THE POST OFFICE – COMMUNICATION SUPPORT FOR DISTRIBUTED ENSEMBLE ARCHITECTURES

K. S. Stevens
S. V Robison
A. L. Davis

Schlumberger Palo Alto Research
Computer Aided Systems Laboratory
Palo Alto, CA.

Abstract

We present the design of a communications processor which operates concurrently with a cooperating program execution processor. This component provides hardware support for the runtime communication overhead inherent in multiprocessor systems. The communication and execution processors are coupled to provide an efficient processing element for constructing *distributed ensemble architectures*. Design issues for a general *Post Office* are discussed along with our specific implementation and architecture.

Introduction

This paper presents the design for a general communications control component for multiprocessor architectures. This effort is part of a research project to construct a specific multiprocessor designed to support symbolic computation[1]. In general, however, we are interested in a particular class of multiprocessor systems which we will call *distributed ensemble architectures*. Distributed ensemble architectures have the following properties:

- They are comprised of an arbitrary number of homogeneously replicated, autonomous processing sites (called *processing elements* or PE's) that contain both processor and memory. Any PE is capable of evaluating all programs which can be stored in its local memory.
- The PE's are interconnected with a *communication topology*.
- The PE's cooperate as a parallel computing system to evaluate single complex parallel programs.
- The processors communicate with each other during program evaluation by passing messages between sending and receiving PE pairs.

In such systems, the local computation sites behave in a manner similar to objects in Smalltalk[2], and other widely used object oriented programming models. Program execution, control, and synchronization are all a direct result of objects communicating via messages. The goal of these systems is to significantly increase performance over conventional sequential systems through the exploitation of concurrency.

Unfortunately, multiprocessing systems contain an intrinsic performance limitation. Message passing, and its associated overhead, is a necessary evil when exploiting concurrency as the primary performance impetus, because:

- Cooperative multiprocessing implies communication.
- Communication implies overhead.

It is very important to ensure that the overhead of communication does not negate the performance benefits of the concurrent operation of the processing elements. If a general mechanism can be found which permits processing and communication to operate in parallel, then this difficulty may not occur in practice. The communications controller described here (subsequently called the *Post Office*) supports significant levels of parallelism in communication and processing. In particular, it represents what can be viewed as a significant reallocation of the hardware budget to create an autonomous communications server that is capable of performing all physical message delivery duties without interfering with the processing duties being performed by the processing elements. This approach is rather different from components such as the INMOS Transputer[5] which steals cycles from the processor in order to manage the interprocessor communication activities.

The coupling of such a Post Office component with an evaluation processor could lead to a significantly improved processing element for distributed ensemble architectures. As a result, one of the most interesting aspects of the architecture is the notion of making communication a first class activity that is directly supported in hardware. It is important to note that this model assumes that the processor has something to do while the Post Office is handling message traffic. This is likely to be the case when the the processor itself supports multi-tasking, and where blocked tasks due to message sends which require a reply can be suspended and a new active task can be evaluated.

The duties of a general purpose Post Office component include:

- Transformation of message contents from their internal representation in the local processing element to the format required for transmission over the communication topology.
- Performing message routing decisions.
- Controlling physical message transmission.
- Error checking and recovery.

Most of these activities are general purpose operations necessary to provide high performance communication capabilities in any distributed ensemble architecture. The one area of specialization is in the routing mechanism itself. Routing decisions are directly influenced by the choice of a particular communication topology. A general version of a communications processor such as the Post Office would have the routing algorithm stored as a program which could be modified to accommodate various topologies. The interconnection topology presented in the next section is intrinsic to the FAIM-1[1] architecture, and has the advantage of allowing a very simple routing algorithm which will be implemented in hardware for increased performance.

Topology

There are many possibilities for the interconnection topology, which is apparent from current research efforts such as the Cosmic

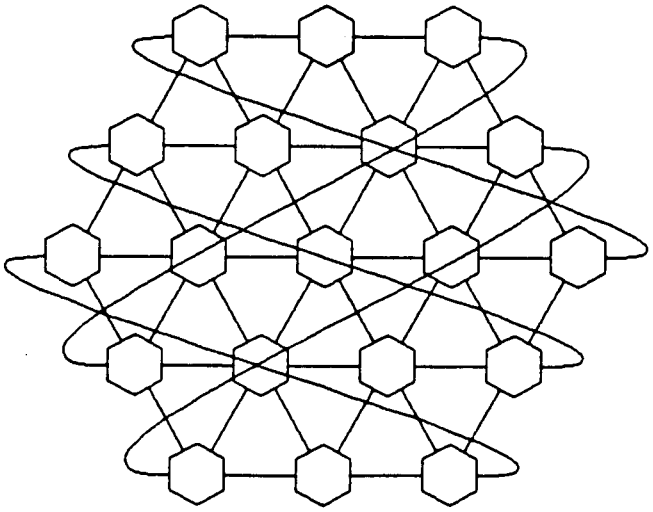

Figure 1: E-3 Processing Surface

Cube[12], Butterfly[4], and DADO[15]. The core of the topology developed for the FAIM-1 is a simple six ported PE interconnected with planar wiring that forms a *processing surface*[14]. This configuration implements a fully distributed multiprocessor system without shared memory. In Figure 1 the basic topology is illustrated along with the wraps that complete the full interconnect structure. For purposes of illustration only one wrap is shown; in the complete topology all edge ports are connected via two additional sets of wraps like the one in the diagram.

When wires leave the planar surface through the processors at the periphery they are folded back onto the surface using a three axis variant of a twisted torus[9]. This folding scheme results in a completely homogeneous interconnect and provides a provably minimal switching diameter for hexagonal meshes.

A primary advantage of the hexagonal interconnection scheme is the scalability of the architecture. The size of the processing surface is defined by the number of processors on each edge of the hexagonal surface, and is referred to as an *E surface*. The number of processors in a surface scales as $3E(E-1)+1$ in relation to the E size, or the number of processors per edge. For example, the E-3 processor surface illustrated has three processors on each of the six edges and contains a total of 19 processors.

The ability to scale an architecture permits more concurrency and processing power. However, the probability of failure of a component increases as more components are added to the system[13], making fault tolerance an important aspect of highly replicated architectures. Distributed ensemble architectures intrinsically contain redundant elements which could be used to support fault tolerant behavior. Koren has shown that hexagonal meshes could use these redundant elements to achieve a fault tolerant topology[3]. The hexagonal network itself also tolerates non-malicious connection failures because messages from a common source may take different paths to reach a common destination.

Direct Communications Support

There are four common delivery mechanisms that are options for the physical realization of a Post Office's communications network:

- Processor Driven Communications (cycle stealing)
- Virtual Circuit
- Message or Packet Switch
- Virtual Cut-Through

As mentioned in the introduction, cycle stealing is inconsistent with our overall architectural goals, since it couples the communications and processing components and reduces parallelism.

The virtual circuit approach sets up a dedicated path through communication links between the sender and the receiver. A message request must be routed through the network, establishing a trail of previously free links to the destination that must be acknowledged before any data transmissions can occur. There is typically a relatively large path setup time. Once this path has been established, the transmission delay of the transaction is only limited by the propagation time of the signals. The dedicated path will continue to exist between the two communications processors until the transaction is completed.

In a message switching or store and forward network, no physical path is established in advance between the sender and receiver. Instead, when a message is to be sent, it will be stored in the first communications processor and forwarded, one hop at a time, until it reaches the destination. Each block is received in its entirety, checked for errors, and then retransmitted.

Packet switching places a limit on the size of data blocks that can be transmitted. This allows the network to be more responsive because links are not blocked for extended periods of time while transmitting large messages. Block size is fixed, simplifying the buffering of messages and allowing packets to be stored in main memory. Large messages must be decomposed into packets and reassembled at the destination. However, the individual packets of large messages can be pipelined through many nodes in the network reducing overall message latency.

The key difference between packet switching and virtual circuit approaches is that links must be statically reserved when making virtual circuit connections. Any unused bandwidth will be wasted. In packet switching, links are acquired and released as they are needed. The same link can therefore interleave packets of different messages. However, packets may arrive out of order requiring reordering at the destination.

A hybrid of the packet switching and virtual circuit methods called *virtual cut-through*[6] is used by the Post Office. This method is based on the packet switching model with the added capability that the packet header can be examined as soon as it arrives to determine the subsequent outgoing link. If that link is free, retransmission is begun immediately. However, if the link is busy, the packet is stored and forwarded in the usual manner. Under light load conditions, most packets will travel straight through the network without queuing delays.

Post Office Architecture

Operational Responsibilities of the Post Office

The Post Office is an autonomous message controller which delivers messages concurrently with program execution in the processor. The Post Office for the FAIM-1 architecture can potentially be active on all six external ports and the internal PE port simultaneously. All message delivery is controlled by the Post Office, freeing the processor from unnecessary communication overhead.

There is a single uniform message format that masks all hardware characteristics of the delivery mechanism from the processor.

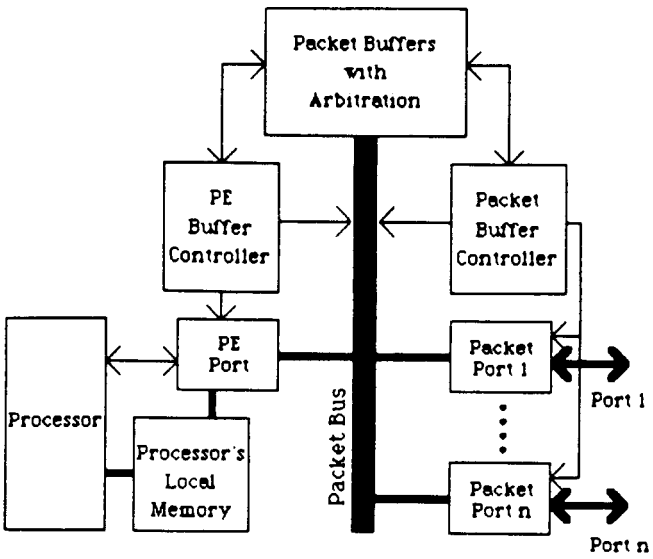

Figure 2: Post Office Block Diagram

Messages consist of a message header and variable length body. To send a message to a remote processor, the processor must build a header that indicates, among other things, the destination of the message. The destination processor will be notified when new messages arrive.

The Post Office is responsible for the physical delivery of messages across the communication network. It therefore must read and write from the local processor's memory and transfer messages between adjacent Post Office nodes.

The packet switching model employs fixed-size packets to be transmitted across the network. This requires a facility to decompose messages into packets and to reassemble the individual packets into messages at the destination node. The packet switching model also requires a reserved amount of memory at each node for packet buffering. A buffer access protocol is needed that allows efficient concurrent access by a plurality of communication ports.

Reliability demands that messages be delivered error-free, requiring a CRC check and retransmission of faulty packets. To avoid deadlock, a mechanism must be employed that either guarantees deadlock will not occur, or that the probability of occurrence is much less than the probability of an unrecoverable component failure.

The Post Office contains routing hardware to calculate the correct ports through which a message should be transferred. Efficiency in the communications controller can be greatly enhanced by re-routing messages around congestion. This mechanism is extended to introduce fault tolerance by routing messages around non-functional nodes and ports.

Post Office Components

The Post Office architecture has three component types, as shown in Figure 2.

- The Port Controllers
- A Buffer Pool
- The Buffer Controllers

Most of the functions of the Post Office are performed by independent ports. There are two port types, packet ports and PE ports. The packet ports transmit packets between the adjacent Post Offices, while the PE port transfers messages to and from the processor's local memory. There are several packet ports and one PE port in a Post Office node.

Both types of ports contain the routing hardware so that packets can be immediately forwarded to the next port. The packet ports have the additional tasks of updating the message header, checking for transmission errors, and retransmitting faulty packets. Packet creation, message assembly and packet ordering is done by the PE port.

The function of the buffer controllers is to arbitrate and control access to a limited shared resource, the packet buffer pool. They are also responsible for emptying the buffers as fast as possible by matching idle ports with the buffered packets which need to be routed through those ports. The Packet Buffer Controller also determines when messages are stagnating in the buffers. When this occurs, an alternate routing mechanism which dynamically re-routes packets via alternative links is invoked. The buffer controllers also contain the deadlock avoidance mechanism.

Message Control

Message status and control information required by the Post Office is contained in the message header, while the message body contains the message data and any header information required by the processor or operating system. The message header contains the delivery mechanism to be used, a message identifier, the message size, and the destination of the message. The original message header created by the sending Post Office is augmented to include the number of packets that must be formed to deliver the message, which packet in that series the current packet corresponds to, and a checksum.

There are a number of options available for message instructions and delivery mechanics.

1. Messages can be delivered according to the Post Office's internal hardware routing function when the address of the destination processor is provided. The implementation of the routing function will depend on the interconnection topology.
2. An AXIAL OFFSET on a set of axes can be given. The order in which the axes are taken is irrelevant. This mechanism can be used to achieve a limited radius broadcast, hardware diagnostics, and status evaluation by serial signature codes.

The two routing methods can be used in conjunction with any of the following instructions:

- MESSAGE-SEND instructions will deliver the message to the destination processor's local memory. The processor will be notified of the arrival and becomes responsible for interpreting the message. This is the standard delivery mode.
- PEEK messages read a block of data in a remote processor's memory and return the value to the message creator. The remote Post Office directly reads the memory block and builds the return message, unaided by the processor. This instruction can be useful for debugging. Reading remote processor load and status information can also aid in dynamic program reallocation.
- The REMOTE-MEMORY-WRITE instructions will write a block of data into an area of the remote processor's memory. This instruction is entirely executed by the receiving Post

Office. The remote processor is not informed of the operation. This instruction is useful for initialization, and has potentially dangerous side effects if improperly used.

- ESCAPE instructions extend the instruction field and are used for Post Office initialization, diagnostics, and debugging.

When the packets are transmitted between two adjacent Post Offices, the receiver may either accept or reject the packet depending on buffer availability, deadlock avoidance, and the correctness of the checksum. If the message is accepted, the sending Post Office then frees the buffer space used by the delivered message.

When a message arrives at its destination, the message header is discarded and the message body is placed in the receiving processor's local memory. The processor is then given an interrupt so that it can process the newly arrived message.

Packetization Control

Large messages are decomposed by being broken into blocks that are the size of the data area of Post Office packets. A message header for multi-packet messages is constructed. The message data blocks and header are placed into a series of packets with ordering information added. A unique message identifier is included in the header so that many large messages can be reconstructed concurrently in a single Post Office node.

When decomposing a single message into several packets, the PE port must have some local memory that saves the status of the packetization process of the current message. If packet buffer space fills up, it will not be possible to packetize and load the entire message at once. The PE port will then need to transfer inbound messages into the processor's local memory and wait for buffer space to free up before continuing.

Multiple packet messages use an *assembly table* to hold information concerning message recomposition. When a multiple packet message arrives, one of three situations can arise:

- The packet is the first of the message to arrive at the destination Post Office. A new position in the packetization table will be allocated using the unique message ID and packet number. A message frame of sufficient size will be allocated in the *message staging area* of the processor's local memory. The base address of the message frame will be saved in the packetization table and the packet body will be indexed into the appropriate location in that frame. Order of packet arrival at the destination Post Office is not important.
- The current packet is an intermediate packet in the message. The packet number is added to the packetization table, and the message body is placed in the appropriate location in the message frame.
- The final packet arrives. The packet body is indexed into the staging area to complete the message. The packetization table entry is then deallocated and the processor is informed of the message arrival.

When a single packet contains the entire message, no assembly information needs to be stored. Delivery of the message then consists of the union of the three situations described above with assembly table accesses removed.

Packet Buffering

All ports in a Post Office share the same buffer pool from which packets are retrieved for transmission and stored on arrival. This buffer consolidation has the advantage of minimizing the number of internal busses, wires and controllers. Fewer total packet buffers are also required to achieve efficient packet delivery because the number of buffers allocated to the different ports at run-time is flexible. However, this also results in potential problems of buffer contention since they are a shared resource.

The critical resource in a distributed message passing system is the topologically constrained communication links. If all of the communication wires can be effectively used, the bandwidth of the network can be maximized. This implies that the bandwidth between the ports and packet buffers should not be the limiting factor, rather the availability of communication links should limit the performance of the Post Office. When this occurs maximum bandwidth of the communication component is achievable as the ports will continuously be driving messages across the network. The speed of the Post Office will then be directly proportional to the rate it can transfer messages between nodes. It is therefore essential to limit the overhead of the inherently serial packet buffer accesses in this model, and the linking of waiting messages in the buffers with free ports.

Virtual cut-through is an attractive option because some packets can completely avoid accessing the buffers. This eliminates buffering contention and overhead for some packets, allowing them to be delivered with less delay. Cut-through also reduces the average number of buffer accesses, which reduces the possibility of buffer contention between those packets that must be buffered.

All communication between ports and the buffers takes place on a Packet Bus[8] as shown in Figure 2. The Packet Bus is very wide giving it a bandwidth comparable to the sum of the bandwidth of all the ports. The ports can buffer a full packet allowing transmission across communication links to proceed even when buffer access is temporarily delayed. When a port has collected up to $n\%$ of a packet it will attempt to store that portion in the packet buffers. Storage of the entire packet will require $\lceil \frac{100}{n} \rceil$ buffer accesses. The ports will incrementally store the packet fractions immediately as they arrive. Choosing the correct value for n gives the ports freedom from probabilistic or deterministic ordering of transmissions and also utilizes the port bandwidth efficiently[7].

Although the packet buffers are physically shared, the two buffer controllers use the memory in such distinct ways that the packet memory can be viewed as two logically separate buffer pools.

The PE Buffer Controller queues up inbound messages to be delivered to the processor's local memory in a FIFO manner. This reduces the disorder of message arrival.

The Packet Buffer Controller accesses the buffer pool in an unordered fashion. This controller attempts to reduce the number of packets in the buffers by continually scanning the buffers searching for deliverable packets. As the controller peruses a packet from the buffer pool, the intersection of the set of currently idle ports and the set of viable ports through which the packet should be delivered is taken. When the resultant set is not empty, the packet is delivered out of any of the ports in this set. This permits the transmission of any of the packets in the buffer pool capable of being delivered regardless of the order of arrival. Packets are only deleted from the buffers when they have been successfully delivered through a port.

Dynamic Routing

Routing hardware included in the Post Office uses a *port list* of viable links through which the message could be delivered. This list is produced from the message header and topology dependent information that can be loaded at initialization time.

The routing algorithm should not require global information or maps to route messages. Such information implies that tables are required, and the size of these tables will be dictated by the number of processors and communications links, which is a non-scalable base. The possibility of using an algorithmic routing function that scales with the size of the hexagonal processor array of the FAIM-1 topology was investigated. A scalable algorithm was developed for our homogeneous architecture which is provably minimal (giving "shortest distance" paths), relies entirely on local information, and is extremely fast[14].

All routing in the Post Office is done dynamically. This is due to the fact that it is possible for a packet to take multiple paths to its destination in richly connected networks. A packet may be routed through any one of a set of ports and if any of these ports is free, the message will be sent through one of the free ports. Dynamic packet delivery compares favorably to FIFO control or algorithms that remove the ability to choose between alternate paths since:

- It is difficult to predetermine link contention and buffer availability in remote nodes at packet reception time without global information. These factors imply that the shortest queue is not necessarily the best or quickest path to the destination. Simulation has shown that it is much more efficient to allow packets to take the first available route rather than to determine at the time of message buffering which link the message will take.
- In a FIFO model it would be difficult to queue up a packet for several destinations and delete all redundant copies when the first packet is successfully delivered through one of the ports.
- The inherent ordering induced by FIFO causes difficulty in dynamically changing messages from one queue to another. If packets can only be removed from the head of one FIFO and placed at the tail of another, dynamically re-routing packets may not result in decreased latency unless the link has experienced a hard failure.
- When congestion occurs on a link or in the remote node, all of the packets in that queue will be delayed. When re-routing is used, an entire set of packets in a single FIFO could "time-out" one after another and need to be re-routed, possibly resulting in congestion on other links.

The Post Office's routing naturally reacts to congestion as messages with multiple paths will automatically begin to preferentially take the more lightly loaded links. Packets without alternate paths will quickly be identified and re-routed. However, dynamic routing is predisposed to deadlock, and must be coupled with a deadlock avoidance mechanism.

Packet Re-Routing

Network failure and severely degraded performance caused by link congestion or failure can be greatly reduced by dynamically re-routing messages. We are experimenting with a novel method of determining whether a packet is to be dynamically re-routed. This algorithm appears to be superior to a time-out scheme because the decisions to re-route packets are determined on a local load basis rather than by a global clock or fixed delay methodology.

The Packet Buffer Controller monitors the staleness of packets in the buffers by incrementing a "stale count" variable that is associated with each packet every time it cannot successfully deliver the packet. After the variable is incremented, it is checked to determine if the number of unsuccessful delivery attempts exceeds a critical value. When this value is exceeded the message is re-routed to a currently free link. If the message has not reached the critical staleness, the buffer controller continues scanning the buffers attempting to deliver other packets.

Some possible approaches for dynamically re-routing packets can utilize either a second best set of paths for the packet, a heuristic approach that analyzes the current situation to produce a path that will likely improve the location of the packet, or random re-routing. The FAIM-1 Post Office uses a combination of the heuristic and random approaches.

When packets are re-routed using the re-routing algorithm, it is possible to induce thrashing. This occurs when a packet oscillates between a set of adjacent Post Offices. This thrashing is reduced by recording the most recent port through which the packet passed, never delivering the packet back through that port.

The locality of this re-routing approach can be demonstrated by two examples. Assume that several ports are all receiving packets which are destined for the same outgoing port of the Post Office. This situation exists when the total bandwidth of the producers is much greater than that of the consumer. This will place the Post Office in a state where there will be a large number of packets waiting to be delivered through a single port. Even though the overall load on the Post Office is light (only one port is being heavily used), there is substantial delay induced because all messages are bottlenecked for delivery through one port. Since the bandwidth utilization of the Post Office is small and there are many free links, it is advantageous to quickly re-route messages through second best ports rather than requiring all packets to be delivered through the same overloaded port.

A short enough time-out delay for this case would be much too short in general circumstances, and may induce thrashing. Using the local algorithm described, the list of messages destined for the bottlenecked port will quickly be scanned and reach the critical staleness necessary to re-route some of the messages and disperse the load.

The opposing case occurs when there is heavy congestion that covers a number of adjacent nodes. The packet buffers will soon fill up with many messages destined for all of the six links. The more heavily used links will in general be more difficult to obtain for message traffic and thus will increase the probability that packets with alternate paths will be assigned to links where the traffic is sparser. In this case the delay before messages need to be re-routed should be larger because of the relatively high utilization of all links, the large number of messages which may be waiting for free ports, and the fact that any alternate route is also heavily loaded. A more even distribution of load will be treated similarly to a lightly loaded network where the average message latency is simply increased. The large number of evenly distributed messages in the buffers will result in an increase in the absolute time it takes for a message to "time-out" and be re-routed.

Deadlock

Deadlock may occur in a communication topology where messages need exclusive access to a resource (the packet buffers or communication channels), reliability requires that the resources cannot be preempted, resources are not reserved ahead of time, and there are closed loops in the topology. Deadlock will occur when all of the packets in a closed loop of nodes need to be placed in the buffers of an adjacent Post Office, and all of the packet buffers in the neighboring nodes are full. To prevent deadlock it must be assured that packets can circulate through the two port types and eventually be delivered to the destination processor which consumes the packets. It is assumed for this analysis that the processor's local memory is unbounded.

There are three general traffic patterns, two of which are subject to deadlock. Message traffic consists of:

1. OUTBOUND packets travel from the local processor's memory to the Post Office's packet buffers.
2. INBOUND packets move from the Post Office's buffers to the processor's local memory.
3. With NETWORK traffic, packets travel between Post Office nodes.

These messages neither augment nor diminish the number of packets in the network as a whole. Outbound packets are formed from new messages, and are loaded into the network buffer pool from the processor's memory, increasing the number of packets in the buffers. Inbound packets have arrived at their destination and are deleted from the Post Office's buffer pool after being loaded into the processor's memory, reducing the total number of outstanding packets in the network. Network packets will be taken from the current global buffer pool, routed through packet ports and stored in an adjacent Post Office's packet buffer. Deadlock can occur with outbound and network traffic.

There is no deadlock detection or recovery mechanism required in the Post Office, alternatively a prevention mechanism is employed. Sufficient free buffer space is reserved to allow messages to disseminate across the network. When the communication topology becomes heavily loaded, outbound message traffic is blocked until the congestion dissipates. Outbound traffic is never allowed to occupy the last free buffer. With free buffers, network packets can move to adjacent nodes. Ultimately packets will arrive at their destination Post Office and be consumed.

It is possible using the dynamic routing algorithm for packets to become undeliverable across their preferred links. However, so long as there is a free buffer in an adjacent Post Office, the rerouter can redirect a packet across a secondary link, which may allow other packets to be delivered along the preferred path.

It is our conjecture that one reserved buffer for network and inbound traffic is sufficient to guarantee deadlock free execution in the Post Office. This will on the average leave one buffer per Post Office free for network message traffic while no more packets are added to the buffer pool. However, the worst case scenario degrades to the point where there is only one free buffer per closed loop. In such a condition, one packet at a time will move, with the implementation assuring that there is a high probability that the transmitted packet will move closer to its destination. The time to resolve such congestion will be very large. Nevertheless, the likelihood of such a scenario is extremely rare. Depending on the topology used, the length of the closed loops could be very large compared to the diameter of the network. Furthermore, a routing algorithm rich in alternate paths will allow many packets to not be constrained to a single loop further reducing the possibility of a loaded network demonstrating pathologic latency.

If more buffers are reserved, outbound messages can be blocked before the network load becomes great enough to degrade performance. Packets can flow along their preferred paths and use a substantial fraction of the raw bandwidth even in the regions of heaviest congestion since nearly all neighbors will contain free buffer space. Extra reserved buffers also allows the network to disperse a local congestion more efficiently so that it doesn't begin to adversely affect global traffic[10].

Congestion is relieved when network packets become inbound packets. It is possible for a packet to arrive at its destination yet be rejected because all of the buffers are being used for network packets. Specializing the reserved buffers can avoid this. The algorithm presented uses this technique. Although this algorithm is customized for the FAIM-1 topology, it is sufficiently general for use in other topologies. Four or more reserved packet buffers are necessary. Three buffers are reserved for network traffic and one buffer slot for inbound packets to be consumed by the processor. The algorithm works as follows:

- The buffers may be used by any port for traffic in any direction until there are only four free buffers left.
- When there are less than or equal to four free packet buffers available, packets arriving on the ports must be tested to assure that they will not cause deadlock by filling up the buffer space with messages that may not be forwardable. All possibly deadlock causing messages will then be rejected according to the following criterion:
  - The PE port will no longer add outbound messages to the buffer pool, and will only relieve congestion by removing packets from the network through delivering inbound packets into the processor's local memory. This is equivalent to shutting off message producers and increasing the priority of message consumers in the network. This may have a side effect of influencing scheduling in the processors.
  - Packets that arrive at their final destination and become inbound packets will only be accepted when there are less than two inbound packets in the buffers.
  - Network packets are accepted when there is an available buffer while always allowing one free buffer for inbound packets.
- When a network packet arrives at its destination and becomes an inbound packet, it will subsequently be transmitted to the processor's memory, freeing up one buffer space. When buffers are freed up such that there are more than four free buffers, once again all packets are accepted.

This mechanism has the advantage of locality because each individual Post Office node determines what action to take depending on the congestion in its own buffer pool. No special cases need be considered for the different types of port congestion and failure, allowing the mechanism to be both rigorous and simple. Simulation also shows that reserving a larger number of buffers permits the throughput to remain high as the frequency of message generation is increased due to the large portion of the bandwidth that remains available in loaded conditions.

Conclusions

Direct support for communications in many of today's multiprocessor systems is necessary to make general purpose multiprocessing an attractive option. However, this demands a significant shift the transistor budget of these machines. The Post Office is an experimental communications accelerator that provides our architecture with hardware that is completely responsible for the physical delivery of messages. By operating concurrently with the program execution hardware and allowing concurrent operation of the individual ports, the performance of the Post Office is sufficient to work well even as networks are scaled up. The Post Office for the FAIM-1 machine has been rigorously simulated and is currently being implemented in custom CMOS VLSI technology.

Acknowledgments

The work presented here is not solely the work of the authors. Key contributions have also been made by Barak Pearlmutter, Judy Anderson, Dick Lyon, and Shimon Cohen.

References

[1] A. L. Davis and S. V Robison. *The Architecture of the FAIM-1 Symbolic Multiprocessing System.* Proceedings of the Ninth International Joint Conference on Artificial Intelligence, August 1985, pp. 32-38.

[2] A. Goldberg and D. Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley Publishing Company, Reading, Mass., May 1983.

[3] D. Gordon, I. Koren and G. M. Silberman. *Fault-Tolerance in VLSI Hexagonal Arrays.* Preprint.

[4] R. Gurwitz. *The Butterfly Multiprocessor.* Talk presented at the 1984 ACM National Convention, San Francisco, October, 1984.

[5] INMOS Limited. *IMS T424 Transputer Reference Manual.* INMOS Limited, 1984.

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

[7] W. E. Kluge and K. Lautenbauch. *The Orderly Resolution of Memory Access Conflicts Among Competing Channel Processes.* IEEE Transactions on Computers, Vol C-31, No. 3, March 1982, pp. 194-207.

[8] R. F. Lyon. *Single Instruction Stream, Multiple Data Stream, Multiple Port Pipelined Processor.* Schlumberger Internal Report, September 1985.

[9] A. J. Martin. *The Torus: An Exercise in Constructing a Surface.* Proceedings of the Second Caltech Conference on VLSI, 1981, pp. 527-538.

[10] G. F. Pfister, V. A. Norton. *"Hot Spot" Contention and Combining in Multistage Interconnection Networks.* Proc. '85 Int'l Conf. on Parallel Processing. August, 1985, pp. 790-797.

[11] C. L. Seitz. *Experiments with VLSI ensemble machines.* J. VLSI Comput. Syst., Vol. 1, No. 3, 1984.

[12] C. L. Seitz. *The Cosmic Cube.* Communications of the ACM, Vol. 28, No. 1, January 1985, pp. 22-33.

[13] D. P. Siewiorek, R. S. Swarz. *The Theory and Practice of Reliable System Design.* Digital Press, Bedford, Mass., 1982.

[14] K. S. Stevens. *The Communications Framework For A Distributed Ensemble Architecture.* Tech. report #47, Schlumberger Palo Alto Research, February, 1986.

[15] S. J. Stolfo, *et al. Architecture and Applications of DADO, A Large-Scale Parallel Computer for Artificial Intelligence.* Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August, 1983, pp. 850-854.