The Cray T3E Network:
Adaptive Routing in a High Performance 3D Torus

Steven L. Scott and Gregory M. Thorson
Cray Research, Inc.
{sls,gmt}@cray.com

Abstract

This paper describes the interconnection network used in the Cray T3E multiprocessor. The network is a bidirectional 3D torus with fully adaptive routing, optimized virtual channel assignments, integrated barrier synchronization support and considerable fault tolerance. The routers are built with LSI's 500K ASIC technology with custom transmitters/ receivers driving low-voltage differential signals at 375 MHz, for a link data payload capacity of approximately 500 MB/s.

1 Introduction and Overview

The T3E is the second in a line of scalable multiprocessors from Cray Research, following the T3D. Both machines use commodity microprocessors [5][8] surrounded by custom "shells", with access to a shared, distributed memory through an integrated interprocessor network.

The shell of the T3E has been substantially redesigned to increase pipelining in the global memory system; well over a kilobyte of memory requests can be concurrently outstanding from each node[17]. This greater latency tolerance allows the T3E network to concentrate primarily on sustainable bandwidth and scalability rather than on latency. This in turn has allowed a switch from ECL to CMOS standard cell for the network routers.

The 3D torus topology of the T3D has proven to be a strength, both for its scalability and its wiring efficiency [1]. The T3E retains this topology, but changes the processor/ network balance. Whereas in the T3D, two processors shared a (three-chip) network node, each processor in the T3E is its own network node (see Figure 1). This change, and a doubling of the link bandwidth, results in a factor of four increase in per-processor network bandwidth. The greater logic density also allows the T3E network to implement fully adaptive routing, provide fairly robust fault tolerance, and incorporate a set of virtual barrier synchronization networks.

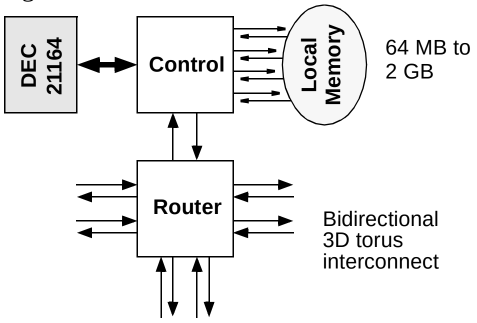

Figure 1. T3E PE block diagram

As the network grows in size, the aspect ratio of the largest to smallest dimension stays within a factor of two up to 1024 processors (8x16x8). Due to mechanical constraints, the X and Z dimensions then remain fixed at 8 and the Y dimension grows to 32 for the 2048-PE system.

Figure 2 shows a block diagram of the T3E router. The router contains a full-duplex PE interface, six full-duplex network ports, a lower-bandwidth I/O port and a central arbiter. The PE interface performs the routing table lookup at the source node and saves return routing information at the destination, presenting a simplified interface to the destination PE. The router also contains also contains a performance monitor, integrated logic analyzer and support logic for system maintenance and barrier networks.

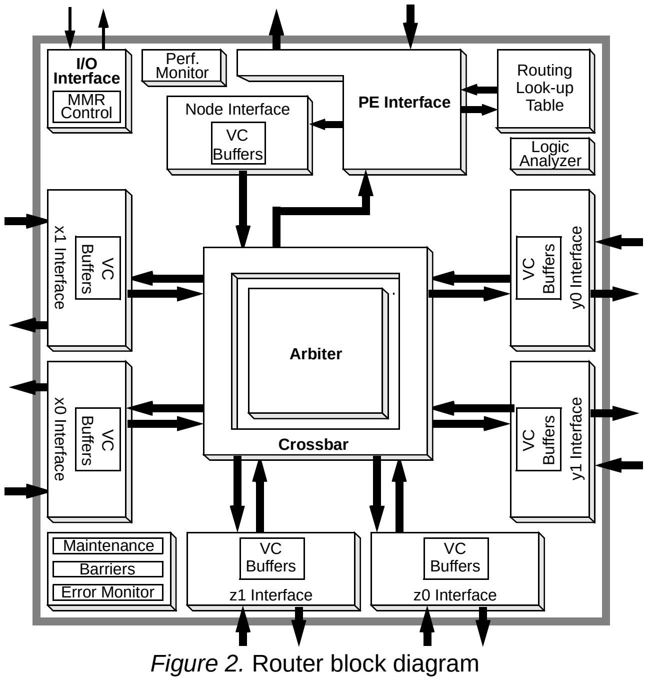

Figure 2. Router block diagram

The routers transmit *packets* of information between nodes in the system. Each packet consists of from one to ten *flits*.

Each flit consists of five *phits*. The network link width (phit size) is 14 bits, leading to a flit size of 70 bits, which carries one 64-bit word plus miscellaneous control information.

The router ASICs operate at 75MHz and use time-multiplexed transmission. During each 13.3 ns system clock, the routers transmit five phits (one flit) over each network link. The network transmission rate is thus 375 MHz and the peak link bandwidth is 600 MB/s. The maximum data payload bandwidth after protocol overheads is approximately 500 MB/s.

The router uses five virtual channels to prevent deadlock and provide adaptive routing. Credit-based flow control is used over each link for each virtual channel. Acknowledgments are sent in the opposite direction, piggybacked on data packets or idle flits. Buffers for each adaptive virtual channel at each network link input hold 22 flits, and the other virtual channel buffers hold 12 flits each. Virtual channel usage is carefully optimized to improve physical channel utilization.

The network directly supports the architecture of the T3E multiprocessor. Packet types include single-word and 8-word read and write requests, 8-word message packets, atomic memory operation packets, and special packets used for diagnostics and system configuration. The network also implements multiple virtual barrier networks, capable of providing global barrier synchronization in less than the latency of a single remote memory reference.

The remainder of the paper describes the network in more detail and provides performance information. Sections 2 and 3 describe basic routing and deadlock avoidance. Sections 4 and 5 discuss virtual channel optimization and adaptive routing, both designed to increase sustainable bandwidth. Section 6 describes the router support for virtual barrier/eureka synchronization networks. Section 7 describes the router core arbitration algorithm. Sections 8 and 9 discuss network maintenance and fault tolerance. Section 10 describes the physical network technology. Finally, Section 11 presents basic performance estimates.

2 Routing

Routing of data packets is accomplished through comparison of routing tags attached to the packets at source routers, with node addresses configured into each node. That is, absolute (static) addresses, as opposed to decrementing hop counts, are used for destination addressing. This requires that a physical “who am I” register be loaded on the router chip. Routing information is looked-up in a 544 entry table located on the router. The index into this table is simply the logical node number of the destination. For systems with more than 544 processors, the two low bits of the logical node address are untranslated so that the upper bits of the destination are used to address the table. This results in processors being mapped on a 4 processor granularity. The table can thus support up to 2176 processors. A request packet must carry the logical node number of its source to be used as the destination for the response. This value is taken from a logical “who am I” register contained on each router.

An entry in the table includes the following information: the physical address of the destination node, the deterministic path to that node, and an indication of whether adaptivity is allowed. The latter is used to disable adaptivity in the event of a faulty link in one of the minimal paths. In other words, the look-up table provides the virtual to physical destination mapping and the path information to route to that destination.

2.1 Direction-order routing algorithm

With the exception of *initial* and *final hops* (see Section 2.3), and adaptive hops (see Section 2.4), all routing is done in *direction-order.* The ordering used in the T3E is +X first, followed by +Y, +Z, -X, -Y and -Z. The combination of direction-ordering, initial-hops, and free-hops gives a fault-tolerant, acyclic virtual network.

Direction-order routing is similar to *dimension*-order routing. In dimension-order routing, the dimensions are traversed in some fixed order, say X, Y and Z. A packet first routes to the correct ordinate in the X dimension, in either the positive *or* negative direction, then satisfies the Y dimension and then the Z dimension. In direction-order routing, packets traverse the six directions in some fixed order, but the ordering of dimensions is not fixed.

This flexibility adds to the fault-tolerance of the system, as different routes between two nodes can now have different *corners*, reducing the probability that an intermediate broken node can prevent routing. For example, with the direction order given above, a route of (+X, +Y, +Z) will travel in the X, then Y, and then Z, while a route of (-X, +Y, +Z) will travel in Y, then Z, and then X. Figure 3 shows these routes in a three by three by three cube. Note that the two routes have different corner nodes. Dimension-order routing always turns on the same corners no matter which directions are taken.

Direction-order routing opens the door to routes that travel in one direction in a given dimension and later travel in the opposite direction of that same dimension. An example of this is shown in Figure 4. This allows for shorter alternate routes in degraded systems, and allows the network to reconfigure around many multiple failures that would otherwise be uncorrectable. This particular case has a string of nodes cut off from the rest of the nodes in one of the Y rings due to two broken Y links.

2.2 Initial and final hops

A single hop can be taken in a plus direction without having to add any bits to the routing information that is carried

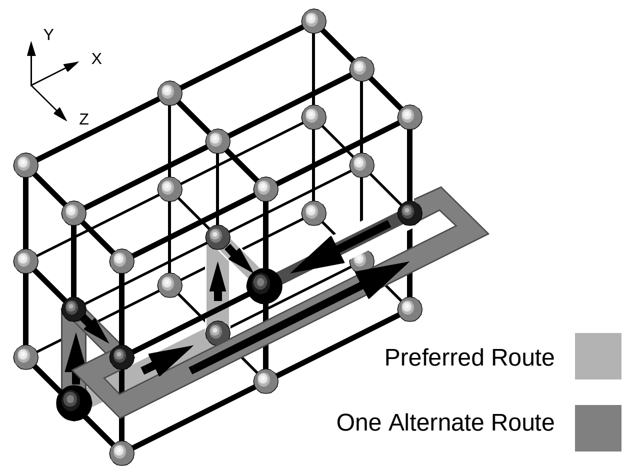

Figure 3. Alternate Routes Can Turn on Different Nodes

along with a given packet. This is accomplished through the use of an “initial direction” given to a packet just before it enters the first router. For example, a packet that routes first in the + directions can be given an initial direction of +X and its routing bits can indicate a -X, +Y, +Z path. This results in a single “initial-hop” in the +X direction followed by routes in +Y, +Z, and finally -X. This initial direction does not have to be in the opposite direction of the other route in that dimension.

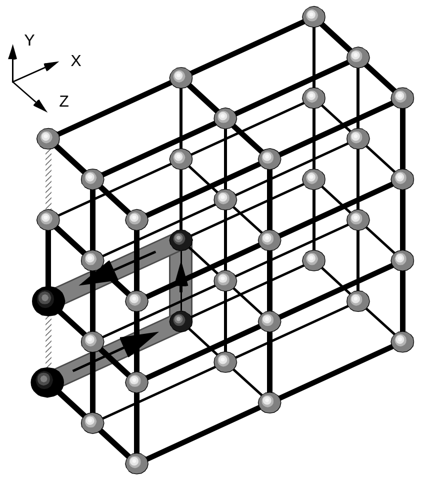

Figure 4. Use of Both +X and -X Paths to Avoid Broken Links

The T3E router allows initial hops in the +X, +Y and +Z directions, and also allows a single hop in -Z once all other routes are complete. The latter is referred to as a “final hop.”

2.3 Partial planes

The network allows for partially filled planes of nodes in order to allow a finer granularity of system size. These are referred to as partial planes. All partial planes are Z planes.

Traffic originating at a partial plane node (see Figure 5) can take an initial hop in +Z to its neighbor and then route from that point with a routing tag that can be identical to that which that neighbor uses. Traffic with a destination of a partial plane node can route to the +Z neighbor of the destination node at which point a final hop in -Z is taken. The initial and final hops only need to be used if a normal direction-order route does not exist.

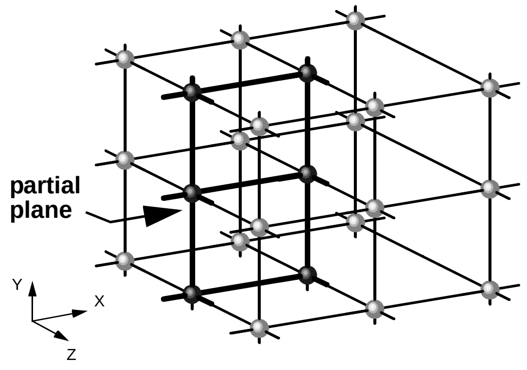

Figure 5. Subsection of Network Showing a Partial Plane

2.4 Adaptive routing

The adaptive virtual network consists of one virtual channel per physical channel. On this virtual network, adaptive packets are allowed to turn in any minimal direction (one that moves them closer to their destination). Adaptive routing may need to be disabled for some source-destination pairs in a degraded network, because it may attempt to take a broken path. Details are discussed in Section 5

3 Virtual Channels and Removal of Cyclic Dependencies

Dependencies due to turns, torus connections, and request-response pairs will deadlock the network if not removed. Cyclic dependencies are removed to form an acyclic component of the network through a combination of turn restrictions and virtual channels (VCs).

3.1 Turn cycles: direction ordering, initial hops and final hops

The acyclic virtual network breaks cycles in the mesh by restricting turns. This is accomplished by traversing the six directions in a fixed order (+X, +Y, +Z, -X, -Y, -Z) as described in Section 2.1. There are many circumstances that do not require the ordering to be this strict. One particular case is the first hop a packet makes. If the first hop is not to a dateline (see 3.2 “Torus cycles”) it can be in any of the first three directions of the ordering shown above without introducing any cycles. After this particular initial hop, the packet is allowed to route normally in the six directions of the direction ordering. A similar argument can be made for an extra hop in one of the minus directions at the end of the route. T3E allows a “final hop” in -Z, to give better access to partial planes. The acyclic property of these initial and final hops is quite similar to that underlying the turn model for partially adaptive routing [11].

3.2 Torus cycles

Cyclic dependencies introduced by the torus connections are broken through the use of a pair of virtual channels on each physical channel, which we refer to as VC0 and VC1. A node in each ring is specified as the *dateline*. With the exception of adaptive traffic, any packet that crosses the dateline on VC0 is switched by the hardware to VC1 leaving the dateline node. This prevents cyclic buffer dependencies in VC0. A packet is not allowed to cross the dateline on VC1, else cyclic dependencies could arise among VC1 buffers. An almost perfectly balanced use of virtual channel resources (under uniform random traffic) can be achieved by picking the virtual channel based on the origin and destination within a given direction (see Section 4).

Each router has a small configurable table of virtual channel assignments for every destination in each direction. Consider a 3 dimensional torus with k_x, k_y, and k_z nodes in the X, Y, and Z dimensions, respectively. Each router has a table containing $2(k_x + k_y + k_z)$ bits, where each bit corresponds to a node in one of the six unidirectional rings in which the given router is connected. The maximum radix supported by the tables is 8, so dimensions larger than 8 nodes are partially constrained in their VC assignments. This is discussed further in Section 4.

The bits in the VC lookup table determine the virtual channel assignment for a packet that has just entered a new direction. For example, upon turning into +Y, the bit corresponding to the destination's Y address in the +Y portion of the table sets the virtual channel to be used. The VC lookup table is loaded independently on each router and thus each origin-destination pair within a direction has an independent assignment. This handful of bits on each router gives an extremely flexible assignment scheme.

3.3 Request-response cycles

Request-response cycles are broken by doubling again the number of virtual channels in each direction. Thus the acyclic network has a total of four virtual channels in each direction. Request packets travel through the network on one set of virtual channels and response packets on another.

3.4 Adaptive cycles

Adaptive packets may leave the deterministic, acyclic network and route on the virtual channels in the adaptive network. The adaptive virtual network is actually cyclic, but packets that might otherwise be deadlocked can always reenter the acyclic network. This is described in Section 5.

4 Virtual Channel Optimization

In addition to deadlock prevention, virtual channels can improve network utilization by preventing a blocked packet from stalling the packets behind it, much the way turn lanes improve automobile traffic flow [5]. A packet that does not need the resource on which the blocked packet is waiting may pass the blocked packet *if* it is on a different virtual channel. The extent to which virtual channels improve network utilization thus depends upon the distribution of packets among the virtual channels.

We have found that the *virtual channel balance* – the relative traffic carried by each virtual channel – has a significant effect on overall network performance [16]. Sustained throughput is increased when the traffic load is more evenly spread across the channels. Not only does better balance increase the probability that a packet can pass a blocked packet ahead of it, but it also makes more efficient use of network buffer space, which typically is statically partitioned among the various virtual channels.

The T3E uses a similar technique to that used in the T3D [16] to improve virtual channel balance. As described in Section 2, a packet consults the VC lookup table each time it enters a new direction. Since each direction is independent, we can focus simply on optimizing the VC assignments for single rings. For every source node on a ring, a VC assignment is needed for each destination node.

Recall that all routes (source/destination pairs) on a ring that cross the dateline must use VC0 (and will switch to VC1 upon crossing the dateline) in order to prevent cyclic dependencies in the VC1 buffers. All other routes (which do not cross the dateline) are *unconstrained*. It is by choosing the VCs used for unconstrained routes that VC usage is balanced. Since the VC lookup tables are limited to 8 entries, large rings have some artificially-constrained routes. Only the lower 8 bits of the destination ordinate is used to index the table, and any routes using the same entry as a constrained route must use VC0.

The simplest VC assignment is all zeros. All packets start routing in each direction on VC0 and switch to VC1 only if/when they cross the dateline. We refer to this as the *time-of-crossing* assignment. It tends to put most of the traffic on VC0, with the exception of the links directly after the dateline, which carry predominantly VC1 traffic in large rings.

In order to optimize the assignments, we use a hierarchical traffic model, in which we consider the case of uniform communication over the whole ring, as well as uniform communication over power-of-two sized *subrings*, corresponding to jobs that are running in partitions of the machine. The search space of possible VC assignments for large rings is intractably large, so a technique based on simulated annealing is used. While a detailed explanation cannot be given here, the technique is quite similar to that used for the T3D [16].

Table 1 compares virtual channel balance for the time of crossing scheme, the optimized routes used in the T3E, and what the optimized routes *could* be without the limited VC table size. The table shows the average and worst-case link VC balance.

Ring Size	Subring Size	Time of Crossing		Optimized T3E		Optimized Full Tables	
		Avg	Max	Avg	Max	Avg	Max
4	4	1	1	0	0	0	0
8	8	.813	1	.031	.250	.031	.250
	4	.625	1	.125	.250	.125	.250
16	16	.813	1	.137	.563	.133	.563
	8	.656	1	.063	.313	.063	.313
	4	.625	1	.125	.250	.125	.250
32	32	.807	1	.220	.875	.173	.797
	16	.664	1	.246	.609	.062	.250
	8	.656	1	.488	1	.031	.063
	4	.625	1	.594	1	.125	.250

Table 1. Virtual channel balance

To compute the VC balance, we assume that each node sends a packet to every other node in the ring or subring. The balance for a given link is then the absolute difference in the number of VC0 and VC1 routes crossing the link divided by the maximum number of routes crossing any link in that ring or subring. Thus, in a subring (which is effectively a mesh rather than a torus), a relative imbalance on a link with less traffic (towards the edge of the subring) is counted less heavily than the same relative imbalance on a more heavily used link. A link with equal VC0 and VC1 traffic has a VC balance of 0. The maximum imbalance is 1.

The optimized assignments significantly improve VC balance, both for global traffic and for traffic within subpartitions of the machine. The hierarchical optimization technique significantly improves VC balance within subpartitions without significantly compromising balance for global traffic[16]. The limited VC table size in the T3E does not impact the quality of the VC assignments for ring sizes up to 16 (a system size of 1024 PEs), although it noticeably degrades the VC balance for the 32-node ring.

Bolding [2] has previously reported performance degradation from the VC assignments in the original Dally and Seitz paper on deadlock avoidance [4]. In fact, the VC balance of their assignments is almost identical to that of the time-of-crossing scheme. In their assignment, all packets use one of the two VCs, except for packets crossing the dateline (in their case wrapping around the torus), which use the other VC *before* crossing the dateline.

5 Adaptive Routing

Adaptive routing allows packets to route around local congestion in a network, thus improving sustained network throughput, especially for non-uniform communication workloads such as transposes that can create severe local hot-spots.

Adaptive routing has long been used in wide area networks, where routing decisions are made in software and packets are transmitted using store-and-forward routing [18]. Several store-and-forward adaptive routing schemes have also been proposed for multicomputer or multiprocessor interconnection networks [10][12][15]. These schemes take advantage of the fact that a packet only holds buffer resources at a single node.

More recently, there have been a number of proposals for adaptive routing in wormhole routed networks using virtual channels. Some of these schemes provide fully adaptive routing, but require a large number of virtual channels. Jessope, Miller and Yantchev's scheme, for example, requires 2^n VCs per physical channel for an n-dimensional mesh [13]. Supporting request-response traffic doubles that number and supporting tori at least doubles it again. Linder and Harden's scheme requires 2^{n-1} virtual networks for an n-dimensional mesh and n+1 VCs per virtual network to support tori. Supporting request-response traffic doubles this number.

Other proposals, such as Glass and Ni's turn model [11] or Chien and Kim's planar adaptive model [3] use a more modest number of VCs, but restrict the routing function to avoid cycles in the channel dependency graph and thus provide only partially adaptive routing. Dally and Aoki propose two schemes that allow fully adaptive routing, but either limits the number of times that a packet can adapt or detect when a blocked packet could potentially cause deadlock and force the packet to route statically for the remainder of its transmission [6].

Duato has proposed a routing mechanism that requires only a single extra virtual channel and provides unrestricted, fully adaptive, minimal-path routing [9]. This scheme relies upon a set of virtual channels and a static routing function that produces an acyclic channel dependency graph (and is thus deadlock free). To this, it adds one or more VCs that can be used for adaptive routing and *can* introduce cyclic buffer dependencies. It restricts the adaptive VC buffers, however, to contain flits from at most one packet at a time, so that a blocked packet on an adaptive channel can always switch back to the deadlock-free, non-adaptive VCs and thus make forward progress.

The adaptive routing mechanism in the T3E is an adaptation of Duato's proposal. Each physical channel contains five virtual channels, four of which are used for static routing of request-response traffic as described in Section 2. The fifth channel is used for adaptive routing of request and response packets. From any node along its path, a packet may either take the static VC in the direction prescribed by the routing function, or it may take an adaptive VC in any profitable direction (one that takes it closer to its destination).

However, we differ from Duato's proposal in that we allow multiple packets to co-reside in an adaptive virtual channel buffer. We require that the adaptive buffer have space available for the entire packet in order to route a packet on that VC, eliminating the possibility of dependencies from an adaptive VC buffer to any other buffer. This decoupling is similar to that in store-and-forward routing schemes, but we still rely on the non-adaptive, deadlock free subnetwork (rather than, say, structured buffer pools), and do not require that the packet actually be fully received before forwarding.

Our adaptive VC buffers are over two times the size of the largest packet used in the T3E. Thus, requiring that the buffer contain space for the entire packet before routing to it does not significantly affect channel utilization. Allowing multiple packets to co-reside in an adaptive VC buffer improves pipelining, avoiding bubbles introduced by waiting for the buffer to drain before routing into it. Of course, this policy prevents routing arbitrarily large packets, but that is not an issue in the T3E.

Our modification of Duato's scheme also removes the requirement of an acyclic extended channel dependency graph [9], because dependencies cannot arise between two non-adaptive VC buffers via intermediate, adaptive VC buffers. This allows us to use the optimized VC assignments discussed in Section 4, for which the extended channel dependency graph would be cyclic (a packet can start on VC0, cross the dateline on an adaptive VC, and then reenter the non-adaptive subnetwork on VC0). We also use a routing function that maps $N \times C \rightarrow \{C\}$ rather than $N \times N \rightarrow \{C\}$, that is, the choice of virtual channel is based on current virtual channel and destination rather than on current node and destination. This feature is used in our VC optimization scheme and is not allowed in Duato's adaptive routing theory [9].

Adaptive routing introduces the possibility of packet-reordering in the network. The T3E network provides two mechanisms for turning adaptive routing off. First, each packet contains a bit that specifies whether adaptive routing is allowed. The processor may issue *ordered* requests which will not adapt (although their responses will because that is not functionally visible). The routing tables can also turn off all adaptive routing for traffic between any two nodes, which is required in certain degraded configurations as discussed in Section 2.

6 Virtual Barrier/Eureka Networks

A frequent feature of interconnection networks is support for group synchronization primitives. The T3D, for example, includes a separate barrier/eureka[1] network, which consists of a four-wire-wide, degree-four spanning tree over the entire machine. Similarly, the CM5 includes a separate control network that provides a number of synchronization feature [CM5].

While barrier synchronization is quite important, these separate networks are expensive, consuming pins and wires that are used *only* for specialized synchronization. The T3E eschews the separate synchronization network for a set of *virtual* barrier/eureka networks that utilize small, high-priority synchronization packets over the existing data network.

Each processor shell contains a set of 32 barrier/eureka synchronization units (BSUs). A set of processors can be given access to a particular BSU through the virtual memory system. A BSU at a processor can be in one of several states, which a processor can read and manipulate via load and store operations to memory-mapped registers in the shell.

Figure 6 (a) shows the local state transitions for a simple barrier. An OP_BAR operation takes a given BSU from the S_BAR state to the S_ARM state. When all participating processors have armed their barriers, the network delivers completion notifications that takes the BSUs to the S_BAR state, at which point the BSUs are ready for the next barrier synchronization.

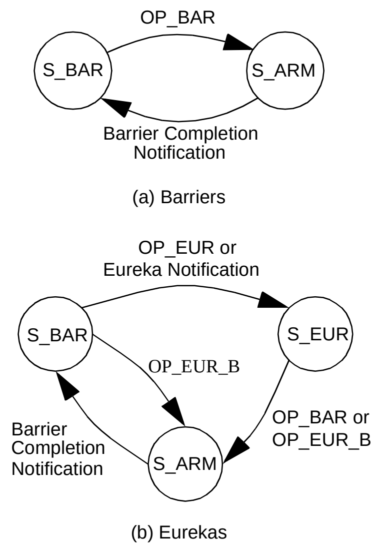
OP_BAR
S_BAR
S_ARM
Barrier Completion Notification
(a) Barriers
OP_EUR or Eureka Notification
S_BAR
S_EUR
OP_EUR_B
OP_BAR or OP_EUR_B
Barrier Completion Notification
S_ARM
(b) Eurekas

Figure 6. Simple barrier and eureka local transitions

A simple, re-usable eureka event, shown in Figure 6 (b), is a three-state transition that includes a barrier to establish that all processors have seen the eureka before performing another eureka. Starting in the S_BAR state, a single processor performs an OP_EUR. This takes its BSU to the S_EUR state, and causes the network to deliver eureka events to all other participating BSUs, taking them to the S_EUR state as well. As processors observe the eureka,

1. A eureka is completed when any *one* processor checks in, rather than requiring *all* participating processors to check in.

they indicate this by performing an OP_BAR. The triggering processor can perform a combined OP_EUR and OP_BAR using an OP_EUR_B. Once all processors have joined the barrier, the network delivers barrier notifications that place all BSUs in the S_BAR state, ready for the next eureka event.

When a processor joins a barrier or signals a eureka, it sends a single-flit barrier/eureka packet to its local router. Each router maintains a register for each of the 32 BSUs. This register allows the node to be configured as an internal node in the BSU's logical spanning tree. The register indicates which of the six network directions plus the local processor are children in the tree, and which direction (if any) is the parent. It also keeps track of the set of children that have signalled a barrier.

When all children have signalled a barrier, or when any child signals a eureka, a corresponding signal is sent up to the parent (by sending the parent a barrier/eureka packet), or, if the node is the root of the tree, completion signals are sent to all of the children (also via barrier/eureka packets). Completion signals are broadcast hierarchically to all children in a barrier/eureka tree, and result in appropriate changes to the child BSUs. Barrier/eureka completion can be detected via polling or can optionally interrupt the leaf processors. A more complete description can be found in [17].

Barrier/eureka packets use their own virtual channel and are transmitted with highest priority. While not as fast as a dedicated barrier network, this scheme keeps global barrier/eureka latency to less than that of a single remote memory reference. The barrier virtual channel requires no buffering and cannot block. Barrier/eureka packets can *always* be sunk by the router's BSU registers.

7 Arbitration

This section describes the arbitration algorithm used in the router core. The algorithm attempts to optimize the number of packets routed each cycle, and guarantees forward progress to all packets.

Figure 2 shows how the functional blocks are laid out around the T3E router. The x0, x1, y0, y1, z0, and z1 interfaces each contain a short-circuit path to allow traffic, that is not turning, to exit the chip immediately without ever entering the crossbar. One of the responsibilities of the arbiter is to let these interface blocks know if there is a reason why the short-circuit path is not to be used. If packets are queued up in a given interface, the arbiter tells the other interfaces that they could get some traffic from the crossbar so they should stop using the short-circuit path.

Any time the arbiter has ruled out the use of the short-circuit path or there are packets that are turning, the interfaces must make requests to the arbiter in order to get access to the desired outputs. The six interfaces mentioned previously each contain buffering for five virtual channels. The node interface contains four VCs. The crossbar only has one path for each of these interfaces, so only one of the multiple VCs within an interface input may flow at any given time. To deal with this, the interface chooses one and only one of its VCs to make a request to the arbiter. The selection of these input VCs is performed in a round-robin fashion among the VCs that contain packets. Once the selection is made, this input VC keeps the right to continue requesting until one of three things happens:

- the packet fails to win its requested output VC,
- the packet bubbles (i.e. the tail and head have been separated resulting in some idle cycles),
- the packet completes, or
- the output VC becomes blocked.

The winner at this interface input arbitration makes its requests to one or two output arbiters. One of these requests is for one of the four deterministic VCs in the physical output corresponding to the lowest ordered direction that is not yet satisfied. For example, if the packet wants to travel in -x, +y, and +z, then its lowest ordered direction is +y as described in Section 2. An optional second request is made if the packet is marked as adaptive. This request goes to the adaptive VC on the output corresponding to the highest ordered direction yet to be satisfied (-x in the previous example). The arbitration at each of these outputs is described below. If the requesting interface wins the physical output at both of its requested interfaces, it flows on the adaptive choice. The one exception to this is when both the deterministic and the adaptive choices are the same physical output. In this case, the deterministic VC is used.

To flow out of a given physical output, the requesting interface must win the use of a VC on that output. In addition, the physical output must be won by that particular output VC. The fight between the VCs at a given physical output is typically arbitrated round-robin. The one exception to this is that the adaptive VC has lowest priority, unless it has a packet that has started, but not yet completed. The winning VC keeps control of the output until:

- the packet bubbles (i.e. the tail and head have been separated resulting in some idle cycles),
- the packet completes, or
- the output VC gets blocked.

The arbitration for the output VCs is a little more complicated. Once an output VC has been won, it is kept until the packet is complete or flows on another output or VC. Arbitration for the adaptive output VCs is simply round-robin. Once an input VC has made a request for a given deterministic output VC, that output VC remembers that the request was made allowing that input to continue to fight for the VC even after the input VC has relinquished the right to request outputs. Recall that a failure to win the output VC results in another input VC getting the chance to make its requests. If an output VC is won by an input VC that has stopped requesting it, the VC simply waits for that input VC to get its turn to use the VC so that no other requester may take it away. In the event that an input VC takes its adaptive path, the previously requested output VC is informed so that it can restart its arbitration for other suitors.

8 Network Administration

Router hardware registers are used for purposes of system configuration, maintenance, performance monitoring, virtual barrier tree configuration, etc. Examples include the router lookup table and reset switches for every ASIC on that router's node. A special virtual network, that is not flow controlled, exists to aid in system initialization and diagnostics. This network is only accessible through privileged commands.

Most registers on the router are addressed as memory, but several critical registers use this special virtual network. Due to the lack of flow control on this virtual network, no deadlocks can occur and a badly confused network can be diagnosed and restored. Conflicts on these virtual channels result in lost packets. That is, one packet eats the other and then continues. The only conflicts that can exist involve packets that are both on this virtual network. Thus, it is a simple matter for the software to ensure that conflicting traffic is not sent.

In addition to routing without flow control, these packets bypass the usual router lookup table and use delta addressing with routes in the following order: ±X, then ±Y, then ±Z, then ±X, then ±Y, and finally ±Z. This gives extreme flexibility in path selection even when the system is not yet configured.

9 Fault Tolerance

The network has several features that add to the entire system's fault tolerance.

The special virtual network described in Section 8 allows recovery from very serious network problems without the need for power-cycling the machine. Even a completely deadlocked network can be cleaned up.

The routing table along with the logical "who am I" registers allows for the nodes to be logically renamed. Thus, a customer can purchase extra nodes and use them as hot spares to be brought on-line in the event of a node failure. Applications continue to see a contiguous range of virtual PE numbers, even though one or more of the PEs may not be physically contiguous.

The router chips can be set to isolate and route around faulty nodes so that the system is no longer communicating with the downed nodes. This allows "hot swapping", in which a faulty board is removed from the system and replaced while the operating system and user jobs are kept running on the healthy nodes.

Routing tables can specify alternate paths between each pair of nodes, using the features described in Section 2. Direction order routing allows up to 8 different paths to be taken, simply by changing the direction of travel in the three dimensions. Initial and final hops add even more potential routes.

The network also provides end-to-end error correction. The router chips detect channel errors along the route of travel and then correct them at the destination router.

10 Technology

The T3E router ASIC is implemented in LSI Logic's 500K process with the following features:

- 3 metal layers
- 0.8 μm metal pitch
- 3.3 V supply
- 85 ps inverter delay
- .5 μm drawn channel length
- .35 μm effective channel length
- 90 Angstrom oxide thickness
- 85 μm^2 memory cells

The router uses a 16.7 mm × 16.7 mm die with ~10^6 raw gates, of which 50% are utilized.

The ASIC operates at 75MHz and uses 5X time-multiplexed transmission. A clock signal is sent along with the data to capture and demultiplex. The transfer from this demultiplexed domain to the core clock of the receiver is handled synchronously with tuned clocks and cables. During each 13.3 ns system clock, the routers transmit 5 phits, over each network link. The network transmission rate is thus 375 MHz. The maximum data payload bandwidth of a link after protocol overheads is approximately 500 MB/s.

The signaling is differential with a nominal 600 mV swing and has a maximum transmission distance of approximately 1 meter.

11 Performance

Performance of interconnects is typically judged by both latency and bandwidth. While serious attention was given to latency in the T3E network, we tended to favor sustained bandwidth over reduced latency in the implementation. With any significant contention in the network, however, features such as the optimized VC assignments and adaptive routing will reduce the realized packet latency.

Table 2 shows the router fall through time (input pins of one chip to input pins of the next chip) in the absence of contention. An optimized straight-through path is aimed primarily at large systems; smaller systems tend to have more turns than straight-through routes. The network is customized to the T3E, and performs certain actions at the source and destination routers (such as routing table lookup and packet format conversion) that increase latency. End-point latency actually varies somewhat with packet type, but is approximately 10 system clocks.

Routing action	Latency
Straight through	3 clks (40 ns)
Turn	6 clks (80 ns)
End-point	10 clks (133 ns)

Table 2. Router fall-through latencies

Figure 7 shows the maximum and average round-trip packet latencies *vs*. system size, assuming uniformly random destinations and using the latencies from Table 2. The latencies do not include memory access time or processing by the source and destination PEs. The latency is twice the delay from the input of the router at the source node to the input of the control chip at the destination node.

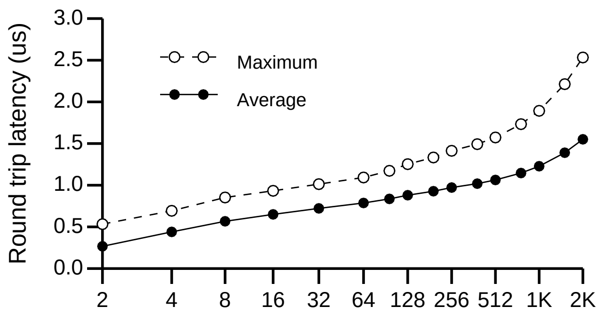

Figure 7. Round-trip network latency

Sustainable bandwidth depends upon traffic type. Table 3 shows the lengths, in flits, of single-word and vector *Get* and *Put* packets. These are the T3E equivalent of read and write packets, and transfer either one or eight 64-bit words [17]. Implementation issues late in the design led to higher overheads than originally intended. As a result, most packets contain two header flits, except for the single-word Get response, which contains one, and the single-word Put request, which contains three.

Packet type	Request	Response
Single-word Gets	2	2
Vector Gets	2	10
Single word Puts	4	2
Vector Puts	10	2

Table 3. Packet lengths (flits)

Table 4 shows the maximum sustainable link bandwidths as determined by the peak link bandwidth of 600 MB/s and the packet lengths in Table 3. The payload efficiency is shown in parentheses for each case. Sustainable bandwidth is lower for symmetrical traffic, as request and response packets share link bandwidth.

Traffic type	One-way	Symmetrical
Single-word Gets	300 (50%)	150 (25%)
Vector Gets	480 (80%)	400 (67%)
Single word Puts	150 (25%)	100 (17%)
Vector Puts	480 (80%)	400 (67%)

Table 4. Payload bandwidth (MB/s)

Figure 8 illustrates the per-processor global bandwidth, as a function of system size. This represents the bandwidth available to each PE assuming all PEs are making requests uniformly across the whole machine (no locality). It is equal to twice the bisection bandwidth, divided by the number of nodes. This number must be scaled by the appropriate efficiency from Table 4 to calculate the per-processor payload bandwidth for a given traffic type.

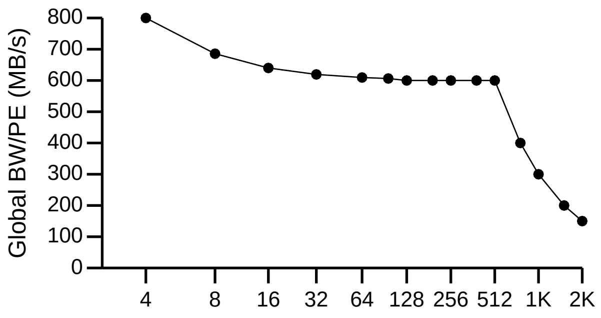

Figure 8. Bandwidth profile

For system sizes up to 64 nodes, the processor-network *port* is the bottleneck, since a fraction (N-1)/N of all packets must cross it. For systems with 128 to 512 nodes (which include a dimension of radix 8), the processor port and most-used network links are equally utilized. Beyond 512 nodes, the Y dimension becomes the bottleneck. Normally, a 3D torus would retain the global, per-processor bandwidth of the 1024-node system up to 4096 nodes (16x16x16). Since the mechanical design of the T3E limits the X and Z radix to 8, however, the maximum bisection bandwidth remains fixed at 153.6 GB/s and the per-processor bandwidth drops accordingly. Scalability beyond 1024 processors, however, is of limited practical interest; the vast majority of T3E systems sold will be 512 or fewer processors, for which the network appears completely scalable.

Acknowledgements

The work represented in this paper was partially supported by the Advanced Research Projects Agency under Agreement No. MDA972-95-3-0032 dated 12 January, 1995.

References

[1] Agarwal, A., “Limits on Interconnection Network Performance,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 398-412, October 1991.

[2] Bolding, K., “Non-Uniformities Introduced by Virtual Channel Deadlock Prevention,” *Technical Report UW-CSE-92-07-07*, University of Washington, Seattle, WA, July 12, 1992.

[3] Chien, A. A. and J. H. Kim, “Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors,” *Proc. 19th International Symposium on Computer Architecture*, pp. 268-277, May 1992.

[4] Dally, W. J. and C. Seitz, “Deadlock-Free Message Routing in Multiprocessor Interconnection Networks,” *IEEE Transactions on Computers*, pp 547-553, May 1987.

[5] Dally, W. J., “Virtual Channel Flow Control,” *Proc. 17th International Symposium on Computer Architecture*, pp 60-68, May 1990.

[6] Dally, W. J. and H. Aoki, “Adaptive Routing using Virtual Channels,” *IEEE Transaction on Parallel and Distributed Systems*, pp. 466-475, April 1993.

[7] Digital Equipment Corporation, DECchip 21064-AA Microprocessor Hardware Reference Manual, 1992.

[8] Digital Equipment Corporation, Alpha 21164 Microprocessor Hardware Reference Manual, 1995.

[9] Duato, J., “A new theory of deadlock-free adaptive routing in wormhole networks,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 1320-1331, December 1993.

[10] Gelernter, D., “A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks,” *IEEE Transactions on Computers*, pp. 709-715, October 1981.

[11] Glass, C. J. and L. M. Ni, “The Turn Model for Adaptive Routing,” *Proc. 19th International Symposium on Computer Architecture*, pp. 278-287, May 1992.

[12] Gopal, I. S., “Prevention of Store-and-Forward Deadlock in Computer Networks,” *IEEE Transactions on Communications*, pp. 1258-1264, December 1985.

[13] Jesshope, C. R., P. R. Miller and J. T. Yantchev, “High Performance Communications in Processor Networks,” *Proc. 16th International Symposium on Computer Architecture*, pp. 150-157, May 1989.

[14] Linder, D. H. and J. C. Harden, “An Adaptive and Fault Tolerant Wormhole Routing Strategy for *k*-ary *n*-cubes,” *IEEE Transactions on Computers*, pp. 2-12, January 1991.

[15] Ngai, J. Y. and C. L. Seitz, “A Framework for Adaptive Routing in Multicomputer Networks,” *Symposium on Parallel Algorithms and Architectures*, 1989.

[16] S. Scott, and G. Thorson, “Optimized Routing in the Cray T3D,” Proceedings of the Parallel Computer Routing and Communications Workshop (PCRCW), May 1994, Springer-Verlag Lecture Notes in Computer Science, pp 281-294.

[17] Scott, S. L. “Synchronization and communication in the T3E Multiprocessor,” to appear in the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, October 2-4, 1996.

[18] Tanenbaum, A. S., *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1981.