The Express Broadcast Network: A Network for Low-Latency Broadcast of Control Messages

Kevin Bolding and William Yost*
Dept. of Computer Science and Engineering
University of Washington
Seattle, Washington, USA 98195-2350

Abstract

We present the Express Broadcast Network (EBN), a network used for quick and reliable broadcast of control messages in multicomputer networks. The EBN can be implemented with a single extra wire per network link and with minimal extra hardware at each routing node. However, it provides very fast broadcast mechanisms that take advantage of all redundancy in the network to deliver messages regardless of faulty network components. We present extensions of the basic network to include multiple-wire, multiple-bit, and bidirectional wire support, as well as describing basic methods of using the EBN for various applications.

1 Introduction

Massively parallel computers, connecting possibly thousands of processing nodes into one large computer, rely heavily on large data networks which connect the processors. Typically the data networks have very high bandwidth and relatively low latency in order to efficiently deliver messages between processing nodes. The network design and associated routing algorithms tend to be optimized to provide the best service for the passing of data between processors. Unfortunately, networks optimized for data transfer do not necessarily provide good service for control messages, which have different network needs.

Data messages need networks that provide high-bandwidth, reasonably low latency, and reasonably reliable delivery for point-to-point communication. Routing algorithms designed for moving data usually send a single copy of a message through a single (possibly dynamic) network path. Failures in the network may cause loss of communication between nodes and may result in loss of data until the faults are detected and the operating system reconfigures the network.

On the other hand, control messages require little bandwidth, but need very low latency and extremely reliable delivery for broadcast (one-to-all or all-to-all) communication. Control messages may be used to signal network barriers, changes in network configuration, starting or stopping of computation, or warning of network errors. They should be able to reach all connected nodes in a network in a timely manner regardless of faulty links. When control messages are sent across a network optimized for data transmission, they are treated in the same manner as all other messages. This means that they are likely to experience large delays in times of high congestion and they are subject to being lost by the network due to component failure. Designing a network to give priority to control messages over data messages helps alleviate the problem of congestion, but does not help when components fail. Also, most data networks do not directly support broadcast, so the broadcast of system control messages must be accomplished by a multitude of single-destination messages.

In order to provide efficient transmission of control messages, a separate network, either physical or virtual, is needed. A small number of currently produced parallel machines have multiple networks for different functions. The Thinking Machines CM-5 computer includes three distinct physical networks: a data network, a control network, and a diagnostic network [4]. The Fujitsu AP1000 also has three networks, one for data, one for control broadcasting, and one for synchronization [3]. However, the additional hardware needed to build three separate high-bandwidth networks

*This work is supported in part by US Office of Naval Research grant N00014–91–J–1007 and US National Science Foundation grant MIP9213469.

is very costly. IBM's Vulcan architecture uses a virtual broadcast network built on top of the data network's links to implement its fault-tolerance mechanisms [5]. Unfortunately, adding a virtual network on top of the existing network complicates the critical path of the network routers. Thus, a low-cost, but physically separate control network is needed for a large-scale machine. To this end, we present the Express Broadcast Network, an inexpensive low-bandwidth, low-latency network which is extremely redundant and optimized for the broadcast of system-wide control and synchronization signals.

2 The Express Broadcast Network

The Express Broadcast Network (EBN) is designed to meet the critical requirements of a network to transmit system control messages, while costing very little to implement. The features we intend to support are as follows:

- Low latency
- Broadcast ability
- Very high reliability
- Performance independent of the data network load
- Scalability
- Reliability and performance nearly independent of failures in the network
- Low implementation cost

In order to achieve these features at a low cost, we restrict the network as follows:

- Very low bandwidth
- Synchronous operation[1]
- Broadcast-only operation – no single source/ single destination operation

2.1 Description

The primary function of the EBN is to broadcast messages across the network. A typical implementation of a broadcast network utilizes a common bus across the entire network or a subset of the network. While the physical configuration is simple and best case performance is near optimal, it requires a complicated protocol to resolve contention and performance can drop sharply as the load grows. In addition, the common bus leaves the system vulnerable to faults. For these reasons, we choose to implement the EBN as a set of point to point links. This creates natural fault tolerance and reliability through redundancy and greatly simplifies channel protocol while maintaining good performance under all reasonable situations and acceptable worst case performance.

A simple method of implementation is to build a static tree structure into the network, where each node has in-degree one and out-degree of greater than one. A spanning tree could then be defined and each node would act simply as a repeater, copying its input message onto each of its outputs. Two limitations to this design limit its usefulness: each link is a single point of failure, and the tree has only one statically-determined source node. A failure on a link causes loss of communication to all of its descendants. Furthermore, the only node that can broadcast in this tree is the single source node. Cray Research's T3D uses a special network to implement barrier synchronization and broadcast primitives using static trees [2]. This allows multiple source nodes, but still suffers from single-point failures. To avoid these problems, the EBN incorporates a dynamic redundant tree structure.

The EBN tree structure is dynamic in that any node may become the root of the broadcast tree at will. This is accomplished by requiring that all nodes in the network maintain a "listening" mode, checking the incoming links for messages. When a node desires to broadcast a message to the network, it simply sends a message to each of its neighbors and internally remembers that it has generated a broadcast message. Each of the receiving nodes then marks itself as having received a message and copies the message to each of its remaining neighbors. A node which has already received a message ignores any subsequent messages. In this way, all nodes in the network receive the broadcast message. Moreover, the broadcast mechanism will automatically take advantage of any redundancy in the network, expanding around any number of faulty links to eventually reach all connected nodes in the network. This is shown in Figure 1, where the broadcast reaches the entire network despite the presence of faults.

The EBN has, in its simplest form, two basic communication primitives: *eurekas* and *barriers*. Eurekas [2] are one-to-all broadcast messages and are used when the entire network is waiting for a single event. Barriers are used when the network is waiting for all nodes to experience a common event. Barriers can be implemented by hav-

[1] It is not necessary that a global clock exists, but each node must be able to consistently count the passage of time.

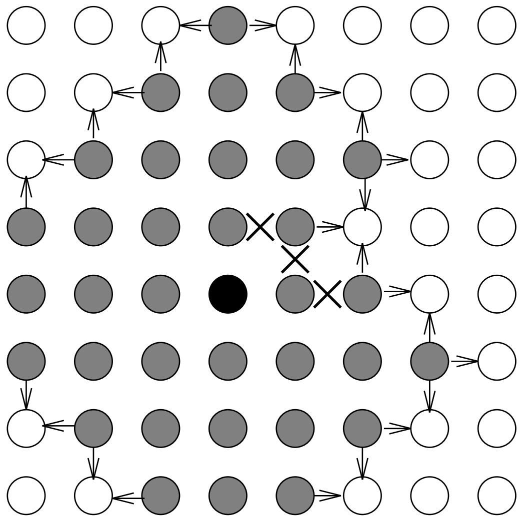

Figure 1: A broadcast using the EBN. A message originates from the black node, which propagates the message to all of its neighbors. This illustration is shown after the broadcast has spread four hops. As the broadcast message wavefront spreads, it bypasses faulty links (marked with X's) to reach the entire network.

ing all nodes broadcast eureka messages if they have *not* experienced the event. When no messages are broadcast, the barrier is complete. Synchronization details for these are presented in Section 4.1.2.

3 Applications

The Express Broadcast Network has a wide variety of potential applications. It was originally designed to provide broadcast of emergency signals in the Chaos Fault Tolerant Router, so we first present that application as a short case study. We also briefly describe how it could be used in system startup, fast barriers, global flow control, and general control for networks.

3.1 Chaos Fault Tolerant Router

In the Chaos Fault Tolerant Router [1, 6], the EBN is utilized as the means by which the error handling procedures in the network are initiated and synchronized. Figure 2 shows the state transition diagram with the various transition conditions marked. In order to maintain synchronization of the EBN, there is a *Network Delay* period after each network broadcast to allow the broadcast message to propagate to the entire network. Sections 4.1.2 and 4.5 give more details on this.

The Chaos Fault Tolerant Router utilizes various operational modes of the EBN. In the *Normal Operation* state, any router may signal a possible error condition by broadcasting, via the EBN, an *error-detected* eureka signal. This is similar to the function of the *global faulting* mechanism in the Vulcan architecture [5]. At this point the routers enter the *System Drain* state, and all the nodes cease injection of new packets into the data network in order to drain the system. Completion of the system drain is detected by a barrier waiting for all nodes to drain. Once the barrier is complete, the routers advance to the *Decision* state. If the system drain has not completed by a prespecified time, it is assumed that there is a faulty component in the network that prevents successful delivery of some packets, and all remaining packets will be dropped, emptying the network. After all packets are dropped, the *Decision* state is entered. In this state, a decision on whether to run the diagnostics is made. Once again, the EBN is used in the eureka mode: any node that deems it necessary to run the diagnostics broadcasts to the network. If there are any affirmative responses, the network runs the diagnostics and, under operating system control, reconfigures the system, returns the EBN to the idle state and restarts normal routing operation.

3.2 System Startup

When a parallel computer system is initiated, one of the necessary functions involves the dissemination of configuration and partitioning control in-

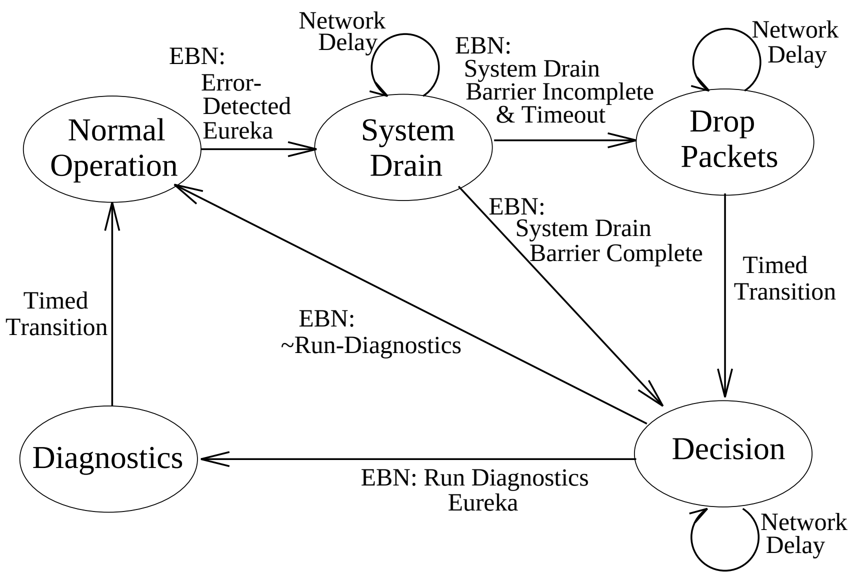

Figure 2: The Fault Tolerant Chaos Router state transition table. The role of the EBN in its fault management procedures is illustrated.

formation by a master node. This is a situation in which the one-to-all broadcast mode could greatly simplify the problem. A master node can simply broadcast the configuration information to all nodes. This requires the use of one of the multi-bit transmission protocols described in Sections 4.2 and 4.3.

3.3 Global Flow Control

The Chaos Fault Tolerant Router uses the EBN to institute a crude form of flow control barrier in the form of the system drain procedure. One reason for the initiation of a drain is the detection of a missing packet. The packet may simply be delayed by network traffic and the drain will resolve the problem. The EBN could be used to implement a more sophisticated barrier. A node missing a packet broadcasts a message on the EBN which inhibits new message injection. The node then signals again upon delivery of the packet, releasing the inhibition on injection. This would lower the drain overhead by avoiding the need for a full system drain.

Other flow-control functions are also compatible with the EBN. The network could be designed such that at some arbitrary load level, a flow-control barrier could be initiated. In a non-minimal router, the need to deroute could be a trigger. The flow-control barrier function is largely independent of the EBN which would simply provide a rapid and efficient means of implementation. Rapid setup of barriers using EBN broadcast allows finer-grained control than is possible with slower data network broadcasts.

3.4 Application Barriers and Eurekas

The barrier and eureka functions of the EBN could be made available to software applications in order to expedite their function. Software barriers can be used to provide many useful functions involving synchronization. Eurekas can be used when parallel processes are attempting to solve the same problem using different mechanisms: the first process to complete signals all of the others by using a eureka signal.

3.5 Control

The selection of a master node for control of certain functions, such as the earlier example regarding system startup, could be simplified with use of the EBN. Each node is assigned a unique priority level. When the network moves to a state which requires a master node, each router listens for another router to ping on the EBN to announce that it has taken mastership. A fixed period is be designated during which the highest priority router pings on the EBN if it is present and active. If the period passes without a ping, then a new period commences during which the next highest priority node pings. This continues until an active node is found to assume mastership. The fault tolerant nature of the EBN ensures that all connected nodes receive the highest priority ping, electing a master in a consistent manner. If the network has become separated by faults, each separate connected block of components elects its own master by this algorithm.

4 Implementation

The EBN, consistent with the low amount of bandwidth needed for control messages and the requirement that the implementation costs be kept low, can be implemented using only a single wire per link. While space (additional wires) and time (time encoded messages) can be balanced to provide greater bandwidth and a greater variety of messages, the single wire implementation can provide full EBN functionality. We first describe an implementation utilizing a pair of unidirectional wires on each link which allows only single-bit messages, and then describe extensions to multiple-wire, multiple-bit, and bidirectional wire implementations.

4.1 Basic Network Implementation

The EBN is constructed using broadcast nodes to join links. These nodes may be connected together in *any* shape that maintains connectivity of the network, regardless of the shape of the independent data network. The connectivity of the EBN may be increased over the data network in order to minimize broadcast latency, or reduced in order to cut costs. However, for the sake of ease in physical layout, the network configuration of the EBN is envisioned to be the same as that of the data network so that the EBN wires can be bundled with the data network wires. Moreover, because the EBN requires only a small amount of logic, it can easily be added, at design time, to the data network routing chips already required in a parallel machine.

4.1.1 Network

For the basic implementation, we assume that the nodes are connected with unidirectional wires (single source/single destination). To establish bidirectional communications between two nodes, two separate wires must be used. A set of unidirectional wires connecting each node and its associated processor is also necessary. Depending upon the desired functionality, a small number of control lines on the processor link may also be needed.

Once the network is set up, the EBN's functions are very simple. At reset, each node enters an inactive state. Each node "listens" to each incoming link. Whenever any input link becomes active, the node switches to the *active* state. In this state, the node re-broadcasts the incoming message to all of its neighbors by driving all outgoing links, including the processor link, to the active value. Thus, every processor that is connected by the network is notified of the active state.

Since the work required at each EBN node is limited, a message in the EBN should propagate through each node with minimal delay. A pipelined design might implement this as two cycles: one for transmission across the wire and one for the broadcast. The broadcast delay, the time for any node to broadcast a message to the entire network, is limited by the product of the node delay and the diameter of the network.

The broadcast delay across the network introduces a potential problem in that during the propagation of one broadcast message, another source may initiate a second broadcast. Requiring that no node can initiate a broadcast when it is already in the active state due to a previous broadcast reduces the problem. However, the second source may have initiated its broadcast before the first message had propagated to it. Fortunately, with single-bit messages, this does not present a problem because all messages at a given time are interpreted the same way, regardless of their source node. The only necessary addition is a method of maintaining network synchronization so that messages will be interpreted by all nodes in the same manner.

4.1.2 Synchronization

Because the single-wire EBN can only broadcast a single bit message, every node in the network must remain in at least loose synchronization in order to agree on the interpretation of a message on the EBN. The EBN itself is used to keep the machine synchronized. Each EBN node keeps track of the state of the network according to a predetermined protocol. Because the propagation of a eureka broadcast message is not instantaneous, synchronization is maintained by allowing time for the network to settle between messages. For example, the network may start with each node in the idle state. When a node initiates a broadcast to the network, each node switches to an active state as it receives the message. Every node then executes a delay of length greater than or equal to the worst-case delay across the network to ensure that every node has received the broadcast and made the state transition before any new transition is initiated. During this time, any action indicated by the message is performed. After the delay expires, the node, depending upon the protocol and the presence or absence of a new synchronizing signal on the EBN, either returns to the idle state and executes another delay in order to ensure that all nodes are idle or makes a transition to another active state. Ultimately, the network is restored to its original all-idle state, and normal operations resume.

Barrier messages require slightly more complex synchronization. A barrier is constructed as follows: a eureka message is used to synchronize all of the nodes, as described above. After a synchronization delay, each node simultaneously broadcasts a eureka message if it has not satisfied the criteria for entering the barrier. Since a broadcast from *any* source reaches all nodes, all of the nodes in the network are notified if any node has not yet entered the barrier. If no responses are received, the barrier is considered complete and processing can resume.

While the router will control the EBN along with the data network routing process, the node intelligence will still be located within the processing node. The EBN's function is limited to providing communication services, and maintaining synchronization of the network. All other control based on EBN communication will be performed by the processing node by means of communication with the EBN router via dedicated control lines. Thus, the EBN does not create messages; all messages must be created either by software in the processing node or by controller hardware outside of the EBN.

4.2 Multiple-wire Channels

Using single-wire channels limits messages to single-bit messages. While messages may have different meanings depending upon the state of the network at the time that the message is sent, it may be necessary to send different messages during the same network state. This can be achieved by using multiple-wire channels. The number of distinct messages that can be sent in the same state is the number of distinct values that can be sent on the wires, w for $\log w$ wires.

However, the extension to multiple wires introduces a new problem. A simplifying assumption used in the design of the single-wire EBN is that the meaning of a message is independent of its source: all messages mean the same thing. That is no longer the case since two sources could simultaneously be broadcasting different messages in the multi-wire scheme. Prioritizing the messages solves this problem. A strict ordering of all possible messages must be made, so that a message is given strict precedence over all messages of lower priority. Then, whenever a node receives and forwards a message, it continues to listen for messages of higher priority arriving on any of its incoming links. If a message arrives at a node that has already received a message of lower priority, the new message will be broadcast by the node just as if the first message had never been received. All nodes, including the sources of lower-priority messages that were pre-empted will receive the highest priority message. These sources can save their original messages for re-broadcast at a later time. One problem remains: higher priority messages may preempt others at any time, making the time for the broadcast of a message very large. This can be fixed by constraining the generation of new messages. A node may not create any new messages, of any priority, after it has received a broadcast message from another node. This reduces the worst-case time for the completion of a broadcast to proportional to twice the diameter of the network since the node farthest away from a broadcasting node may source a higher priority broadcast just before the original broadcast reaches it. Each of these broadcasts takes time proportional to the diameter of the network to complete, so the total operation is proportional to twice the diameter. Thus, each EBN node will have to maintain itself in "listening" mode for a long enough period to ensure that the current message will not be pre-empted before the node makes the transition to a new state.

4.3 Bit-serial Messages

Messages may be extended by time encoding the bits on each signal wire, an orthogonal process to extending the number of wires. The same effect is achieved via different implementations. Effectively, it is simply a tradeoff of time vs. space in order to achieve a greater space of possible messages on the EBN. Time encoding allows an extended serial broadcast in the one-to-all broadcast mode, while the equivalent extension in space, greatly extending the number of wires in the link, is clearly not feasible. Of course, a hybrid solution with multiple wires and multiple bit messages is also a legitimate extension of the EBN. One-to-all broadcast messages have a two phase structure. In the first phase, a node wishing to broadcast a one-to-all message broadcasts a setup message. Any collision of messages will be resolved through means of priority bits encoded into the setup message. At the end of phase one, a single node has established broadcast rights to the entire network, and all nodes are listening for that message. Each node will have marked as the parent direction the channel from which the broadcast request that ultimately prevailed was received. This sets up a pseudo-static tree for the ensuing one-to-all data broadcast. Any node that received more than one instance of the same broadcast signal on the same cycle may resolve the conflict arbitrarily in setting a parent direction. The second phase consists of broadcasting the data portion of the message to the entire network, which can be done without conflict. Finally, the source node relinquishes control and the network returns to its idle state.

The lack of control lines associated with the EBN makes it necessary that a multiple-bit message be either of fixed length or that there be a termination sequence to flag the end-of-message. Synchronization becomes complicated because of the possibility that messages of different priority will arrive at a router out of phase and on different channels. Any router may update the message that it propagates and the current parent direction if it receives a new, higher priority message, therefore it is possible for a router to be the recipient of an overlapping stream of new messages from which it must decide on one to propagate. It will resolve this situation by propagating any fully received message of higher priority than any message either previously propagated or currently waiting to be propagated. Messages received while the router is in the process of propagating another lower priority message will be buffered for the next open propagation slot. The buffer will be dynamic with only the highest priority message preserved while all others are dropped. In the presence of multiple messages, this protocol can result in a lot of frantic activity but with new message injection inhibited in any router that has received or injected a message into the network and with the damping of lower priority messages, the highest priority message will make monotonic progress across the network

4.4 Bidirectional Channels

By using bidirectional wires, which can be driven by either connected node[2], the number of wires needed to build the EBN can be halved. However, a protocol must be enforced in order to ensure that data is not lost by driving both ends of a wire concurrently. Since there is no explicit hardware on the network which arbitrates mastership of the bidirectional channel, the lines are driven by open collector outputs in the nodes along with passive pull-ups, avoiding the physical hazard of having two active drivers of opposite sense on the line. The logical hazard of two messages driven onto the same line is only a problem if the node is in a state in which a number of different messages are possible. The problem is easy to detect. A router will not initiate transmission across a link that is already in use, therefore, any collision that occurs will involve in-phase messages. A router that has a message to propagate that is of higher priority than the message it is currently in the process of receiving will wait until the message is fully received and then, if needed, will utilize the cycles that immediately follow to reply with its own message.

Because the EBN connections to the node are I/O ports, the node can monitor its own outgoing transmission. If the transmission does not match what was sent, then the node concludes that the other side is transmitting in synch and a collision is detected. While it is possible that the message from one side of the link will be uncorrupted, it is not possible for both sides to see uncorrupted messages from colliding messages. At the end of the transmission, there will be a dedicated cycle in which either side can pull down the line as a sign that there has been a collision. Both sides will see this signal and initiate a protocol that forces and alternates mastership to one end of the link at a time. This temporarily creates a virtual unidirectional link which allows for the resolution of link ownership conflicts.

The use of bidirectional channels is orthogonal to the use of multi-wire channels and multi-bit messages. Any combination can be utilized. Because the message collision problem arises only when there is a choice of messages that can be sent, it is not a problem when single-wire, single-bit networks and protocols are used.

[2]We use the term *bidirectional* to refer to half-duplex wires, which can be driven in only one direction at a time.

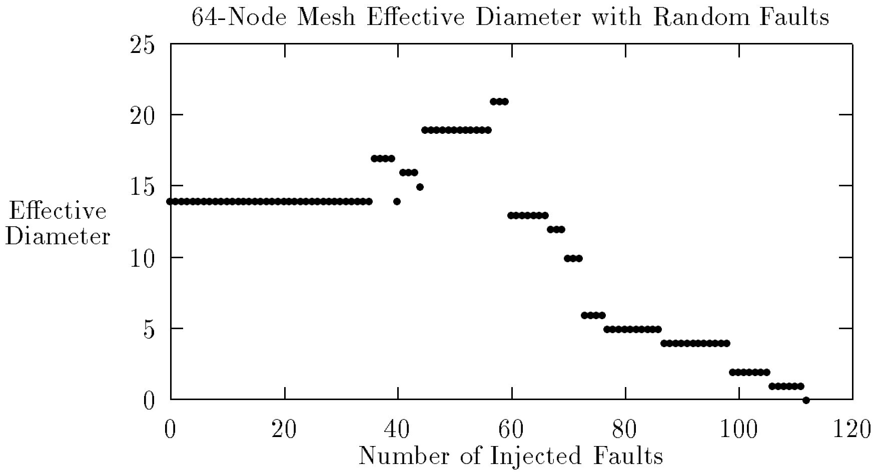

Figure 3: Network diameter vs. number of (randomly injected) faulty links for an 8×8 mesh network.

4.5 Performance

The time to complete a broadcast in the EBN is proportional to the diameter of the network, d, where d is defined as the longest minimal distance between any two nodes in the network, *taking faulty components into account.* For single-bit networks, broadcast is bounded by $d \cdot s$, where s is the delay incurred during each hop a message takes. The effect of multiple-bit messages is to multiply the broadcast time by a constant factor proportional to the number of bits in a message. The granularity introduced by the need to wait for the complete transmittal of the current message before a new, higher priority message can be propagated provides a constant cost term. Therefore, neither of these effects will cause more than a constant factor change in performance.

For any network, the worst possible value of d is N, the number of nodes in the network. However, most networks have a much smaller diameter. For instance, a n-dimensional mesh with no faulty components has a diameter $d = n\sqrt[n]{N}$. When components fail and are configured out of the network, the diameter may grow. However, because the EBN sends messages on all possible paths from source to destination, the diameter stays near the minimum for small numbers of faults.

To gauge the rate at which the diameter of a network grows as faults occur, we performed the following experiment: Randomly-placed link faults were injected into a two dimensional mesh network. After each new fault was injected, the network diameter was determined. We continued the process until all links in the mesh were faulted. The results show that the network diameter initially tends to increase as minimum length paths between nodes become blocked. However, as the number of faults increases, a point is reached where the next faults severs the path that determines the diameter and disconnects completely the two nodes connected by that path. The diameter of the two resulting subnetworks is less than the network diameter prior to the last injected fault. The results are not monotonic from that point and the diameter may increase of decrease with each new injected fault. Eventually, though, the diameter begins to shrink, dropping to a value of zero when all links are faulted and the network is totally disconnected. A plot of the network diameter vs. the number of injected faults for a typical experimental run is shown in Figure 3. The maximum network diameter achieved in this process for each random sequence of faults was then tabulated. The overall experimental results, shown in Table 1, indicate that the maximum network diameter for two-dimensional meshes under the condition of random faults is roughly proportional to $\sqrt{N}$. This is significant because of the need to delay state transitions for a period proportional to the network diameter. The required delay will therefore grow slowly relative to N. It appears that setting the delay for a network diameter of $5\sqrt{N}$ gives a sufficient worst case margin. The experiment was repeated with entire routers faulted instead of simply links. This is the equivalent of injecting four simultaneous link faults. The maximum network diameter achieved is 15% to 30% less than for link faults due to earlier disconnection of the network.

Consideration must be made of the potential

N	Mean Max Diameter $\overline{d_{max}}$	Std. Dev. σ	$\overline{d_{max}} + 3\sigma$	$\frac{\overline{d_{max}} + 3\sigma}{\sqrt{N}}$
16	8.9	1.5	13.4	3.4
64	21.8	3.5	32.2	4.0
100	28.9	4.7	43.0	4.3
144	35.8	5.4	52.1	4.3
196	42.8	6.7	63.0	4.5
256	49.7	7.0	70.8	4.4
400	61.9	8.4	87.0	4.3
576	76.5	11.9	112.3	4.7
784	92.8	14.9	137.6	4.9
1024	109.5	14.9	154.1	4.8

Table 1: Maximum diameter of 2-dimensional mesh networks with random faults. The mean maximum diameter, $\overline{d_{max}}$, and standard deviation, σ, are reported for 100 trials. The mean plus 3σ column indicates the number that bounds the actual maximum 99.8% of the time.

effects of correlated faults. As is shown by the above experiment in which nodes where faulted instead of links, simply applying faults in close proximity will not cause an increase in network diameter beyond that observed under random injection. Even if the faults are carefully selected, it takes a significant number to increase the diameter of the network to $5\sqrt{N}$. For a fault to cause the network diameter to increase, it has to force the minimum path between two nodes to backtrack (make negative progress). Intuitively, it would take more than $3\sqrt{N}$ *directed*, not random, faults to increase the network diameter from $2\sqrt{N}$ to $5\sqrt{N}$. Any problem that could generate such a set of correlated faults would have to be global, not local. The expectation for systems utilizing the EBN is that periodic maintenance and repairs would be performed and the fault rate will be such that large numbers of faults cannot accumulate.

4.6 Synchronous/Asynchronous Behavior

The EBN does not require that the end nodes of each link be synchronized, but synchronous behavior will simplify the protocol. It is necessary that each node have a means to count the passage of time. Because of the need to delay state transitions for a period proportional to the end-to-end delay across the network. In an asynchronous system, timed state transitions by the router state machine controller will need to be made more robust, probably through the use of a synchronizing signal on the EBN. Asynchronous behavior would require more complicated protocols on the links that would allow for internal synchronization of a received message by the EBN node. The difficulty of implementing an asynchronous protocol on bidirectional lines would probably limit the usefulness of that implementation option.

5 Future Work

While the network itself is inherently fault tolerant because of the multiple paths between nodes in most networks combined with the broadcast nature of signal propagation, there is a need by the links themselves to incorporate additional fault tolerance. The failure of an EBN signal, stuck at active, must not be allowed to imitate a legitimate broadcast message. There are a number of ways to protect the line integrity including the use of a heartbeat that would signify the health of a line or the addition of a complementary line (or a parity line if the network were to be implemented with more than one line). The tradeoffs need to be considered closely. The heartbeat adds considerable complexity to the protocol and could add considerable overhead cost to the propagation time of the broadcast message. The additional line(s) are simpler to implement but have a heavy overhead cost in pins and lines.

The addition of a low latency broadcast network to a data network could make fundamental changes possible in the way that the network is controlled. As described in the discussion on flow control, fine-grained global control is possible in situations in which only local control or very coarse high latency global control can be achieved using the data network. This characteristic could potentially be useful for problems such as deadlock avoidance or recovery. There is the potential for very positive tradeoffs in which active control replaces routing algorithm complexity.

6 Conclusions

Most current multicomputers use a single network to transmit all data and control messages. Unfortunately, the network is usually optimized for the needs of transmitting data from node to node, namely providing high-bandwidth for single-source, single-destination messages. Control messages have different requirements such as very low latency, broadcast ability, and very reliable delivery. Moreover, control messages rarely require very much network bandwidth.

We have designed the Express Broadcast Network with the needs of control messages in mind, while striving to keep the implementation costs minimal. The EBN can be implemented with only a single wire per link, and with minimal circuit support. However, it takes advantage of all redundancy available when sending broadcast messages, allowing very reliable delivery of control messages. We have developed extensions to allow for multiple-bit, multiple-wire, and bidirectional wire variations of the basic EBN links. Also, we have shown by example how the EBN's various modes of operation can be combined to provide simple control structures.

7 Acknowledgments

It was in discussions with Larry Snyder that the idea of the Express Broadcast Network was first born. Many of the concepts presented in this work were due in part to him as well.

References

[1] Kevin Bolding and William Yost. Design of a router for fault-tolerant networks. In *Proc. Parallel Computer Routing and Communication Workshop*, pages 226–240, May 1994.

[2] Cray Research. *Cray T3D System Architecture Overview Manual*, 1993.

[3] Hiroaki Ishihata et al. Third generation message passing computer AP1000. In *Proceedings of the Int. Symp. on Supercomputing*, 1991.

[4] Charles Leiserson et al. The network architecture of the connection machine CM-5. In *Proc. Symp. on Parallel Algorithms and Architectures*, pages 272–285, 1992.

[5] Craig B. Stunkel et al. Architecture and implementation of Vulcan. In *Proceedings of the Intl. Parallel Processing Symposium*, pages 268–274, 1994.

[6] William Yost. Cost effective fault tolerance for network routing. Master's thesis, University of Washington, 1995.