Networks, Routers and Transputers:

Function, Performance and applications

Edited by: M.D. May, P.W. Thompson, and P.H. Welch

This chapter was written by C. Barnaby, V.A. Griffiths and P.W. Thompson.

6 Models of DS–Link Performance

This chapter contains analytic studies of the performance of DS-Links, the IMS T9000 virtual channel processor and the IMS C104 packet routing switch.

The first section considers the overheads imposed by the various layers of the DS–Link protocol on the raw bit–rate. Results are presented for the limiting bandwidth as a function of message size, which show that the overheads are very moderate for all but the smallest messages (for which the cost of initiating and receiving a message will dominate in any case).

The next section analyses the diminution of bandwidth caused by latency at both the token flow–control and packet–acknowledge layers of the protocol. The losses due to stalls at the packet level of the protocol when only a single virtual channel is active are plotted in the latter part of the section.

The final section considers the performance of the C104 routing switch under heavy load, both in the average and the worst case.

6.1 Performance of the DS–Link Protocol

This section looks at the maximum throughput of user data on a DS-Link implementing the virtual channel protocol (described in chapter 3) for a given message size. Two values are calculated, for unidirectional and bidirectional link use. These give bounds on the data transfer rate for a given message size. The DS–Link protocol requires use of flow–control tokens, packet headers and termination tokens. The analysis calculates how many bits have to be transmitted along a DS–Link in order to transfer a message, taking all of these overheads into account.

It is useful to define the ceiling function $\lceil x \rceil$:= (least integer greater than or equal to x).

6.1.1 Unidirectional data transfer

Assume that we have a message of size m bytes. This will be transmitted as n_p packets. If the message is sent as a single large packet, $n_p = 1$. If the message is split into packets of a maximum size 32 bytes,

$$n_p = \left\lceil \frac{m}{32} \right\rceil$$

Let s be the header size, in bytes. The number of bits transmitted for the message is

$$b_d = 10m + (10s + 4)n_p$$

since there are 10 bits for every byte of data, and a header-terminator overhead per packet. This overhead is 10 bits for each byte of header, and 4 bits for the terminator.

In the synchronised message–passing protocol used by the IMS T9000, each packet of a message must be acknowledged by an acknowledge packet, which we assume uses the inbound link. Since there will be one acknowledge for every outbound packet of data, the whole message will require n_p inbound acknowledge packets. The inbound acknowledge packets require outbound flow control tokens. There are s data tokens in the header of each acknowledge packet, and one data token in the terminator of each acknowledge packet. The total number of inbound data tokens for the acknowledge packets is

$$n_{dt} = (s + 1)n_p$$

For every eight inbound data tokens, there is an outbound flow control token. The number of flow control tokens is rounded up to the nearest integer for the purposes of the model

$$n_{ft} = \left\lceil \frac{n_{dt}}{8} \right\rceil$$

A flow control token is 4 bits. The total number of outbound bits, B, required to transmit a message is the sum of the data bits and the flow control bits.

$$B = b_d + 4n_{ft}$$

The number of bits in the message transferred is $8m$, and this requires B bits to be transmitted on the 100 MBit/s link. Thus the data rate on the link, D, is given by:

$$D = \frac{8m}{B} \times 100 \ Mbits/s$$

6.1.2 Bidirectional data transfer

The message to be transferred has m bytes of data, and the number of packets required to transfer this data, n_p, is, as in the unidirectional case, given by

$$n_p = \left\lceil \frac{m}{32} \right\rceil$$

The data rate will differ from the unidirectional case because the outbound will have to carry a greater number of flow–control tokens corresponding to the increased amount of data on the inbound link, and also acknowledge packets for the message packets received on the inbound link.

Without loss of generality, the message analyzed is assumed to be on the outbound link. The inbound link is assumed to carry the same amount of data as the outbound link.

The outbound link will carry the data packets for the outbound message, the acknowledge packets for the inbound message, and the flow control tokens for all packets on the inbound link. The number of outbound data packets is n_p. The number of outbound acknowledge packets equals the number of inbound data packets, which in turn equals the number of outbound data packets (since the inbound link is assumed to carry the same amount of data as the outbound link). The number of acknowledge packets is therefore also n_p. Each acknowledge packet is transmitted as $(10\,s + 4)$ bits. The number of bits transmitted for the outbound message and the outbound acknowledgement packets is

$$b_d = (10m + (10s + 4)n_p) + (10s + 4)n_p$$

Now consider the flow control requirements. The outbound link will carry the flow control tokens for the packets received on the inbound link. The data tokens on the inbound link will be the sum of the number of data tokens for the inbound data transfer, and the number of data tokens for the inbound acknowledge packets. Recall that the inbound link carries the same amount of data as the outbound link. The number of data tokens on the inbound link is

$$n_{dt} = (m + (s + 1)n_p) + (s + 1)n_p$$

The number of flow control tokens required on the outbound link is (rounded up for the purposes of the model)

$$n_{ft} = \left\lceil \frac{n_{dt}}{8} \right\rceil$$

The total number of outbound bits for the message transfer is given by the sum

$$B = b_d + 4n_{ft}$$

and the outbound link data bandwidth is, as before,

$$D = \frac{8m}{B} \times 100 \ Mbits/s$$

Note that the bandwidth on the inbound link is the same, by assumption.

6.1.3 Asymptotic Results

Consider first the case where the message is split into packets of maximum size 32 bytes. For large messages, the overhead of the final, possibly not full size, packets will become negligible. In this case, the asymptotic values for throughput may be calculated.

Unidirectional link use

From the previous derivations, assuming that only 32–byte packets are used, we have

$$n_p = \frac{m}{32}$$

$$b_d = (10m + (10s + 4)n_p) = 10m + (10s + 4)\frac{m}{32}$$

$$n_{dt} = (s + 1)n_p = (s + 1)\frac{m}{32}$$

$$n_{ft} = \frac{n_{dt}}{8} = \frac{(s + 1)m}{8 \times 32}$$

$$B = 10m + (10s + 4)\frac{m}{32} + 4\frac{(s + 1)m}{8 \times 32}$$

collecting terms,

$$B = m\left(\frac{649 + 21s}{64}\right)$$

giving D in terms of s,

$$D = \frac{8m}{B} \times 100 = \frac{51200}{649 + 21s} \ Mbits/s$$

Bidirectional Link use

The bandwidth for the bidirectional case is calculated similarly, giving the asymptote

$$D = \frac{25600}{345 + 21s} \ Mbits/s$$

6.1.4 Results

The model is used to calculate data throughput for varying message size. This is the throughput in the outbound direction only, for both unidirectional and bidirectional link usage. This is calculated for both 1-byte and 2-byte header sizes. The graphs show data throughput, in Mbytes per second, for varying message size, header size and link usage. The asymptotic values are calculated below. The graphs also show the bandwidth that would result from sending the entire message as a single packet. This illustrates the relatively small cost of dividing messages into packets, which has considerable advantages in terms of fine–grain multiplexing and small buffer requirements.

Consider figure 6.1. It shows the data throughput for unidirectional and bidirectional link usage, with 1-byte headers, for messages up to 128 bytes. The larger discontinuity in the throughput curve occurs when an extra packet is required to transmit a message, for the maximum packet size of 32 bytes. The small discontinuities are due to the requirement to send an additional flow–control token. This is more pronounced in the bidirectional case. The overhead of the extra packet has less effect on throughput for the larger messages. Note that the knee in the graph occurs for very small messages. Only messages of 10 bytes or less cause appreciable degradation in the throughput rate.

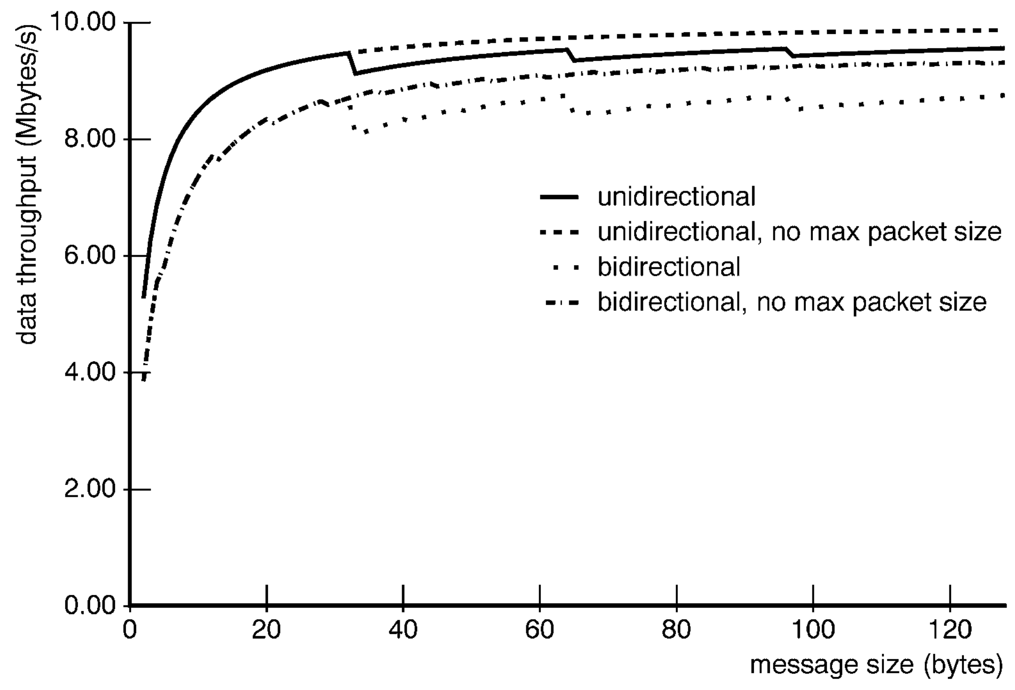

Figure 6.1 Outbound link throughput for small messages (1 byte headers)

The second graph, figure 6.2, shows the throughput for larger messages. Again a 1-byte header is assumed. Throughput is calculated for messages of size $32 \times i$ and for size $(32 \times i) + 1$ for integer values i.

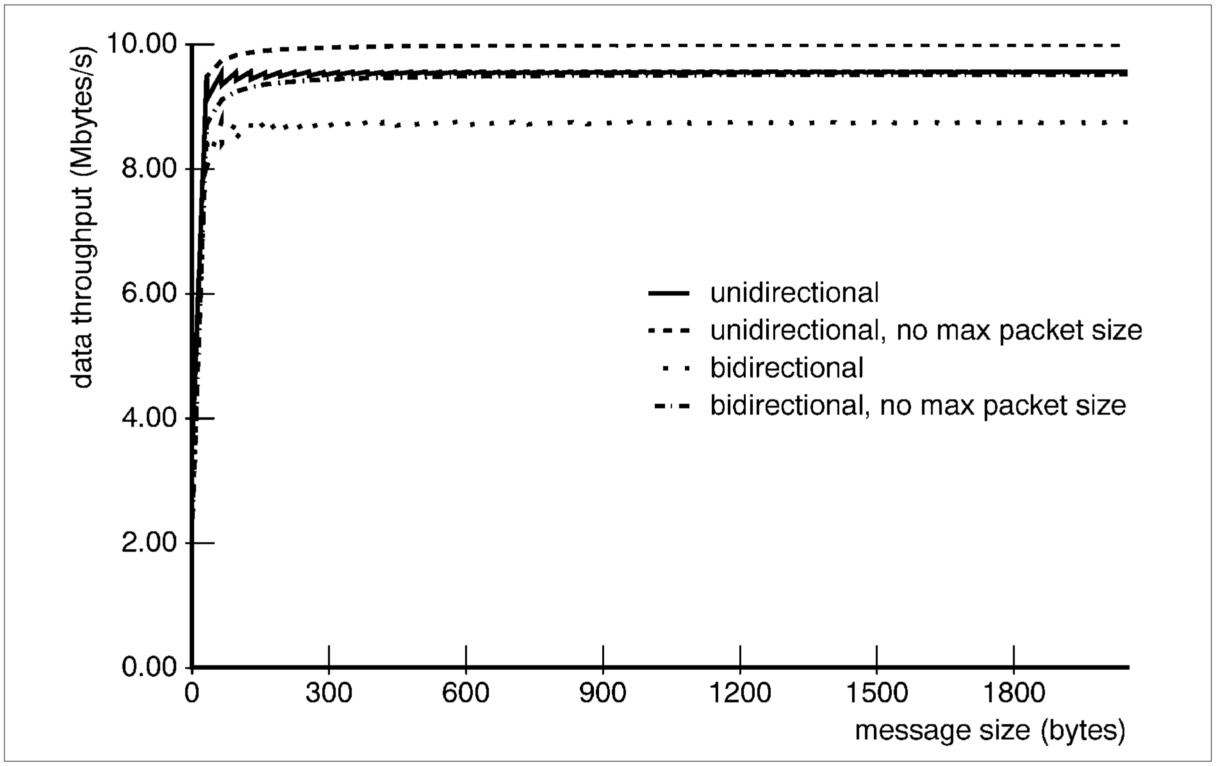

Figure 6.2 Outbound link throughput for large messages (1 byte headers)

The use of a two-byte header increases the overhead of each extra packet needed for the message. In figure 6.3, the data throughput for small messages with 2-byte headers, the overhead shows as a larger ‘‘dip’’ in the curve when an extra packet is used for the 32 byte maximum packet size. Figure 6.4 shows the use of 2-byte headers with larger message sizes.

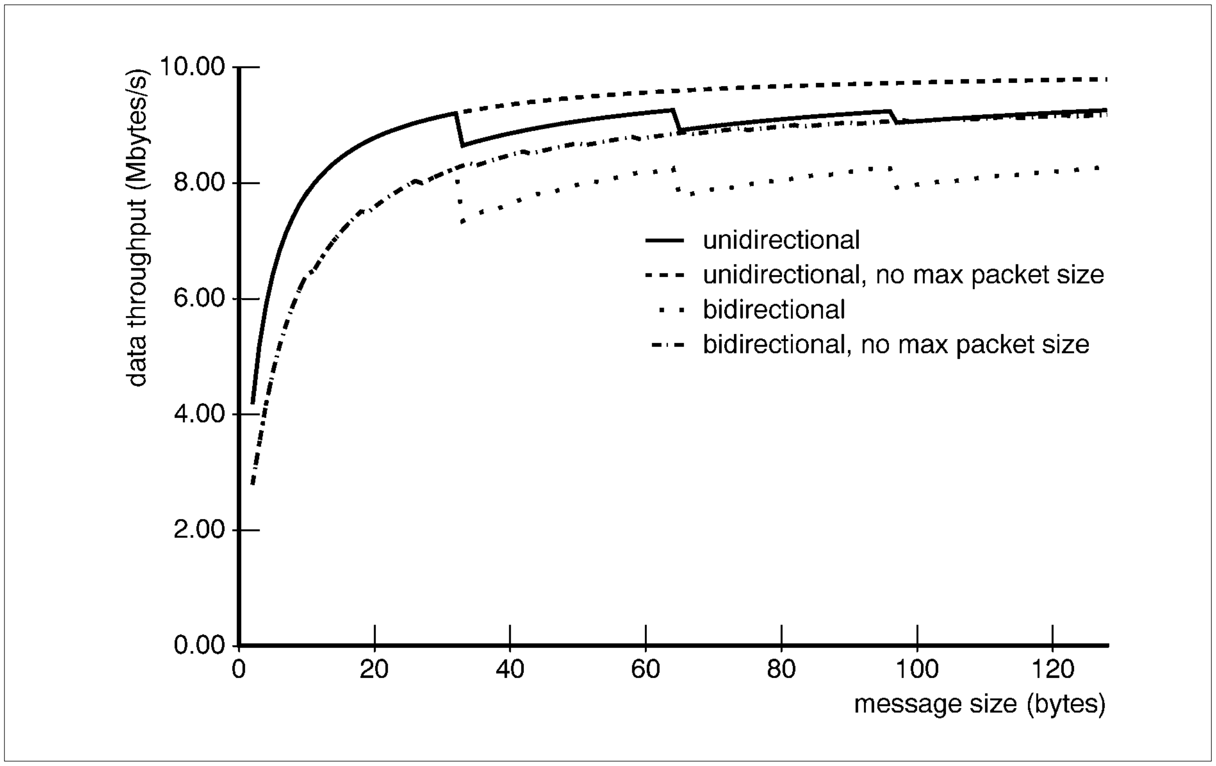

Figure 6.3 Outbound link throughput for small messages (2 byte headers)

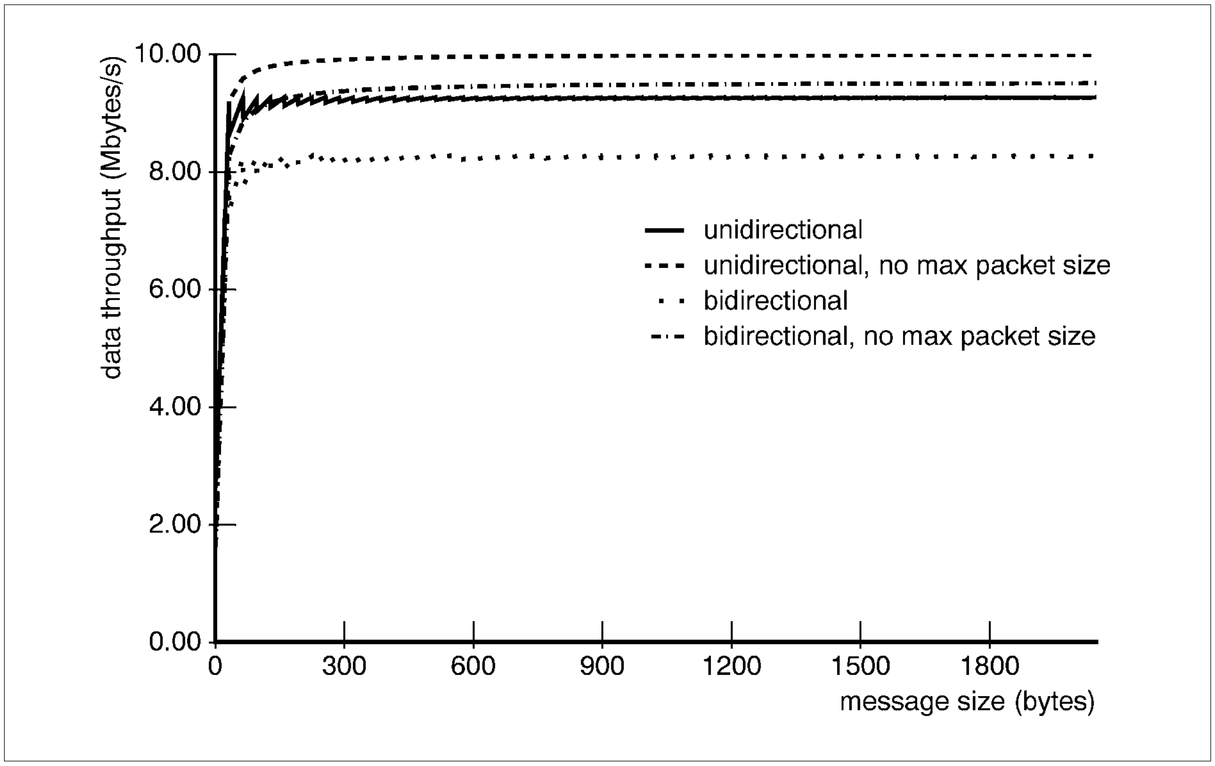

Figure 6.4 Outbound link throughput for large messages (2 byte headers)

6.1.5 Asymptotic Results

The values s=1 and s=2 are substituted into the limiting expressions for D given earlier. The results for the 32 byte maximum packet size are shown in table 6.1. The figures given are in Mbytes per second. Note that these figures are the asymptotes of the graphs.

Table 6.1 Effect of header size and usage on link throughput

s	**Unidirectional**	**Bidirectional**
1	9.55	8.74
2	9.26	8.27

Effect of maximum packet size

If the message is not split into packets then, for unidirectional data transfer the expressions for throughput derived above give

$$D = \frac{8m}{10m + 10s + 8} \times 100 \rightarrow 80\ Mbits/s$$

For bidirectional data transfer, there is a slightly larger overhead due to the flow control information. Again from the previous derivations,

$$D = \frac{8m}{10.5m + 21s + 9} \times 100 \rightarrow 76.19\ Mbits/s$$

6.2 Bandwidth Effects of Latency

In practice the bandwidth achieved at the user level is sometimes less than the theoretical peak calculated in the previous section, because latencies in the system cause the link to become idle for part of the time. In this section we first of all consider the effect of device–to–device latencies on the token-level protocol, and then consider the effect of end–to–end latency on the upper levels of the virtual channel protocol.

6.2.1 Bandwidth of Long Link Connections

Signals propagate through wires with a finite speed, and so long lengths of wire are themselves a source of latency, which can be significant at the speed of DS-Links,. What follows is a formal model of the flow–control mechanism of the DS–Links, which is used to calculate the maximum tolerable device–to–device latency before a link is forced to become idle.

Specification of DS–Link Flow–control

We consider a pair of links connected together. Each link is connected to both a source and a sink of data. Transmission/buffering delay between the links is modelled by a pair of buffers between them. The picture is shown in figure 6.5.

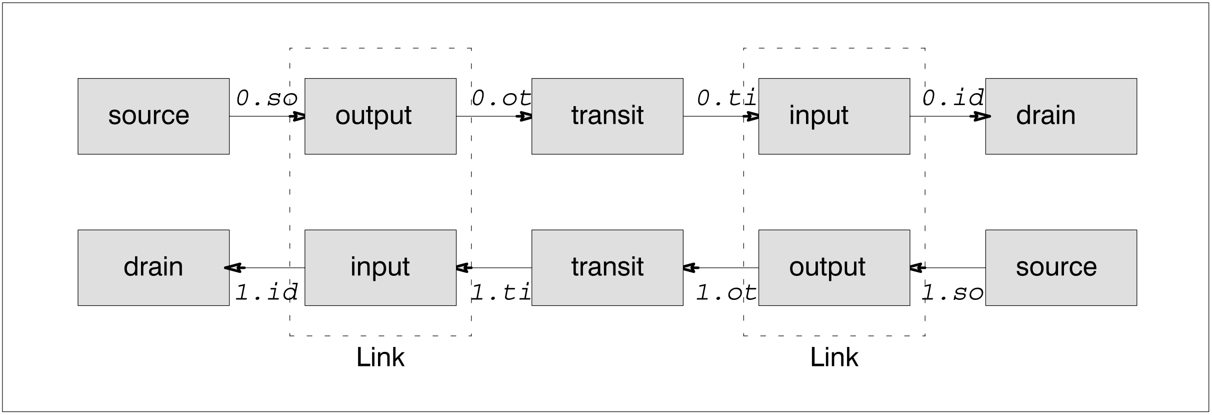

Figure 6.5 Token streams in a bi–directional DS–Link connection

Formally, we regard each labelled channel as a *trace*, i.e. a sequence of tokens transmitted upto the current time.

There are 256 different data tokens, an end-of-packet token (*EOP*), an end-of-message token (*EOM*), the flow-control token (*FCT*) and a null token. The set of tokens is thus $T = D \cup F \cup S$ where $D = \{data, EOP, EOM\}$, $F = \{FCT\}$ and S is the null token. We indicate the restriction of a trace to a sub-alphabet by $\lceil$, and the length of a trace by #. $<>$ is the empty trace. $a \preceq b$ means that the trace a is an initial subsequence of the trace b (so b can be thought of as a 'continuation' of a).

Almost all relations are given in one direction only; an exactly equivalent set hold with '**0**' and '**1**' interchanged.

Firstly, note that the streams from the source and to the drain contain only data, EOPs and EOMs; there are no flow–control or null tokens other than between the two link interfaces, so the restriction of the other traces to the set $F \cup S$ is empty:

$$\mathbf{0}.so\lceil(\,\boldsymbol{F} \cup \boldsymbol{S}\,) \;=\; \mathbf{0}.id\lceil\,(\,\boldsymbol{F} \cup \boldsymbol{S}) \;=\; \mathbf{1}.so\lceil(\boldsymbol{F} \cup \boldsymbol{S}) \;=\; \mathbf{1}.id\lceil(\boldsymbol{F} \cup \boldsymbol{S}) \;=\; <>.$$

The sequence of tokens is preserved, so the trace of data tokens received by the drain is a strict initial subsequence of the trace of data tokens sent by the source (the difference being those still in transit):

$$\mathbf{0}.id \preccurlyeq \mathbf{0}.ti \lceil \boldsymbol{D} \preccurlyeq \mathbf{0}.ot \lceil \boldsymbol{D} \preccurlyeq \mathbf{0}.so$$

$$\mathbf{0}.ti = \mathbf{0}.ot$$

The number of tokens held in each box is the difference between the number of tokens input and the number output. All the boxes (except the sources and drains) have finite capacities, thus:

$$\begin{array}{lll} 0 & \leq \#\mathbf{0}.so - \#(\mathbf{0}.ot \lceil D) & \leq output.cap.\mathbf{0} \\ 0 & \leq \#\mathbf{0}.ot - \#\mathbf{0}.ti & \leq t.delay \\ 0 & \leq \#(\mathbf{0}.ti \lceil D) - \#\mathbf{0}.id & \leq input.cap.\mathbf{0} \end{array}$$

The total credit received is the number of *FCT*s received times the flow-control batch–size *bsize*. The output credit remaining for that link is the difference between this and the number of data tokens sent. The total credit sent is the number of *FCT*s sent times the flow-control batch–size; the input credit remaining is the difference between this and the number of data tokens received[18]. Since we have a ‘credit’ based system all of them must be positive, and from the sequence relations above we can deduce:

$$input.credit.\mathbf{0} = \#(1.ot \lceil F) \times bsize - \#(\mathbf{0}.ti \lceil D) \geq output.credit.\mathbf{0}$$
$$output.credit.\mathbf{0} = \#(1.ti \lceil F) \times bsize - \#(\mathbf{0}.ot \lceil D) \geq 0$$

The input credit must never exceed the buffer space available, which is the difference between the size of the buffer and the number of tokens held. Thus we require:

$$input.cap.\mathbf{0} - \#(\mathbf{0}.\mathrm{ti} \lceil D) + \#\mathbf{0}.id \geq input.credit.\mathbf{0}$$

Combining the above gives the simple relation:

$$input.cap.\mathbf{0} + \#\mathbf{0}.id \geq \#(\mathbf{1}.ot \lceil F) \times bsize$$

This shows that the total credit issued must not exceed the buffer capacity of the input plus the amount sent to the drain.

Initially all the traces have zero length. The condition for actual transfer of data is that at least one of the traces into the drains (e.g. $\mathbf{0}.id$) must become non-empty. By the above this implies that $\#(\mathbf{0}.ot \lceil D)$ becomes non-zero. Now for this to happen $\#(\mathbf{1}.ti \lceil F) \times bsize$ must become non-zero, this is bounded from above by $input.cap.0$, since at the start $\#\mathbf{0}.id = 0$. Since the length of traces is integral, this shows that no data can ever be transferred unless $input.cap.0 \geq bsize$. Thus this form of flow−control requires an input buffer at least as large as the flow−control batch−size.

6.2.2 Effect of Inter–Link Delay

Now consider the consequences of the transit delay. If the delay is constant, it behaves as a fixed-size buffer which can only output when it is *full*. This means that in the steady-state there is always a fixed difference in the lengths of its input and output streams, i.e.

$$\#\mathbf{0}.ot = \#\mathbf{0}.ti + t.delay.0$$

Thus, for any set A:

$$\#(\mathbf{0}.ot \lceil A) \leq \#(\mathbf{0}.ti \lceil A) + t.delay.0$$

Equality only occurs if *all* the tokens held in transit belong to A. If we assume (for the moment) that data flows only on the **0** channels and transmission is continuous (i.e. no tokens from S are interspersed) this means:

18. The difference between the input and output credits is due to tokens in transit.

$$\#(\mathbf{0}.ot\lceil D) \;=\; \#(\mathbf{0}.ti\lceil D) + t.delay.0$$

On the **1** channels, to maintain the steady state we must send an *FCT* for every *bsize* tokens transmitted on the **0** channels (since these are, by assumption, all tokens from *D*). Thus, on average,

$$\#(\mathbf{1}.ot\lceil F) bsize \;=\; \#(\mathbf{1}.ti\lceil F) \times bsize + t.delay.1$$

Thus,

$$\begin{aligned} input.credit.0 &= \#(\mathbf{1}.ot\lceil F) \times bsize - \#(\mathbf{0}.ti\lceil D) \\ &= \#(\mathbf{1}.ti\lceil F) \times bsize + t.delay.1 - \#(\mathbf{0}.ot\lceil D) + t.delay.0 \end{aligned}$$

Thus:

$$input.cap.0 \;\geq\; input.credit.0 + \#(\mathbf{0}.ti\lceil D) \times bsize + \#\mathbf{0}.id$$

$$= (t.delay.0 + t.delay.1) + (\#(\mathbf{1}.ti\lceil F) \times bsize - \#(\mathbf{0}.ot\lceil D) + (\#(\mathbf{0}.ti\lceil D) + \#\mathbf{0}.id)$$

By the above, the third term of the last equation is ≥ 0.

The second term is *output.credit*.0, which can attain $\left\lfloor \frac{in.cap.0}{bsize} \right\rfloor \times bsize$ when $\#(\mathbf{0}.ot\lceil D)=0$ and $\#(\mathbf{1}.ti\lceil F)$ is equal to the maximum number of flow-control tokens that can be sent.

From the starting condition we can write $input.cap.0 = bsize + extra$ (where *extra* is non-negative) so that the previous expression is simply *extra*.

Thus we deduce:

$$extra \;\geq\; t.delay.0 + t.delay.1$$

So we see that the input buffer must have a minimum capacity of *bsize* for transmission to start, but to maintain continuous transmission of data through a delay, there must also be some some 'slack' to deal with tokens in transit, which depends on the latency of the connection.

If we assume that data flows on both sets of channels, by a similar argument we obtain the result:

$$extra \;\geq\; (t.delay.0 + t.delay.1) \times \left(\frac{bsize}{1 + bsize} \right)$$

Inverting this equation gives a restriction on the maximum delay through which full bandwidth can be sustained:

$$t.delay.0 + t.delay.1 \;\leq\; extra \times \left(\frac{1 + bsize}{bsize} \right)$$

Thus for the DS-Link, for which *bsize* = 8, a total buffer capacity of 20 tokens means that *extra*=12, so the maximum delay which can be endured without loss of bandwidth is:

$$max.delay \;=\; 12 \times \left(\frac{9}{8} \right) \;=\; 13.5$$

where the unit of time is the average time to transmit one token. This depends on the ratio of data tokens to *EOP/M* tokens, the worst case being 1:1. This makes the average token length 7 bits, so at 100 MBits/s the average token transmission time is 70ns. Thus the maximum delay is 945ns. In practice there is some latency associated with the front-end circuitry of the DS-Link, buffers etc.. This could be added explicitly to the model by introducing more buffer processes, but the effect will simply be to reduce the latency budget for the wires. Latencies in the link account for approximately 400ns, so a conservative estimate would allow 500ns for the round–trip wire delay, which corresponds to a distance of about 80m. Allowing for delays in buffers leads to the conclusion that 50m would be a suitably conservative figure.

6.2.3 Bandwidth of a Single Virtual Channel

The T9000 VCP is pipelined in order to sustain the high rate of packet processing required by the virtual channel protocol, just as the DS-Link is pipelined internally to achieve a high bandwidth. When many virtual channels are active simultaneously on a link, the VCP ensures that the link is never idle so that its full bandwidth is exploited. The disadvantage of pipelining is that it introduces latency, which can become a limiting factor on bandwidth when only one virtual channel is active on a link.

The reason that latency can limit bandwidth is because of the requirement that each data packet must be acknowledged before the next may be sent. Although the VCP sends the acknowledge packet as soon as possible, so that its transmission can overlap that of the bulk of the data packet, if the data packet is short and/or the latency of the system is large, it is possible for the acknowledge to arrive too late to prevent a stall in data transmission. When the VCP has packets to send for other channels this does not matter, but if only a single channel is active, the bandwidth may be reduced by the system latency.

Analysis

We consider the particular case of two processes communicating over a single virtual channel. This can be represented in occam by:

```
CHAN OF [message.size]BYTE channel :
PLACED PAR

  [message.size]BYTE message :
  SEQ
    channel ! message
    .
    .  (repeat n times)
    .
    channel ! message

  [message.size]BYTE message :
  SEQ
    channel ? message
    .
    .  (repeat n times)
    .
    channel ? message
```

We ask: what is the bandwidth that this pair of processes observes, as a function of the message size? The bandwidth is defined as the total amount of message data transferred divided by the total time taken (as measured by the outputting process). Whenever the time is limited by the latency of the system, the bandwidth is proportional to the length of the message, since the time is constant. When the time to transmit the data exceeds the round-trip latency it is this which limits the bandwidth. In this case the time to transmit the header is significant, and so there is a difference in bandwidth depending on the size of header used.

We assume that all data is in the cache, the inputting process is initially ready, and that only one process is using each machine. Communication is thus mainly unidirectional, and so we ignore the effects of token level flow-control, since this has been analyzed in the previous section. We assume that the two T9000s are directly connected by short wires.

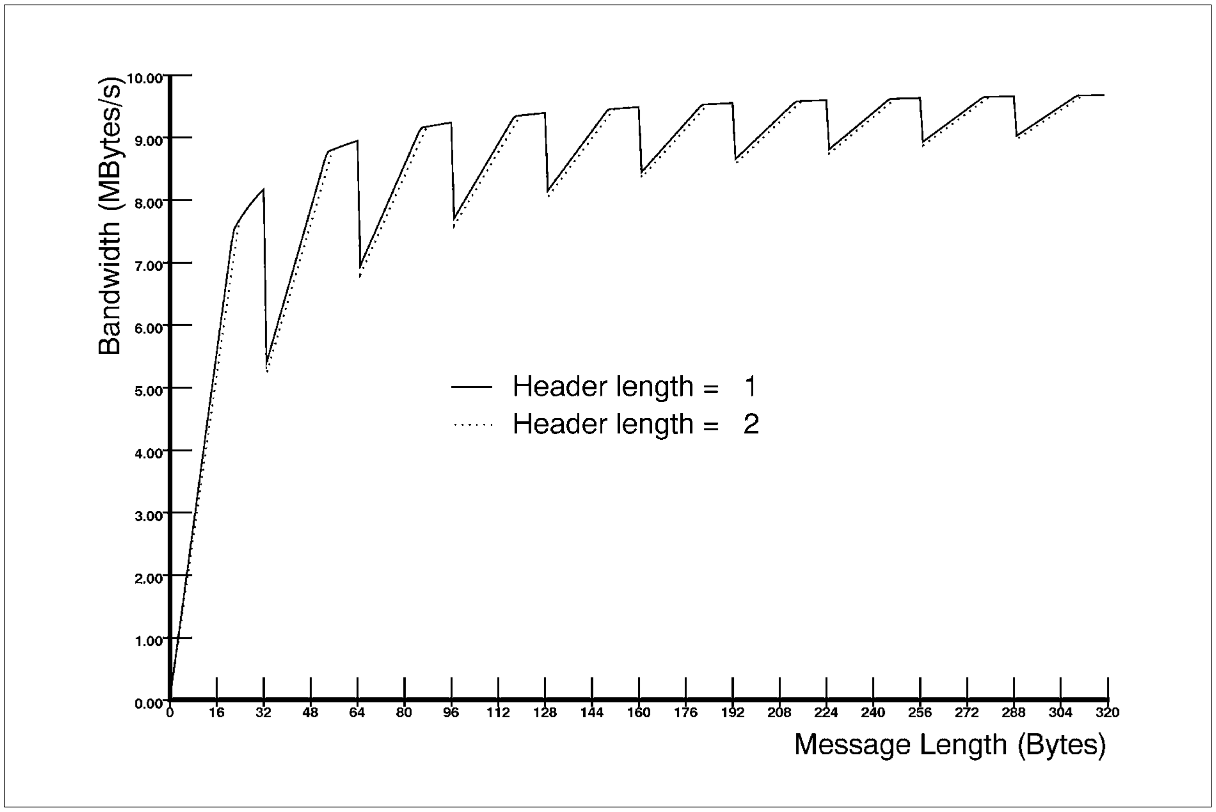

Figure 6.6 Single channel bandwidth vs. message size

The results are illustrated in figure 6.6. It can be seen that for messages of less than about 20 bytes, latency dominates. Packet transmission time becomes limiting until the message size exceeds 32 bytes, when a second packet is required. The time to acknowledge the second packet then becomes limiting until the message size reaches about 50 bytes. This pattern is repeated at each multiple of 32 bytes, but as the message size increases the effect of latency on the final packet becomes proportionately less significant, and so the dips in the graph become smaller. The envelope of the curve is that derived in section 6.1. If the length of the individual connections is greater, and/or the message is routed via one or more C104 routers, the latency is larger, and so the dips in the curve become both wider and deeper. Latency can be hidden by having more channels active at once, since in this case an acknowledge does not have to be received until a packet has been sent for *each* active channel.

6.3 A model of Contention in a Single C104

In this section we develop a statistical model of contention for the C104. The model allows a number of C104 input links to be trying to route packets to a number of C104 output links. The C104 will allow one packet which requests an output link to succeed, for each output link[19]. Each packet has a header, which is one or two bytes, and a terminator. A byte of information is transmitted as 10 bits on the link, and the packet terminator is transmitted as 4 bits.

The model allocates time in slots. Conceptually, at the start of the time slot all of the input links attempt to route a packet to their selected destination link. A subset of these transmissions will succeed, and the other packets are discarded – note that this is *not* what occurs in the implementation. The model developed here assumes that the destination links are chosen at random, and this assumption is appropriate for the the actual behavior of stalling and buffering of unsuccessful

19. Grouped adaptive routing is not considered in this model.

packets for the next time slot, because the destinations of all the packets in the next slot will again be independent and random. The model describes the probability that a packet is successfully transmitted from its chosen outbound link.

6.3.1 Time slots

The size of a time slot is the time for which a packet occupies the input/output pair of links, given that it is successfully routed. This is the sum of the header routing time and the time taken for the following bits to cross the switch.

Let h denote the header delay, and b denote the bit delay. Then the slot time, S, for k-bit packets is $S = h + k' b$, where k' is the number of bits transferred after the header. For one-byte headers, this is $k + 4$ since the bits transferred are the data bits and the 4 terminator bits. Two-byte headers may be modelled by setting $k' = k + 10 + 4$, where the extra 10 bits correspond to the extra byte of header to be transferred.

6.3.2 The Contention Model

At the start of a time slot, each input link submits a packet. If there is contention for the output links, then one packet for each output link is successful.

Let the number of input links in use be in, and the number of output links in use be out, with the ratio $r = out/in$. Then the probability that an output link is not used by any of the inputs is given by

$$p(free) = \left(1 - \frac{1}{out}\right)^{in}$$

The probability that an output link is used is therefore

$$p(use) = 1 - p(free) = 1 - \left(1 - \frac{1}{out}\right)^{in}$$

The data throughput, T, of an input link is given in bits/s by

$$T = \frac{1}{slot\ time} \times p(use) \times rk$$

Substituting in the above expressions gives

$$T = \frac{1 - (1 - \frac{1}{out})^{in}}{S} \times rk$$

6.3.3 Average delay

The model so far describes the expected throughput of each input link. This may be used to calculate the expected number of time slots it will take for a submitted packet to cross the switch. This time is the delay due to the contention within the switch. In order to look at system delays, the time taken to reach the front of the input queue also needs to be taken into account.

The expected delay of a packet, in terms of time slots, is given by

$$L = \sum_{i=1}^{\infty} i \times p(i)$$

where i is the time slot number. Now $p(i)$ is the probability of an input success on the i^{th} trial, so that $p(i) = p(failure)^{i-1} \times p(success)$. Substituting into the expression for L and taking the common factor outside of the summation we get:

$$L = \frac{p(success)}{r} \times \sum_{i=1}^{\infty} i \times p(failure)^{i-1}$$

summing the series, this gives

$$L = \frac{1}{r} \times \frac{1}{p(use)}$$

The absolute expected delay, D, is the value L multiplied by the length of the time slot, $D = L \times (slot\ time)$. Substituting,

$$D = \frac{1}{r} \times \frac{S}{1 - (1 - \frac{1}{out})^{in}}$$

6.3.4 Summary of model

The throughput per terminal link, in units data bits per second is

$$T = \frac{1 - (1 - \frac{1}{out})^{in}}{S} \times rk$$

The total bandwidth of the system is $B = in \times T.$

The expected delay of a packet, in seconds, is

$$D = \frac{1}{r} \times \frac{S}{1 - (1 - \frac{1}{out})^{in}}$$

where the slot–time $S = h + k'b$, for k bit packets where k' is the number of bits transferred after the header, and

- h is the time taken to route a header, in seconds
- b is the time taken to transmit a bit, in seconds
- k is the number of bits in the packet
- in is the number of input links used, $in \leq 32$
- out is the number of output links used, $out \leq 32$
- r is the ratio $r = out/in$

6.3.5 Asymptotes of the model

The expression $\left(1 - \frac{x}{n}\right)^n$ tends to the value e^{-x} as $n \to \infty$. As both in and out grow, the asymptotic throughput and delay for a particular set of switch parameters may be calculated. The limit of the expressions is

$$\left(1 - \frac{1}{out}\right)^{in} = \left(1 - \frac{1}{r \times in}\right)^{in} \rightarrow e^{-1/r}$$

$$T_{lim} = \frac{1 - e^{-1/r}}{S} \times rk$$

$$D_{lim} = \frac{1}{r} \times \frac{S}{1 - e^{-1/r}}$$

In the special case where $r = 1$, i.e. the numbers of input and output links in use are the same,

$$T_{lim} = \frac{0.632 \times k}{S}$$

$$D_{lim} = 1.582S$$

6.3.6 Results

The IMS C104 chip has a header routing time, h of approximately 500ns, and a bit delay, b of 10ns (assuming links operating at 100Mbits/s). The throughput of each input link is calculated as a function of the number of links in use, and of the packet size (in data bits).

The model is first used to describe the throughput of the chip when the number of input links in use is the same as the number of output links in use. In the equations, this is setting *in=out*, or $r = 1$. Figures 6.7 and 6.8 show the resulting throughput and delay respectively for one-byte packet headers. All of the message is sent in a single packet, with one header and one terminator. The value of S is $S = h + (1.25k + 4) \times b$. Note that the curves for both the throughput and delay quickly flatten.

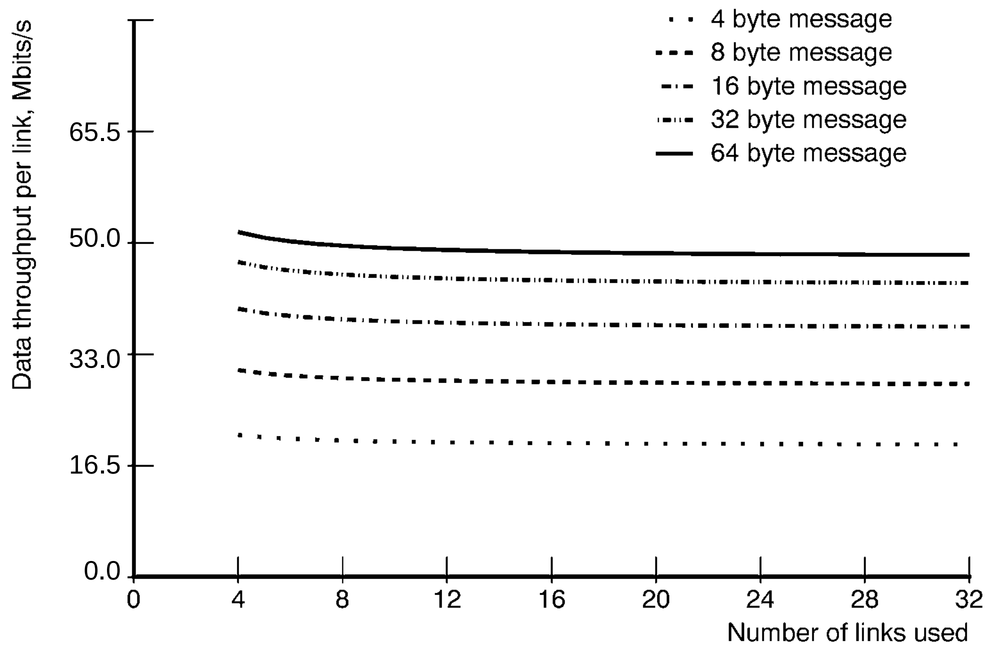

Figure 6.7 Throughput per link vs. packet size