting behind full length packets. However, as the packet size goes down, effective throughput also goes down due to overhead. Metcalfe has previously commented on some of these points (Metcalfe 73).

Kleinrock and Naylor (Kleinrock 74) recently suggested that the ARPA Network packet size was suboptimal and should perhaps be reduced from about 1000 bits to 250 bits. This was based on optimization of node buffer utilization for the observed traffic mix in the network. However, in (Crowther 74), we point out that the relative cost of node buffer storage vs. circuits is possibly such that one should not try to optimize node buffer storage. The true trade-off which governs packet size might well be efficient use of phone line bandwidth (driving packet size larger) vs. delay characteristics (driving packet size smaller). If buffer storage is limiting, should just buy more. Further, it is probably true that if one is trying for high bandwidth utilization, buffer size must be large. That is, high

bandwidth utilization probably implies the use of large packets, which implies full buffers; when idle, the buffer size does not matter.

As noted above, the choice of packet is influenced by many factors. Since some of the factors are inherently in conflict, an optimum is difficult to define, much less find. The current ARPA Network packet size of about 1000 bits is a good compromise. Other packet sizes (e.g., the 2000 bits used in several other networks) may also be acceptable compromises. However, note that a 2000-bit packet size generally means a factor of two increase in delay over a 1000-bit packet size, because even high priority short packet will be delayed behind normal long packets which are in transmission at each node. The use of preemptive priority might make longer packet sizes efficient.

Davies and Barber (Davies 73) are often quoted as recommending a minimum length "packet" of about 2000 bits because they have concluded that most of the messages currently exchanged within banks and airlines fit nicely in one packet of this size. To clarify this point, we note that they use the term "packet" for the unit of information we call a "message" and thus are not actually addressing the issue of packet size. We discuss message size below.

2.6 The ARPA Network IMP-to-IMP Transmission Control Procedure

We now take a close look at the algorithm used in the ARPA Network for IMP to IMP transmission control. As has been noted elsewhere, the inter-IMP modem interface hardware has the capability of generating checksums for outgoing packets and checking the checksums on incoming packets. This allows packets which are damaged in transmission to be detected and discarded without acknowledgement. Packets correctly received are acknowledged. A good IMP to IMP transmission control algorithm must detect errors, acknowledge good transmissions, and provide retransmission in the event of errors. In addition, the IMP to IMP transmission control algorithm is improved if it detects duplicates that are sometimes generated by retransmission. An algorithm which performs all four of these tasks is described below.

A number of logical "channels" are maintained between each pair of IMPs. Consider but one channel to begin, and further consider packet transmissions in only one direction on this channel. Of course, acknowledgements go the other direction on the channel. At both the transmit and receive end of this channel a one bit sequence number is kept. We call this bit an odd/even bit. Both transmit and receive odd/even bits are initialized to be zero. Also, at the

transmit end, a used/unused bit is kept for the channel. It is of course initialized to zero, meaning unused. When it is time to transmit a packet, a check is first made for the channel being unused. If it was previously unused, it is marked as used and the packet is transmitted. The state of the transmit odd/even bit is included with the packet. When the packet arrives at the receiver, assuming the packet is received correctly, the packet's odd/even bit is checked against the receive odd/even bit. If they match, the packet is accepted and the receive odd/even bit is complemented. Otherwise, the packet would be ignored. In any case the receive odd/even bit is returned as an acknowledgement. At the transmitter, if the acknowledgement bit does not match the transmit odd/even bit, the packet has been successfully sent and acknowledged and the packet can be discarded, the channel marked unused, and the transmit odd/even bit complemented. Otherwise the acknowledgement is a duplicate and is ignored. Suppose now a second copy of the packet arrives at the receiver, a packet which was sent before the first acknowledgement had a chance to get back to the transmitter. When this packet arrives at the receiver, its odd/even bit does not match the receive odd/even bit and so that packet is discarded as a duplicate. Nonetheless, an acknowledgement is sent for the packet using the present state of the receive odd/even bit. When the acknowledgement gets to the transmitter, it does match the transmit odd/even bit, so the acknowledgement is a duplicate and is ignored.

The only subtlety in this algorithm is that the acknowledgement bit is the state of the receive odd/even bit after it is perhaps complemented rather than before it is perhaps complemented. Hence the need for the "not match" rule when the acknowledgement arrives at the transmitter. A closely related algorithm was reported in (Bartlett 69).

Because of the potentially long distances between IMPs, one channel is not enough to keep the inter-IMP lines fully loaded. Therefore, eight logical channels are supplied between each pair of IMPs (32 are supplied between Satellite IMPs). It is not necessary to maintain ordering of IMP to IMP transmissions since packet ordering is performed at the destination IMP. This means that the transmit channels can be filled in any convenient order, and at the receive side, packets can be forwarded onwards as soon as they are correctly received regardless of the channel over which they arrived.

To avoid requiring separate packets for acknowledgements, acknowledgement bits are "piggy-backed" in packets going the other way on the line. In fact, all eight receive odd/even bits are transmitted with every packet going the other way. In the absence of any traffic going the other way on the line, a packet carrying only the eight

acknowledgements is sent. In either case, the acknowledgements get back to the transmitter as fast as possible. Therefore, the transmitter knows very soon whether a packet requires retransmission or not, allowing the use of a minimal timeout before retransmission. "Piggy-backing" all acknowledgements into every packet going the other way saves program bandwidth, line bandwidth, and buffer space over a system which sends individual acknowledgements in individual packets.

In view of the use of a number of channels and the delay encountered on long lines, some packets might have to wait an inordinately long time for transmission. Traffic that is essentially interactive should not be subjected to waiting for several thousand-bit packets to be transmitted, multiplying by ten or more the effective delay seen by the source. Therefore, the IMPs maintain two sets of queues for each output line, and service all priority transmissions before any regular transmissions. Preemptive priority is not employed.

2.7 Details of the Inter-IMP Packet Format

The following figure shows the format of packets as they appear on the inter-IMP circuits.

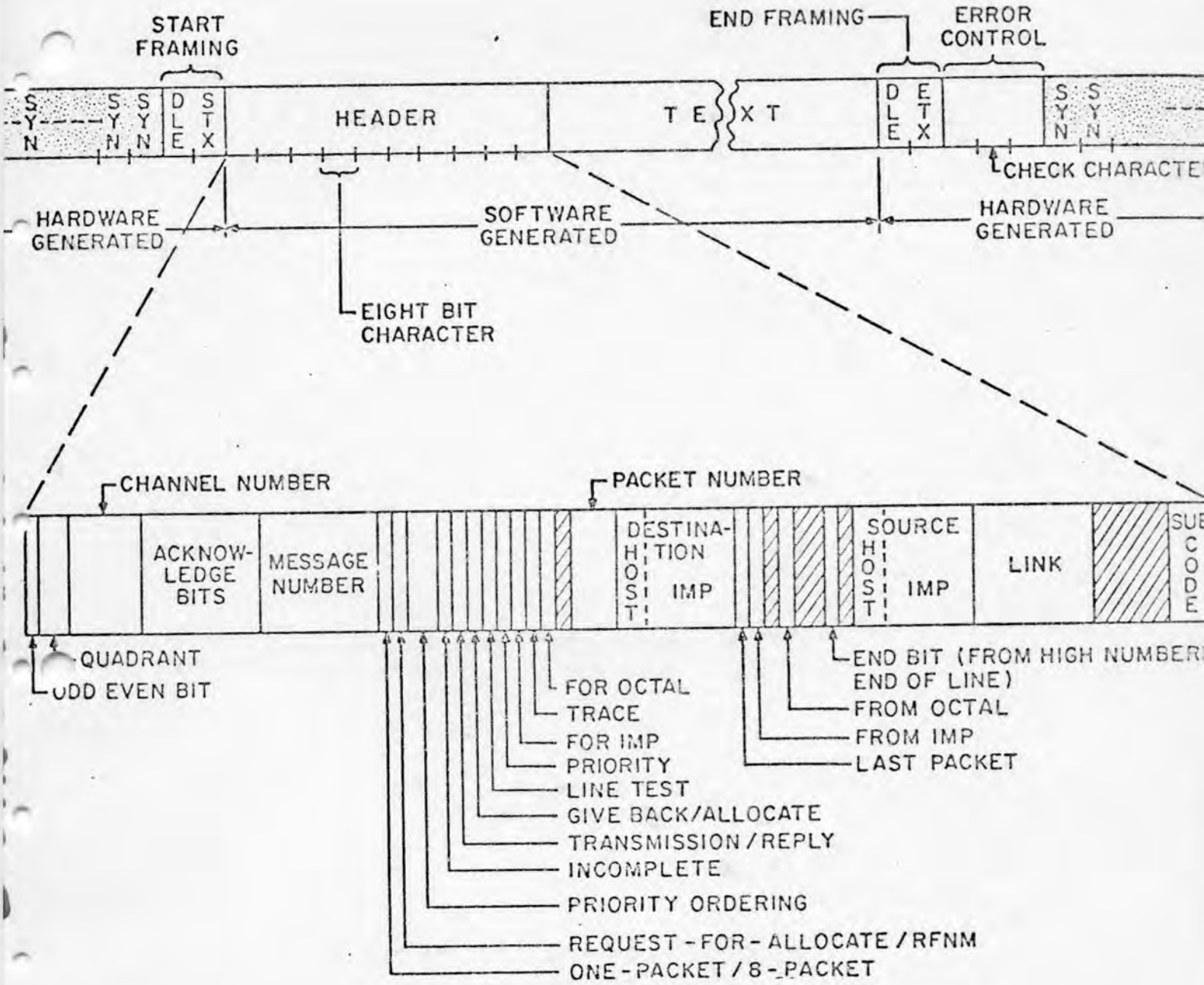

Figure 1 Inter-IMP Packet Format

Packets are permitted to be varilable length up to a maximum of about 1000 bits. The IMP's modem interface transmission hardware adds framing characters to each packet as it is transmitted onto an inter-IMP circuit. At the front of the packet two characters (DLE and STX) are added indicating the beginning of the packet. At the end of the packet two characters (DLE and ETX) are added indicating the end of the packet. The checksum is appended after the end of packet characters. The IMP's modem interface reception hardware has the capability of detecting the start and the end of a packet from these framing characters thus freeing

the program from the burden of detecting packets boundaries. Between packets the transmission hardware automatically generates synchronizing characters (SYN) which the reception hardware automatically discards.

There is no restriction on the content of a packet. Arbitrary sequences of bits may be transmitted without restriction. This transparency is achieved by a method known as DLE-doubling. If the data in the packet itself contains a DLE-character (the character which introduces the packet start and packet end sequences), that DLE-character in the data is doubled by the transmission hardware. At the receiver, the hardware collapses double DLEs in the data back into one. So there is complete transparency on the inter-IMP channels. This is a very important point. All too many networks require transmission to be limited to characters from a particular character set. Besides preventing the network's users from sending arbitrary message, the designers of such networks are themselves prevented from such things as loading programs over the network.

In addition to showing the part of the packet format done with hardware, the above figure also shows the channel field, "piggy-back" acknowledgement field, etc. mentioned in the preceeding section. The portions of the packet which carry the end-to-end control information (e.g., destination address, message sequence numbers, etc.) is also shown.

Notice that much of the inter-IMP communication algorithm is performed with hardware. Software is particularly bad at generating powerful checksums, scaning for packet boundaries, and so forth, especially when it must be done on a character-by-character or bit-by-bit basis. Therefore, this is done with hardware in the ARPA Network.

2.8 Inter-IMP Buffering and Allocation

Consider the following figure.

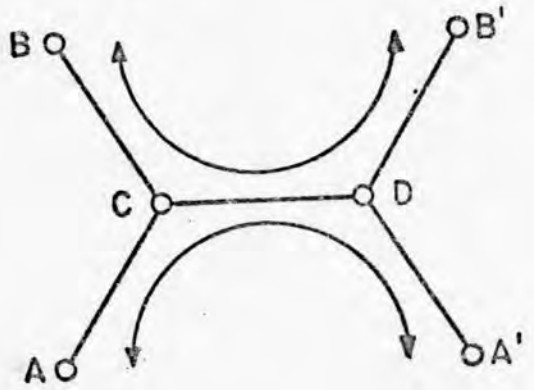

Figure 2 Store-and-Forward Lockup

Two-way single-packet traffic flows between A and A' and also between B and B' and is constrained by network topology to use the circuit between IMPs C and D. Suppose all the buffer storage in IMP C can become filled with packets on the output queue to IMP D, and all the buffer storage in IMP D can become filled with packets on the output queue to IMP C. In this case, both IMP C and IMP D would be engaged in a direct confrontation in which both IMPs must lose all incoming packets (and acknowledgements) as neither machine has any buffer space with which to receive inputs. We call this a store-and-forward lockup.

It is straightforward to prevent such store-and-forward lockups, and this is done in the ARPA Network implementation. The key technique used is to guarantee that sufficient buffers are reserved so that it is always possible to input and to output one packet over each circuit. Thus, packets (and acknowledgements) can always be passed between neighboring machines (albeit at a trickle perhaps). In particular, in the IMP system, one buffer is always allocated for output on each line, guaranteeing that output is always possible; and double buffering is provided for input on each line, which permits all input traffic to be examined by the program, so that acknowledgements can always be processed, which frees buffers. Additionally, an attempt is made to provide enough store-and-forward buffers

so that all lines may operate at full capacity.

We conclude this section with two interesting notes: 1) negative acknowledgements could be useful in activating dormant buffers more quickly, but add complexity and they are not used in the ARPA Network; 2) more complex forms of store-and-forward lockup than that given in the example above are possible, and while most are protected against in the ARPA Network implementation, at least one rare case is not protected against. See (Kahn 71) for further discussion of this point.