A Communications Model for a PUMA Machine

Tim Oliver
Centre for Mathematical Software Research
University of Liverpool

October 1990

Abstract

This document presents the problem of selecting suitable cost models for channel communication on PUMA machines. We consider various implementations of point to point communication and develop detailed cost models for these. Simple cost models, suitable for use in modelling the cost of numerical algorithms, are proposed and compared with the more complex models.

1 Introduction

A cost model of a numerical algorithm for a PUMA machine must take account of the inter-processor communications. The communications model used should be as simple as possible to allow quick development of models whilst still providing sufficient accuracy. For a T8 network the cost of channel communication can be modelled well by just a single parameter giving the time taken to transfer a `REAL32` across a link [8, 5]. However the complexity of H1/C104 channel communications requires a more complicated model. (Information about the architecture of the H1 and C104 is drawn from IMMOS documents [6, 7, 3].)

An algorithm and accompanying cost model for T8 networks was specific to a particular configuration of Transputers, for example a chain, ring or grid topology. However, the physical configuration of C104 switch chips on a PUMA machine does not restrict the logical configuration of processors required by an algorithm since the architecture provides 'virtual links' between any pair of processes. As an algorithm is now independent of the switch network configuration, we would like the cost model for the algorithm also to be independent of the configuration. Unfortunately, the cost of channel communication depends on the number of switch chips that messages traverse and hence will depend on the machine configuration. In general the configuration of the switch network will be unknown; machines with the same number, p, of H1 processors may have different numbers of C104 switch chips connected in different topologies balancing the requirements of network performance and financial cost.

In the following sections we examine the H1/C104 architecture in more detail developing cost models that include some of the architectural features explicitly. Then we see how we can develop simple cost models of communication for use in modelling numerical algorithms.

2 A Plethora of Parameters

Let us consider the following scenario: a source process on a processor wishes to send a multi-packet message to a destination process on another processor connected to the switch network. There are s switch chips on the path that the message will travel from source to destination processor. We assume that the bandwidth of the network is great enough to allow us to neglect the possibility of collisions between different messages. For simplicity we also assume that the destination process is ready to receive the message when the first packet arrives. This avoids the need to consider idle time in source and destination processor. However, the model for a complete numerical algorithm should take account of the idle time due to synchronisations between processes. The following parameters may be useful in modelling the cost of this scenario:

n *number of bytes in message*
: Let us assume that n is a multiple of b (see below).

h *number of bytes in header*
: We will use a value $h = 3$ giving 2^{24} distinct channels which should be ample even for large networks and algorithms which use many channels.

b *maximum number of bytes in a packet*
: Value: $b = 32$.

s *number of C104 switch chips a packet passes through*

α *transmission time*
: The time taken to transmit a single byte on a link. We will use a value of $\alpha = 100$ns (i.e.10 MBytes/s link speed).

β *packet initialise time*
: The time taken for the Virtual Channel Processor (VCP) to initialise the output of a packet on a link. Value: $\beta = 200$ns.

γ *channel initialise time*
: The time taken by the integer unit to set up a Virtual Link Control Block (VLCB) for output of a message on a virtual link. Value: $\gamma = 500$ns.

δ *switch delay time*
: The delay introduced by a C104 switch chip as a packet passes through it. Value: $\delta = 1\mu$s.

(Values for these parameters are estimates only.)

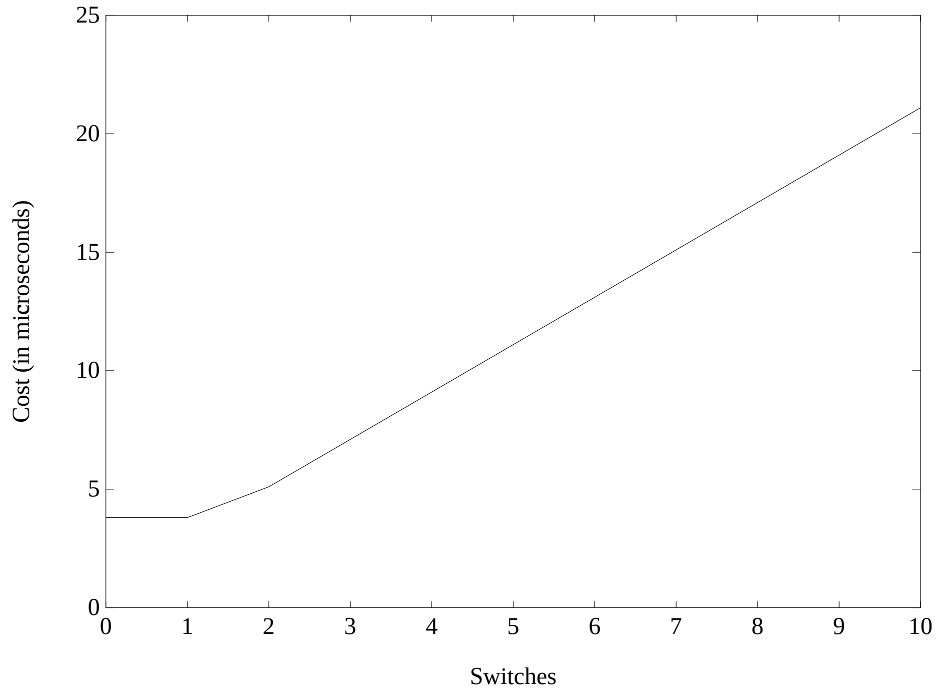
3 Packet Transmission

For the transmission of a single full packet between processors there are two different expressions for the cost (as measured by the source processor) depending on the number, s, of switch chips that the packet traverses. If s is small enough so that the source receives an acknowledge packet (ACK) before it has transmitted all of the bytes in the packet then the cost is simply the time taken for the source processor's VCP to be initialised and to transmit the header, data bytes and EOP token i.e., $\beta + (h + b + 1)\alpha$. (For simplicity, we use the same cost for communicating a token as for a single byte). However, as s increases the time taken to receive the acknowledge is increased by the delay introduced by the intervening switch chips. For sufficiently large s the time to receive the ACK is larger than the cost to output the packet and so the cost is determined by the time taken to receive the ACK. The time taken for the VCP on the source processor to initialise and output the header is $\beta + h\alpha$. After a delay of $s\delta$ the header has been received by the destination VCP. The VCP processes the header to generate an ACK in time β and this is transmitted back at cost $(h + 1)\alpha$. The ACK is received by the source VCP after a further delay $s\delta$. The total cost in this case is $2\beta + (2h + 1)\alpha + 2s\delta$. So the cost of transmitting a full packet is given by:

$$T_p = \max\begin{cases} \beta + (h + b + 1)\alpha \\ 2\beta + (2h + 1)\alpha + 2s\delta \end{cases} \tag{1}$$

Graph 1 shows the cost for transmitting a full packet across varying numbers of switch chips using estimates for the parameters as given in Section 2. The point at which the cost of transmitting a packet begins to depend on s, the number of switch chips traversed, is derived from Equation 1:

$$s \geq \frac{(b - h)\alpha - \beta}{2\delta} \tag{2}$$



Graph 1: Cost of Transmitting a Single Packet

4 Message Transmission

In this section we are concerned mainly with the communication of long messages ($n > 100$bytes). This is common in numerical library codes, which frequently distribute large matrices and vectors. To simplify the models we assume that n is also a multiple of b. Given the time to transmit a packet in Equation 1 the cost to transmit a message of length n is:

$$T_m = \gamma + \frac{n}{b} T_p$$

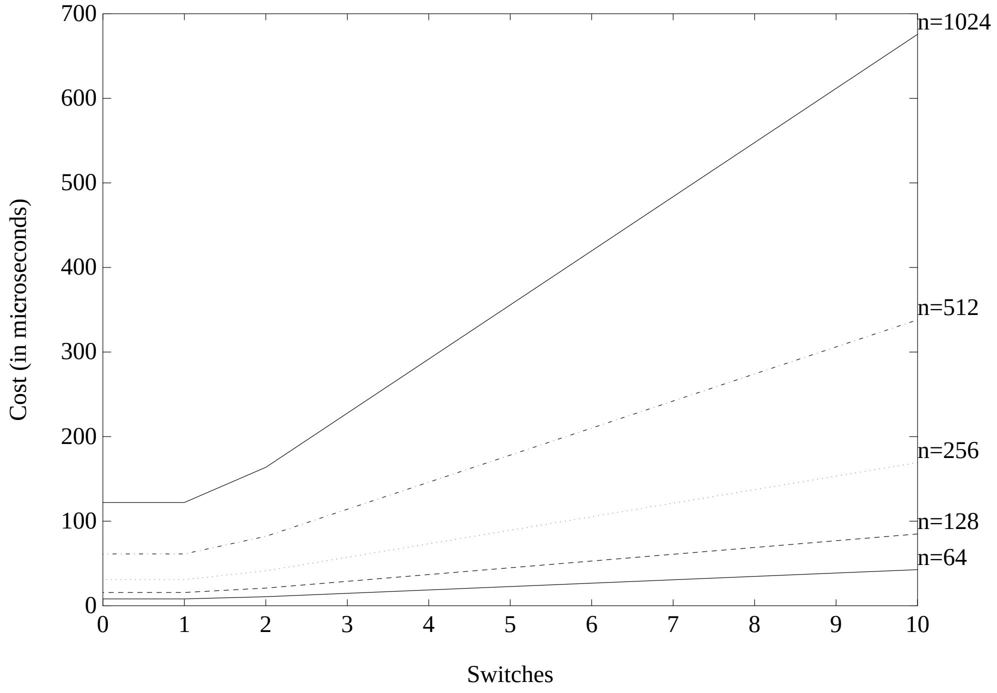
i.e.,

$$T_m = \max \begin{cases} \gamma + \frac{n}{b}(\beta + (h + b + 1)\alpha) \\ \gamma + \frac{n}{b}(2\beta + (2h + 1)\alpha + 2s\delta) \end{cases} \tag{3}$$

Graph 2 shows the cost for transmitting messages of various sizes across varying numbers of switch chips.

Equation 2 gives the point at which the message transmission cost begins to depend on s. With the estimated values for the parameters this means that the best utilisation of a link for a single channel communication is only obtained when at most a single switch chip is traversed. If there are two or more switch chips on the path then idle time of the source VCP and link engine are introduced whilst the ACK is awaited. PUMA machines consisting of more than 32 H1 processors connected by a switch network will require more than one switch chip and so the best link utilisation will not be obtained on a general purpose machine with only one channel to a link.

There are two ways of improving the utilisation of the link bandwidth. One way would be to introduce *excess parallelism* into the algorithm, placing m identical processes on each processor. In this case while a delayed acknowledge packet prevents transmission of the next packet on one channel on the link, packets on other channels mapped onto this link may be transmitted. If there are enough channels wishing to communicate at a time then the full utilisation of the link can be achieved. This method does not require



Graph 2: Cost of Transmitting a Single Message on one Channel

complex programming by the algorithm developer, but only that the algorithm is fine-grained enough to be able to use mp processes. It should be noted that compute processes may also execute in parallel with the communicating processes since they use separate execution units within the processor.

The second method for increasing link utilisation is to split the single message that a process wants to transmit into multiple blocks and transmit these blocks on several channels mapped onto the same link. This technique does not require a very fine-grained algorithm, but will involve the use of complicated communication routines to divide, transmit and reassemble the message correctly. Complete routines should be provided as part of a standard communications library so that the user does not need to consider their implementation. Listing 1 shows a full `occam` fragment that communicates a vector between two processes using multiple channels. Ways in which this can be implemented as a general library routine are unclear; for example, how can a user specify the source and destination processes to a library routine? This requires 'named process' functionality in the compiler.

The number of channels that need to be used to saturate a link depends on the number of switch chips that the message must traverse. The number of channels, c say, should at least be large enough so that when the first channel to transmit a packet has just received an ACK the last channel has finished transmitting its packet, i.e.:

$$\begin{aligned}
\text{time to output from other channels} &\geq \text{time between end of output and receive } \mathsf{ACK} \\
(c-1)(\beta+(h+b+1)\alpha) &\geq 2\beta+(2h+1)\alpha+2s\delta-\beta-(h+b+1)\alpha \\
c &\geq \frac{2\beta+2s\delta+(2h+1)\alpha}{\beta+(h+b+1)\alpha} \qquad (4)
\end{aligned}$$

Let us consider an example of a typical general purpose MIMD local memory PUMA machine. The machine has 256 H1 processors and 64 C104 switch chips. Each switch chip has the links from 4 processors connected to it using 16 of the switch chip's links. Twelve of the links on each C104 are used to configure the switch network as a double hypercube of dimension 6. The remaining 4 links on each switch chip may connect the network to IO devices and a host machine.

```
...  some code
-- Declare c channels
[4]CHAN OF []REAL32 chanvec:
PLACED PAR
  -- Source process
  [1024]REAL32 source.vec:
  INT block.size:
  SEQ
    ...  some code
    block.size := 256
    PAR i = 0 FOR 4
      chanvec[i] ! [source.vec FROM block.size*i FOR block.size]
    ...  some more code
  -- Destination process
  [1024]REAL32 dest.vec:
  INT block.size:
  SEQ
    ... some code
    block.size := 256
    PAR i = 0 FOR 4
      chanvec[i] ? [dest.vec FROM block.size*i FOR block.size]
    ...  some more code
...  some more code
```

Listing 1: Full `occam` Code to Communicate a Message using Multiple Channels

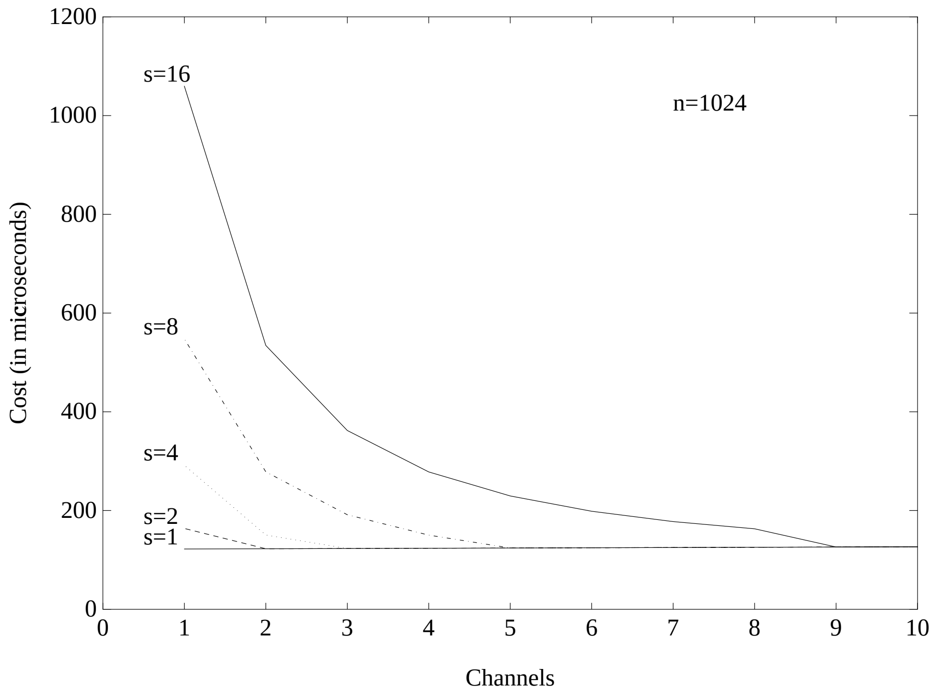
For this example machine with random routing of messages disabled the number of switch chips that a message traverses will be at most 6. Using $s = 6$ in Equation 4 suggests that about 3 channels will be required to communicate in parallel on a link to fully utilise the link bandwidth. It should be noted that if $s = 0$, i.e. the processors are connected together directly, one channel on a link saturates the link bandwidth.

The total cost of communication using multiple channels for a single message is derived as follows. If the number of channels used is not enough to saturate the link bandwidth then each channel transmits its n/cb packets with a wait for ACK between each one. The time between the output of the first packet on the first channel and the receipt of the last ACK on that channel is $(n/cb)(2\beta + (2h+1)\alpha + 2s\delta)$. The last channel receives its last ACK after a further time $(c-1)(\beta + (h+b+1)\alpha)$. On the other hand, if enough channels are used to saturate the link bandwidth transmission of packets is continuous with no waiting for ACK packets and the transmission cost is $(n/b)(\beta + (h+b+1)\alpha)$. In both cases there is the additional cost, $c\gamma$, incurred by the initialisation of the VLCBs by the integer unit. However, because of the parallel execution of the integer unit and the VCP, communication may take place on the first channels while the remaining channels are still being initialised. In this paper we have included the full cost of the channel initialisations in the models. The total cost for a multi-channel communication is:

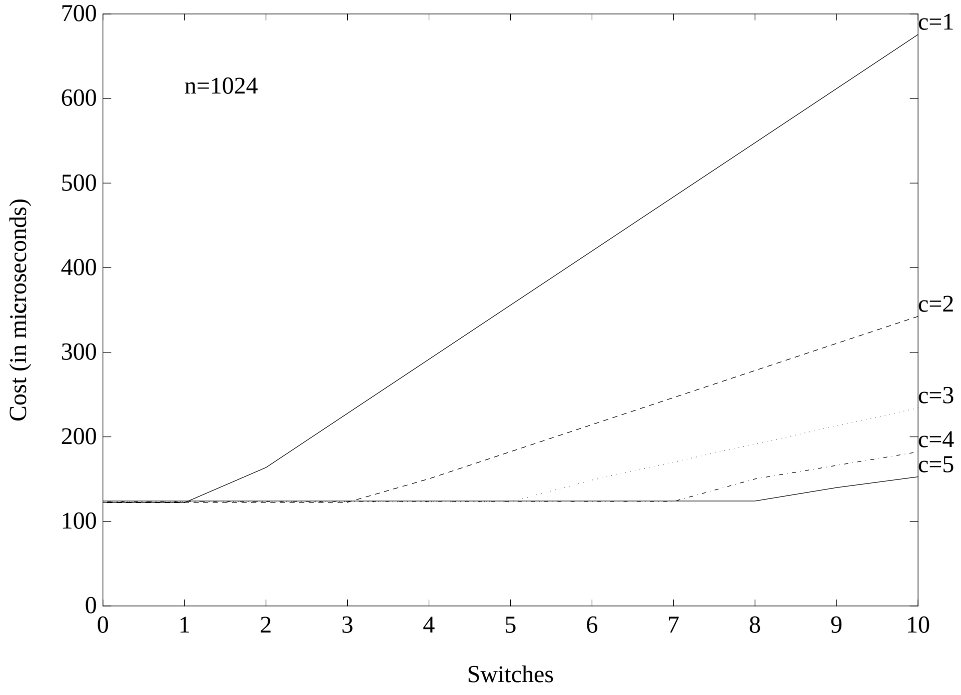
$$T_{mc} = \begin{cases} c\gamma + \frac{n}{cb}(2\beta + (2h+1)\alpha + 2s\delta) + (c-1)(\beta + (h+b+1)\alpha) & \text{if } c < \frac{2\beta + 2s\delta + (2h+1)\alpha}{\beta + (h+b+1)\alpha} \\ c\gamma + \frac{n}{b}(\beta + (h+b+1)\alpha) & \text{otherwise} \end{cases} \quad (5)$$

Graph 3 shows how the cost of the multi-channel communication of a single message varies with the number of channels and switches. Graph 4 shows that the cost of the communication will be independent of the number of switch chips traversed provided that enough channels are used.

At worst, this method gives as good a performance as when a single channel saturates a link, except for the cost of the additional channel initialisations. However, the overhead due to channel initialisation is small compared with the full cost of the communication. For large networks where the round trip



Graph 3: Cost of Multi-Channel Communication against Number of Channels



Graph 4: Cost of Multi-Channel Communication against Number of Switches

time to receive the ACK dominates the cost for the single channel communication, this method provides a significant improvement in performance. Hence this method of communication seems worthwhile in all practical situations whether c and s are known or not. If these routines are provided as part of a communications library then a fixed value of c, say $c = 4$, would probably be used which was sufficiently large to saturate the link bandwidth for communications across the largest number of switch chips that is likely to occur on a machine.

If small messages (a few packets in length) are transmitted then the multi-channel method still provides a significant performance improvement over a single channel communication provided that more than one switch chip is traversed (and thus a single channel cannot saturate the link bandwidth). Even with only one packet per channel the cost of the extra channel initialisations is only a small part of the total cost. For a very small message (up to one full packet i.e., $n \leq 32$) the simple single channel communication is better. Most numerical library programs will communicate messages of a wide range of lengths, so both multi-channel communications and single channel communications will be used. For example in Gaussian elimination multi-channel communications would be used for distributing the matrix, and a single channel might be used to broadcast pivot information at each step of the algorithm.

A further refinement to the technique for communicating a message between two processors would be to make use of several, l say, of the links on both the source and destination processor. Each link would have enough channels mapped onto it to saturate the link bandwidth. The VCP would first initialise a packet communication on a channel on one link. After that, in the previous multi-channel method, the VCP must wait until that packet had been output before it could initialise output of a packet on a different channel mapped onto the same link. To avoid this VCP idle time, the new method instead initialises the transmission of a packet on a channel mapped onto a different link. The VCP cycles round the links outputting packets on each in turn and returns to the first link hopefully before it has finished transmitting its packet. The VCP then outputs a packet on the first link again. This method again tries to saturate the bandwidth of each link but uses multiple links to increase the output rate of the source processor. Full utilisation of each link is only attained if the time taken by the VCP to service all four links is less than or equal to the time taken by a link engine to transmit a packet:

$$4\beta \leq (b + h + 1)\alpha$$

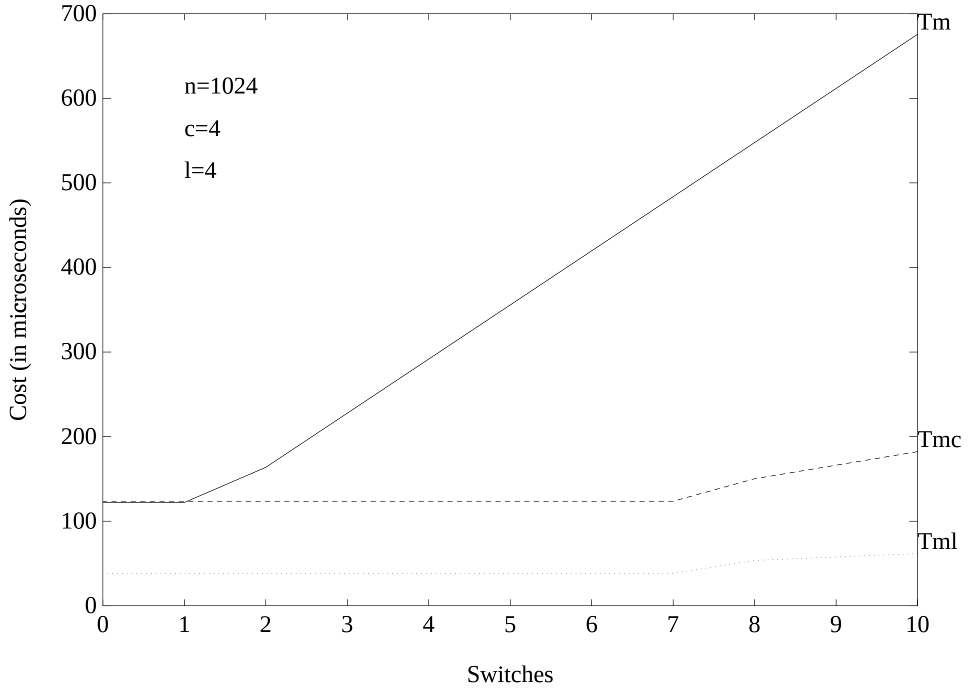
With the estimated values of Section 2 this is true and so we expect the VCP to be able to sustain communication on all the links at their full bandwidth simultaneously. This reduces the number of packets output on each link by a factor l, giving a cost of:

$$T_{ml} = \begin{cases} lc\gamma + \frac{n}{lcb}(2\beta + (2h+1)\alpha + 2s\delta) + (c-1)(\beta + (h+b+1)\alpha) & \text{if } c < \frac{2\beta + 2s\delta + (2h+1)\alpha}{\beta + (h+b+1)\alpha} \\ lc\gamma + \frac{n}{lb}(\beta + (h+b+1)\alpha) & \text{otherwise} \end{cases} \tag{6}$$

A comparison of the predicted costs of the three methods of communicating a message is shown in Graph 5. This graph shows that the use of multi-channel (and even better multi-link) message communication routines increases the communication performance significantly for any communication across more than one switch chip.

5 A Simple Model for Communications

The communications models presented above are not suitable for use in our algorithm models. They are far too complicated for our desired aim of ease of model development. The complexity is introduced to present an accurate model of the true costs of communication. However, it is unlikely that the degree of detail involved will give the expected accuracy in practice. There are many other factors which have not been included in the models which may be significant. These include the overheads of program code execution, inefficiency of the implementation language, transmission of partially full packets, memory bandwidth, cacheing, pipelining and parallel execution of the processor units. As far as the switch network is concerned two important unknowns which will affect costs are the actual number of switch chips traversed and the presence of hot-spots in the network causing collisions. On top of all of these concerns there is the problem of measuring idle time as the two communicating processes synchronise initially. This idle time may be



Graph 5: Comparison of the Costs of Transmitting a Message

due to the presence of other processes on the communicating processors or to imbalance in the workload of the two processes. For all of these reasons we need a much simpler model which will predict the communication cost adequately enough for us to be able to make decisions about the general performance of different algorithms on a PUMA machine.

Let us consider first a simple model for a single channel communication. As mentioned in Section 1 the simplest model $T = kn$ does not match the true cost well enough, since it ignores the significant start up time, γ, as the channel is initialised. The model $T = k_1 + k_2n$ includes this initialisation time but does not take account of the variation in cost with s. This variation is very significant (refer to Graph 2) and dominates the cost when a large number (say 10 or more) of switch chips are traversed. The extra cost incurred by switch chips scales with the message size since a delay is intoduced for each packet, not just for the whole message. Hence the delay is equally significant for both small and large messsages.

One way to model this delay is to introduce a further term in the cost expression involving n and s:

$$T = k_1 + k_2n + k_3sn \tag{7}$$

Unfortunately, the resulting cost expression is becoming as complicated as the models presented above. Also, we will not know the value of s to be used; even for a specific machine with a known number of switch chips in a known configuration each individual channel communication will cross different numbers of switch chips depending on the locations of the processors involved and whether random routing is enabled or disabled. It has been suggested that the expression $s = \log p$ could be used as an average, assuming the switch network was configured as a hypercube. This is not entirely satisfactory as the dimension of the hypercube, if a hypercube configuration is indeed used, will vary depending on the number of switch chips, and this is unlikely to be the same as the number of processors in the machine for reasons of economy. Alternative configurations of switch chips would give different relationships between s and p: a grid configuration would require $s \propto \sqrt{p}$; and a linear chain: $s \propto p$. A further complication in evaluating a value for s occurs if an algorithm uses only one partition of a large multi-user machine: in this case the number of switch chips is more likely to depend upon the total number of processors in the entire machine rather than the number being used for this particular computation. A reasonable solution seems to be to use Equation 7 as a model and expect a suitable expression for s to be specified by the user which reflects the size and configuration of his machine. This expression for s must give an approximation to the average number of switch chips in the path of a general message, for example half the worst case number. This model does not satisfy one of our main aims, that the cost model be independent of the switch network configuration, but this seems unavoidable.

A simple cost model for the multi-channel or multi-link communication methods is much easier to specify since this dependence on s is removed provided that the link is saturated. In this situation we can use the model $T = k_1 + k_2 n$ without losing any accuracy. The relationship between the multi-channel cost model and our simple model is given by:

$$\begin{aligned} k_1 &\equiv c\gamma \\ k_2 &\equiv \frac{\beta + (h + b + 1)\alpha}{b} \end{aligned}$$

In practice values for k_1 and k_2 would be estimated by running test communications programs on a real H1/C104 machine and fitting the model to the results obtained. This attempts to take account of the cost of program control flow and other features not included in the models.

We are now in the difficult situation where we propose two different cost models for point to point communications depending upon the implementation of that communication. The cost model for a numerical algorithm may contain terms due to both types of communication and will only be correct if each individual communication in the algorithm implementation has been performed using the method expected by the model. For numerical libraries this may not be too much of a problem; for many algorithms most communications are of vectors which may be large. In these situations multi-channel communications would be used throughout and the associated cost model would be independent of s and therefore applicable to machines with any switch network configuration. On the other hand many algorithms involve the communication of small amounts of data. These would be implemented using single channels for communication and hence the algorithmic cost would depend on the switch network.

6 Conclusions

In this paper, we have discussed various ways of implementing process to process communications on a PUMA machine and developed cost models to compare their performances. None of the models presented is ideal so a compromise between complexity and accuracy, biased towards maintaining simplicity, is proposed.

For single channel communication we propose to model the cost by:

$$T = k_1 + k_2 n + k_3 s n$$

This expression is dependent on the configuration of the switch network whilst the algorithm itself is not. It has been shown that utilising the full bandwidth of the links is not possible with only a single channel on a link when multiple switch chips are traversed by a message. To overcome this inefficiency, we suggest that a communications library is developed which includes a primitive routine to implement multi-channel communications for a single message. We model the cost of such a routine:

$$T = k_4 + k_5 n$$

This cost model is independent of the switch network. This method of communication may be used to transmit longer messages such as the vectors and matrices in numerical algorithms. A further set of library routines implementing higher level communications operations such as broadcast and distribute could use this primitive to improve their performance.

This work complements the work presented in [3, Section 6] which describes the costs of link communication in terms of the number of cycles required by the various processing units to perform a communication. That work also shows the effect on performance of partially full packets. It is comforting to note that these two different approaches broadly agree on the costs predicted for channel communication on a PUMA machine.

Examples of cost models for numerical algorithms using the suggested communications models may be found in several PUMA working papers[1, 2, 4].

References

[1] Cliff Addison, Gabriel Howard, Tim Oliver, and Chris Phillips. Interim report on algorithm design. PUMA Deliverable 5.2.1, University of Liverpool, August 1990.

[2] Rod Cook. Timing models for preconditioned conjugate directions. PUMA Working paper 13, University of Liverpool, August 1990.

[3] M. Goldsmith, C. Liddiard, D. May, C. O'Neill, M. Poole, B. Roscoe, A Sturges, and P. Thompson. Architecture consolidated report. PUMA Deliverable 1.1.1, INMOS, August 1990.

[4] Gabriel Howard. Timing models for Gauss elimination on the H1 transputer. PUMA Working paper 11, University of Liverpool, May 1990.

[5] Gabriel N Howard. Solving simultaneous linear equations on transputer arrays. Working paper, Centre for Mathematical Software Research, University of Liverpool, 1988.

[6] INMOS. H1 transputer: Advance information. Technical report, June 1990.

[7] INMOS. IMS C104 packet routing switch: Advance information. Technical report, September 1990.

[8] Tim Oliver. Calculating eigenvectors on transputer arrays. Working paper, Centre for Mathematical Software Research, University of Liverpool, February 1988.