Modelling Communication in Message-Passing Multicomputers

Keywords: Parallel Computers, Message-passing, Performance Modelling

This paper is written at a time when the model is still in the development phase, but the main features are present. The following features highlight the key contributions of this work which differ from work published earlier.

The model extends the existing communication modelling concept to include the additional costs in throughput and latency at the higher level of the communication system — the software level. Also, at the hardware level, increased accuracy is sought by modelling the complexity of routers with respect to conflicting design and implementation issues and modelling traffic using realistic characteristics of parallel applications.

Modelling Communication in Message-Passing Multicomputers

Abstract

To improve performance in parallel computers, the challenge is to reduce the communication bottleneck by decreasing latency and increasing throughput. Recent advances in interconnection network design have improved throughput. However, these design changes require additional software support thus increasing latency at the software level. As such, the total latency of the communication system is increased. Similarly, improvements at the software level have led to increased complexity at the hardware level and again an overall increase in latency. As such, a performance analysis tool is required which models both the software and hardware levels of the communication system.

This paper presents an overview of a proposed performance model of communication systems in message-passing multicomputers. The model aims to provide a platform for analyses of the communication system as an integrated part of a parallel computer. As such, both the software and hardware (existing and proposed) components of communication systems are modelled with respect to the performance metrics — throughput and latency. Network traffic is modelled using realistic characteristics of a restricted set of parallel algorithms.

1 Introduction

This paper presents the motivation behind and key features of a model of communication systems in message-passing distributed-memory parallel computers. It is written at a time when the model is still in the development phase, but the main features are present.

There are three distinctive features of the model. First, it views the communication system as an integrated part of a parallel computer instead of a self contained subsystem. It extends the existing communication modelling concept to include the additional costs in throughput and latency at the higher level of the communication system — the software level.

Secondly at the hardware level, increased accuracy is sought by modelling the complexity of routers with respect to conflicting design and implementation issues.

Finally to represent network loading, realistic traffic characteristics taken from the communication pattern of chosen algorithm classes will be obtained from experiment on an existing parallel computer system. Benchmark algorithms will be used where the required algorithms are unavailable. The use of realistic characteristics enables not only queueing and decision delays at the hardware level to be modelled but also the demands on the operating system to be modelled with increased accuracy.

It is proposed that the model be based on a combination of mathematical equations and a simulator. Outputs will be expressed in time units and will reflect both the average and the standard deviation of latency and network throughput.

Work to date has focused on development of the architecture of the model and development of the required mathematical equations, especially at the network level. The principle latency equations which describe the total latency in the communication system are presented and justified in this paper.

This paper is organised as follows. Section 2 explains the need for models of communication systems in parallel computers and describes other work in related areas. Section 3 describes the architecture of the model. In section 4 a description of the analytical equations is given upon which, the model is being built. Ongoing and future work is presented in section 5 and a summary of the work presented is given in section 6.

2 Background and Motivation

In parallel computers, much research has been dedicated to improving throughput and reducing latency in interconnection networks. Recent research has led to increased throughput but more complex routers with the addition of virtual channels [Dea87] or the use of various new adaptive routing schemes [Dua91]. It is acknowledged, however, that software is a major bottleneck in communication systems. Especially for small messages, as is the case in fine-grained processing, the software overhead constitutes a large percentage of message latency [HB90]. Hardware level improvements such as the implementation of advanced routing schemes, as discussed above, give rise to increased latency at the software level [KC94b] and therefore increase the software bottleneck.

Research into reducing the delay incurred at the software level has focused on three main areas. One field of thought is to move some of the software functionality into hardware e.g. compressionless routing [KLC94]. Another, is to improve the interface to the hardware i.e. the Shrimp project [Bea95], Active Messages [VEea92] and the J-Machine [Dea89]. However these solutions lead to increased hardware complexity and as such an increase in latency at the hardware level. Another solution, as seen in the Intel Paragon [Int91], is the addition of a co-processor to handle communication operations. This increases the overlap of computation to communication operations , thus hiding more of the communication latency, but results in an increase in hardware costs.

It can be seen from the above discussion that there exists both a software versus hardware and a throughput versus latency trade-off in the design of communication systems. Parallel computer systems are emerging fast, but as suggested by Spertus [Sea93] the time has come to shift from the 'big new ideas' to careful quantitative analysis. An analytical tool is required which can describe and evaluate a given communication system design at both the software and hardware level with regards to the two performance metrics — throughput and latency.

Another factor which has a large effect on the performance of message-passing multicomputers is the communication patterns and message characteristics of parallel algorithms [HB90]. These characteristics are represented in the interconnection network as network traffic. Assumptions such as Poisson message arrival, exponential distribution of message length and even distribution of message destination [KK79] are often used to describe network traffic for performance analysis purposes. Other assumptions are based on task granularity and inter-task dependencies. However, it has been shown by Hsu and Banerjee [HB90], through experimentation on the hypercube topology, that many of these assumptions do not match the characteristics of real applications running on parallel computers. As such, a more accurate description of traffic is required for realistic performance analysis.

Traditionally performance analysis at the network level has been based on cycle counting. However comparison of the results of various studies cannot be made as the network clock cycles differ between featured parallel computers. It has also been shown by Patel and Jayasimha [PJ95] that cycle counts can give misleading results. A more accurate method is to base results on time units. Average message latency is a frequently used performance metric. However, it is also shown in [PJ95] that the standard deviation in message latency can be quite large and as such standard deviation results should be included in performance analysis.

It has been the custom in interconnection network performance analysis to assume a fixed delay through each of the routers independent of their complexity [Aga91]. However, it is shown by Chien [Chi93] that router complexity has a significant effect on the delay through the router. It is

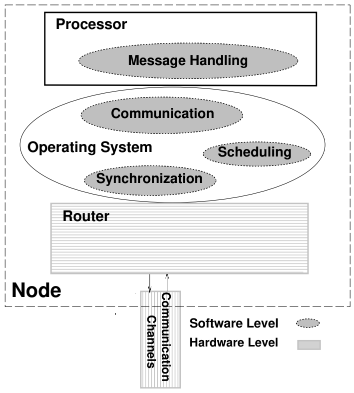

Figure 1: The Communication System of a Generic Parallel Computer Node

also important to note that while complexity effects the unloaded delay, that network traffic gives rise to decision and queueing delays and as such traffic characteristics must be included in router analysis.

3 The Model Architecture

In a message-passing multicomputer each processing node contains an autonomous processor and it's local memory that is physically separate and logically private to the memories of the other node computers. Interprocessor communication occurs by routing messages explicitly through a message-passing network.

An interface is required to control the flow of information between the processor and the interconnection network. The complexity of this interface depends on the quality of service offered by the computer system, the topology of the interconnection network and the division of work between the interface and the compute processor itself. This interface is termed a router.

A node is represented in the model by only those components relevant to message passing as shown in figure 1. It is assumed that there exists a direct coupling between a processor and it's router.

The software level includes both software overheads - communication related system-time and processor idle time. Software overheads include not only the time taken to process message related system calls, but also the time taken to execute message handling code. These message related system calls handle both communication and synchronisation inherent in parallel applications [ZYH94]. A message passing library, which consists of compiled blocks of code representing message handlers, is stored in memory or registers which are activated to handle outgoing and incoming messages.

The routers' external communication channels (router/network), internal communication channels (router/processor) and the router itself make up the hardware level of the communication system.

The router may be considered as consisting of three main parts, as shown in figure 2. The central part implements the functionality of the router — switching, routing and flow control. The router applies this functionality to traffic entering and leaving at the two outer parts i.e. at the processor and network interface.

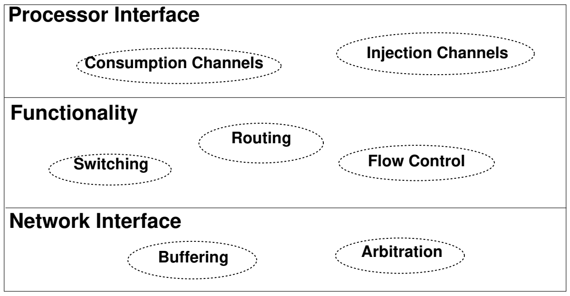

Figure 2: A Functional Router Description

4 Derivation of the Principle Latency Equations

The total delay incurred in communicating between processes across an interconnection network includes not only the interconnection network delay but also the delays incurred at the higher levels of the communication system. As such, the overall latency in the communication system — *Total Latency*, can be expressed as:

$$TotalLatency = ProcIdleTime_{comm} + SoftOhds + NetLatency \tag{1}$$

where ProcIdleTime$_{comm}$ represents communication related processor idle time, SoftOhds represents software overheads and NetLatency represents Network latency.

Communication Related Processor Idle Time

Processor idle time occurs when a processor, which is currently running a process — a task of the program, is either finished executing the process or a context switch has occurred and is waiting on the operating system to schedule a new process. It also occurs when a process makes a call to the operating system and is waiting for the call to return.

In the case of a communication related system call, a delay is incurred while the call itself is handled and the required communication is set up. In the case of synchronous communication, the process waits until the communication is complete. However, in asynchronous communication, the operating system returns control to the running process as soon as the communication is set up.

To determine processor idle time due to communication events, only the time taken for communication related context switching need be accounted for. The time taken for communication related system calls is included in the software overhead analysis.

Assuming no process migration between nodes, communication related processor idle time can be expressed as:

$$ProcIdleTime_{comm} = t_{exec} - (t_{CPU} + ProcIdleTime_{non-comm}) \tag{2}$$

where t_{exec} is the total process execution time, t_{CPU} is the active processor time and ProcIdleTime$_{non\text{-}comm}$ is the non communication related processor idle time.

Software Overheads

Software overheads, also referred to as start-up latency [NM93, BP93], includes the parallel primitive execution time for message handling by the operating system and the execution time of message handling code. Software overheads are incurred at both the sender and receiver. However, overheads incurred at the sender are much lower than those at the receiver. This can be explained with respect to the applications running on a parallel computer [Sea93]. A sending operation is synchronous with the application at the source i.e. the values required are available to send. At the destination values will tend to be stored until the receiving application is scheduled. It should be noted that implementation of Active Messages [VEea92] eliminates this additional cost at the receiver.

It is assumed that interaction between the software and hardware levels of the communication system occurs only at the source and destination nodes of a packets path. As such, software overheads (SoftOhds) can be expressed as follows:

$$SoftOhds = (t_{os-send} + t_{mh-send}) + (t_{os-recv} + t_{mh-recv}) \tag{3}$$

where $t_{os\text{-}send}$ and $t_{os\text{-}recv}$ represent the execution times of the relevant operating system communication calls and $t_{mh\text{-}send}$ and $t_{mh\text{-}recv}$ are the execution times of message handling code at the sender and receiver respectively.

Network Latency

Earlier analysis of network latency often included only the communication delay across the interconnection network [Aga91]. However, the delays encountered between the source and destination node and the router itself should be included in the network latency. As described in [BP93] their contribution to end-to-end delay may be substantial especially at the destination where bottlenecks may appear.

Network latency (NetLatency) is described as follows:

$$NetLatency = t_{inj} + t_{cons} + t_{comm} \tag{4}$$

where t_{inj} represents queueing for the interconnection network (injection channels); t_{cons} is queueing for the processor (consumption channels) and t_{comm} is the communication delay.

Delays incurred at the injection channel are dependent on the effective bandwidth of the network:

$$EffectiveBandwidth = \frac{1}{t_{wire} + t_{node}} \tag{5}$$

where t_{wire} is the transmission time between nodes and t_{node} is the switching time at a node.

If the effective bandwidth is sufficient in that the network can handle all messages sent to it then the injection delay can be neglected [Chi93, BP93]. This assumption is often seen in the literature. However, as can be seen from equation 5, node delay is inversely related to the effective bandwidth and therefore more complex routers, which give rise to larger node delays, decrease the effective bandwidth and can lead to injection delays. Therefore it has been chosen to include injection delays in our analysis.

In studies of advanced routing techniques, infinite internal channels — consumption channels are assumed [LH91]. However if the bandwidth is high and there are insufficient consumption channels to handle traffic arrival in parallel, then traffic will have to be queued and handled sequentially at the destination. An additional queueing delay may, therefore, be seen at the exit of the network. There is also an additional decision delay due to arbitration of messages when more than one message arrives at the same time. As such, lack of sufficient consumption channels can be a large bottleneck in the interconnection network. A solution to these problems is multiple consumption channels. The chosen number of consumption channels is given as a parameter to the model.

Communication delay, t_{comm}, arises from the node delays (switching delays) and wire delays across the network. Node delay is dependent on the routing algorithm and particularly on the degree of routing freedom; how arbitration of resources occurs and how deadlock prevention is achieved. Node delay is encountered not only at the source and destination node but at the intermediate nodes aswell. Wire delay is made up of not only the propagation delay (t_{phy}) at each stage but the delays through the sending (t_{send}) and receiving circuitry (t_{recv}) aswell.

There is an important distinction between the node delay encountered by the header (t_{node}) and that encountered by each of the remaining flits (t_{prop}). Since the decision delay is only encountered by the header then the delay encountered by the header is greater than that of the other flits [BP93]. It can be shown [Dal90] that this can have a large effect in high diameter networks with short messages. The effect is, off course, dependent on the difference in size of the respective delays. However, for long messages the effect may be quite insignificant. As such, it is chosen to separate these two delays and model the size of the packet in terms of the number of flits in the header and body respectively.

Communication delay is expressed as:

$$t_{comm} = t_{header} + t_{body} \tag{6}$$

where t_{header} is the total delay encountered by the header flits and t_{body} is the total delay encountered by the the rest of the packet.

As stated earlier, there exists a bottleneck at the software level of the communication system and a solution to this problem may be to move some of the software to hardware. This solution will effect only the source or destination node. As such, it has been chosen to separate the delays encountered at the source and destination from those encountered at the intermediate nodes.

Traffic in the network can lead to contention. Contention resolution is a part of the switching technique chosen for a given communication system. To increase flexibility in the model, it has been chosen to parameterise the model with a choice of switching techniques. Wormhole routing which uses a blocking solution and virtual-cut-through and store and forward routing which use a buffering method are included in the model. As such, the components t_{header} and t_{body} are first expressed in terms of wormhole routing and these expressions are then expanded to reflect virtual-cut-through and store and forward routing respectively.

In wormhole routing flits flow through the network in a pipelined fashion. When the header flit blocks, assuming only a one flit buffer at each node on route to the destination, the following flits are blocked in-place. When the path is free, the head can continue and the flits again follow in a pipelined fashion. The additional delay due to contention is modelled as a queueing delay for the header flit and as such is included in the component t_{node}. Equation 6 is, therefore, expanded as follows:

$$t_{header} = h * (t_{src} + ((D - 1) * (t_{node}) + t_{dest}) + (D * t_{wire})) \tag{7}$$

$$t_{body} = (L - h) * (t_{wire} + t_{prop}) \tag{8}$$

$$t_{wire} = t_{send} + t_{phy} + t_{recv} \tag{9}$$

where t_{node},t_{prop},t_{send},t_{phy}, t_{recv} are as described earlier. D — the network diameter; L — the number of flits in a packet and h — the number of flits in a header are parameters to the model.

In virtual-cut-through routing the header delay is as in equation 7. However, unlike wormhole routing, sufficient buffering for the complete packet must be available at each node. The delay experienced by the body is effected by the amount of buffering actually used at each node. In the best case the buffering used is equivalent to wormhole routing and in the worst case, equivalent to store and forward routing. When the header is blocked, the following flits begin to fill up the packet buffer at the blocked node. When the header can progress, the flits again follow in a pipelined fashion. For simplicity, it is assumed that if the packet blocks, blocking continues until after the last flit has been stored at the blocked node. As such we have:

$$t_{body-VCT} = (B+1) * t_{body} \tag{10}$$

where B is the number of times blocking occurs.

In store and forward routing the header delay is as in equation 7. However, the header is held at a node until all the flits of the packet are received. In wormhole routing the flit delay is only seen at the final hop — due to the pipelined transmission. However, in store and forward routing, flit delays are encountered at each node. There is no additional buffering overhead when contention occurs as all the flits are stored at the blocked node. This can be expressed as:

$$t_{body-SF} = D * t_{body} \tag{11}$$

Summary

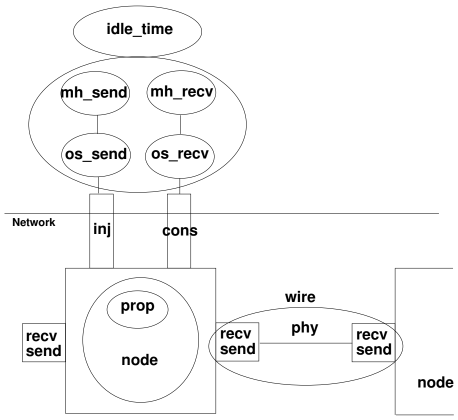

Figure 3: The Components of Latency in Message Transmission and Reception

The above equations form the basis of the model. These highlight the key components that require to be developed — as shown in figure 3. At the software level we have the components processor idle time and software overheads — $t_{os\text{-}send}$, $t_{os\text{-}recv}$ and $t_{mh\text{-}send}$, $t_{mh\text{-}recv}$. At the hardware level the components are t_{inj}, t_{cons}, t_{node}, t_{prop},t_{send}, t_{phy} and t_{recv}.

5 Ongoing and Future Work

A study of the CM-5 architecture by Karamcheti and Chien [KC94a] showed that 50-70% of software costs are directly attributable to interconnection network features. These features are arbitrary delivery order, finite buffering and limited fault tolerance. A study of an existing message-passing multicomputer will be undertaken to characterise the components of software overheads at both the sender and receiver with respect to network features. These features will then be broken down to their component form, similar to the work currently begin undertaken at the network level. Proposed solutions to the software bottleneck problem can then be modelled by removing the required components from the software level and adding the comparative components to the hardware level.

Node delay is represented in the network latency equation as the components t_{node} and t_{prop} for the header and remaining flits of a packet respectively. If the node is considered as an isolated module, then it can be described as a collection of connected components. The node delay can then be categorised as including both the component delay — the time taken for a flit to flow through the components, and the inter-component delays. The component delays arise from the basic gate delays that make up the components and their complexity is dependent on the size of the architecture and the implementation technology. However, as the node is part of a communication system with traffic flowing through, then queueing and decision delays which are dependent on network loading and the arbitration policy, must be included. A detailed analysis of router design issues has been undertaken and the results of this analysis will be used to parameterise node delay.

6 Summary

This paper has highlighted the need for a platform for analysis of the communication system as an integrated part of a parallel computer. It introduces an ongoing project which aims to meet this requirement. The main features of the model of communication are presented along with the analytical equations upon which the model is being built.

The model being developed will provide a useful tool for parallel computer designers to study the viability of a communication system design with a specific class of applications in mind. It will also provide a tool for programmers (users) to support performance analysis of a message-passing multicomputer with respect to an algorithm chosen from a set of parallel algorithm classes. For system administrators a similar evaluation of system throughput can be obtained.

7 Acknowledgements

I would like to thank Mohammed Ould-Khoaua and James Strout for helpful reviews of earlier versions of this paper and Lasse Natvig for his continuing support and advice in this ongoing project.

References

[Aga91] A. Agarwal. Limits on Interconnection Network Performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4), Oct. 1991.

[Bea95] M. A. Blumrich et al. Virtual-Memory-Mapped Network Interfaces. *IEEE Micro*, pp 21–28, Feb. 1995.

[BP93] S. Balakrishnan and D. K. Panda. Impact of Multiple Consumption Channels on Wormhole Routed k-ary n-cube Networks. In *Proc. of the Intern. Parallel Processing Symposium '93*, pp 163–167, 1993.

[Chi93] A. A. Chien. A Cost and Speed Model for K-Ary N-Cube Wormhole Routers. In *Proc. of Hot Interconnects '93*, Aug. 1993.

[Dal90] W. J. Dally. *Network and Processor Architecture for Message-Driven Computers*. MorganKaufmann Publishers, Inc., 1990.

[Dea87] W.J. Dally et al. Deadlock-Free Message Routing in Multicomputer Interconnection Networks. *IEEE Trans. Comp.*, C36(5), 1987.

[Dea89] W.J. Dally et al. The J-Machine: A Fine-Grain Concurrent Computer. In *Information Processing 89, Proceedings of the IFIP Congress*, pp 1147–1153, August 1989.

[Dua91] J. Duato. On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Design Methodologies. In *Parallel Architectures and Languages European Proceedings*, pp 390–405, June 1991.

[HB90] J. Hsu and P. Banarjee. Performance Measurement and Trace Driven Simulation of Parallel CAD and Numeric Applications on a Hypercube Multicomputer. In *Proceedings of the 17Th Annual International Symposium on Computer Architecture*, pp 260–269, May 1990.

[Int91] Intel Corporation. *Paragon XP/S Product Overview*, 1991.

[KC94a] V. Karamcheti and A. A. Chien. Do Faster Routers Imply Faster Communication? In *Proceedings of Parallel Computer Routing and Communciations Workshop*, May 1994.

[KC94b] V. Karamcheti and A. A. Chien. Software Overhead in Messaging Layers: Where Does the Time Go? *ASPLOS VI*, pp 51–60, Oct. 1994.

[KK79] P. Kermani and L. Kleinrock. Virtual-Cut-Through: A New Computer Communication Technique. *Computer Networks*, 3 pp 267–286, 1979.

[KLC94] J. H. Kim, Z. Liu, and A. A. Chien. Compressionless Routing. In *Proceedings of ISCA'94*, April 1994.

[LH91] D.H. Linder and J.C. Harden. An Adaptive and Fault-Tolerant Wormhole Routing Strategy for K-Ary N-Cubes. *IEEE Trans. Computers*, 40(1) pp 2–12, Jan. 1991.

[NM93] L M Ni and P K McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, pp 62–77, Feb 1993.

[PJ95] K C Patel and D N Jayasimha. Measures of Message Latency for Wormhole Routed Networks. Technical report, Dept. of C and IS, Ohio State University, 1995.

[Sea93] E. Spertus et al. Evaluation of Mechanisms for Fine Grain Parallel Programs in the J-Machine and the CM-5. Technical report, AI, Lab MIT and EECS, University of California, 1993.

[VEea92] T. Von Eicken et al. Active Messages: a Mechanism for Integrated Communiction and Computation. In *Proceedings of the 19Th International Symposium on Computer Architecture*, May 1992.

[ZYH94] X Zhang, Y Yan, and K He. Evaluation and Measurment of Multiprocessor Latency Patterns. In *Proc. Of the 8Th Int. Parallel Processing Symposium*, April 1994.