# The Gemini Network

**Rev 1.1**

**Cray Inc.**

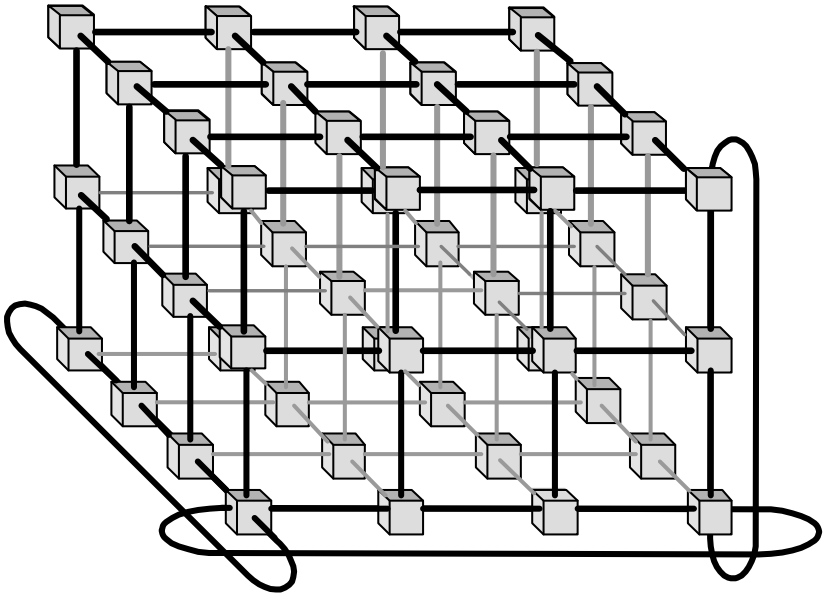
For further information on the Gemini network or other Cray products please contact crayinfo@cray.com

For sales inquiries please contact: 1-877-CRAY-INC (1-877-272-9462) domestic or +1-651-605-8817 international

# 1 Overview of the Gemini Network

Gemini is the new network for Cray's supercomputer systems. It develops the highly scalable Seastar design used to deliver the 225,000 core Jaguar system, improving network functionality, latency and issue rate. Gemini uses a novel system-on-chip (SoC) design to construct direct 3D torus networks that can scale to in excess of 100,000 multi-core nodes. Gemini is designed to deliver high performance on MPI applications and filesystem traffic, in addition it provides hardware support for global address space programming. Gemini enables efficient implementation of programming languages such as Chapel, UPC and Co-Array Fortan on massively parallel systems.

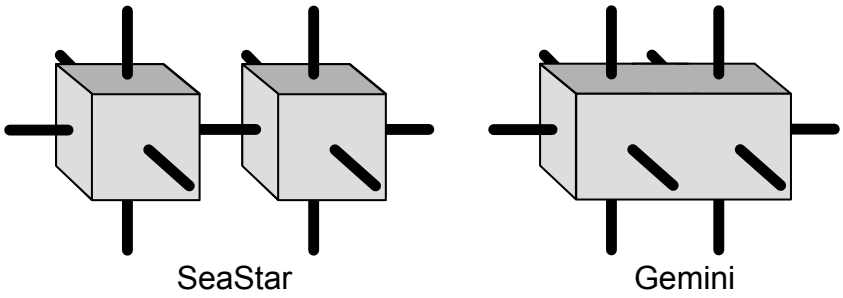
Figure 1 Cray XT 3D Torus Network



Each Gemini ASIC provides two network interface controllers (NICs), and a 48-port "Yarc" router[1]. Each of the NICs has its own HyperTransport 3 host interface, enabling Gemini to connect two Opteron nodes to the network. This 2 node building block provides 10 torus links, 4 each in two of the dimension ('x' and 'z') and 2 in the third dimension ('y'), as shown in Figure 2. Traffic between the two nodes connected to a single Gemini is routed internally. The router uses a tiled design, with 8 tiles dedicated to the NICs and 40 (10 groups of 4) dedicated to the network.

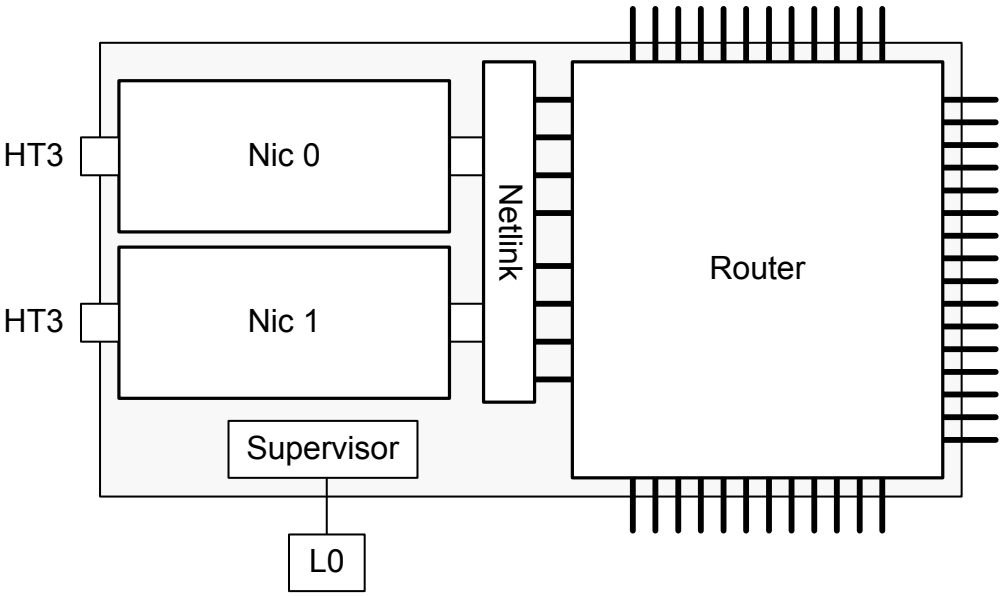
1 The Yarc router was first used in Cray's X2 vector system.

Figure 2 Each SeaStar router provides one node of the 3D Torus, Gemini provides two.




The block structure of the Gemini design is illustrated in Figure 3. The Netlink block connects the NICs to the router. It also handles changes in clock speed between the NIC and router domains. The supervisor block connects Gemini to an embedded control processor (L0) for the blade and hence the Cray hardware system supervisor (HSS) network, used for monitoring the device and loading its routing tables.

Figure 3 Gemini blocks




Gemini is designed for large systems in which failures are to be expected and applications must run on in the presence of errors. Each torus link comprises 4 groups of 3 lanes. Packet CRCs are checked by each device with automatic link level retry on error. In the event of the failure of a link, the router will select an alternate path of adaptively routed traffic.Gemini uses ECC to protect major memories and data paths within the device.

For traffic designated as adaptive the Gemini router performs packet by packet adaptive routing, distributing traffic over lightly loaded links. With 8 links connecting each Gemini to its neighbors in the ‘x’ and ‘z’ directions and 4 links in the ‘y’ dimension, there are multiple paths available. Hashed deterministic routing can be selected as an alternative when a sequence of operations must be performed in order.

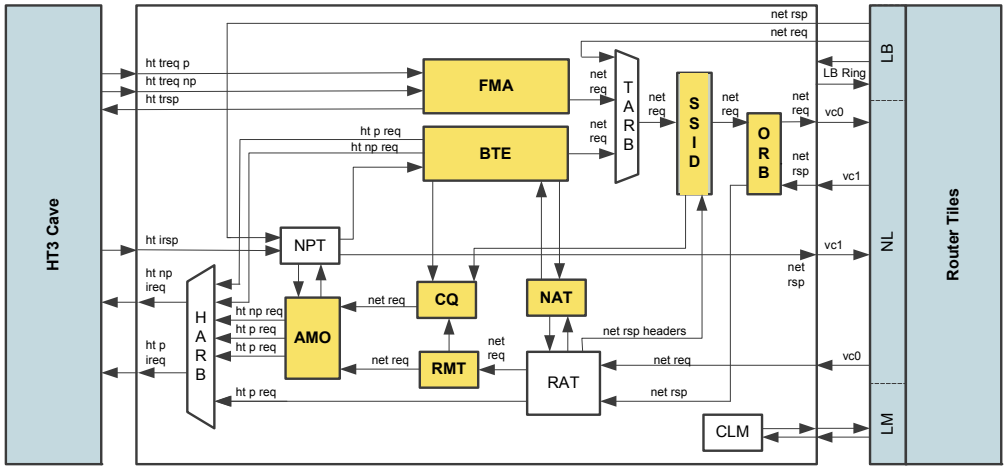
Gemini provides the ability for user processes to transfer data directly between nodes without OS intervention. For example, one process in a parallel job can initiate a put directly from its memory to that of another process. To do this it specifies the data (or source address), the destination virtual addresses, the destination process id and the size of the transfer. Additional hardware primitives include remote get, atomic operations, block transfer and completion notification. The Gemini NIC is a hardware pipeline that maximizes the performance of these simple operations. More complex communications protocols such as message passing and TCP/IP are implemented using these primitives.

User space communication is supported by the User Gemini Network Interface (uGNI) and Distributed Memory Application (DMAPP) APIs. These libraries are called by Cray MPI and Shmem. DMAPP is also used in the run time for Cray Chapel, UPC and Co-Array Fortran compilers. Inter-kernel communication is provided using Kernel Gemini Network Interface (kGNI) which provides both messaging and RDMA. The Lustre filesystem is supported via a Lustre Network Driver (LND) for kGNI. Other filesystems such as NFS, GPFS and Panasas are provided via DVS, the Cray Data Virtualization Service layered over LND. TCP/IP communication over the Gemini Fabric is provided by the IP over Gemini Fabric (IPoGIF) module.

# 2 Gemini NIC

Each Gemini ASIC has a pair of NICs, each with its own HyperTransport 3 interface (known as the HT Cave and shown on the left hand side of Figure 4). The NICs are connected to the Gemini router via the Netlink block (on the right hand side of Figure 4). The NIC is a hardware pipeline. The node issues commands, writing them across the HyperTransport interface. The NIC packetizes these requests and issues the packets to the network, with output flowing from left to right at the top of Figure 4.

Figure 4 Block Diagram of the Gemini NIC




Packets are routed across the network to a destination NIC. The input pipeline flows from right to left at the bottom of the figure. The Gemini network employs a 3-tuple, the

*Network Address,* to specify a logical, address in a user process on a remote node. The address consists of a processing element identifier (or *PE),* a *Memory Domain Handle* (MDH) associated with a memory segment registered at the remote node, and an *offset* into this segment. This 58-bit network address extends the physical address space of the node, enabling global access to all of the memory of a large system.

Gemini supports both virtual addressing and virtual PEs. The MDH is combined with the offset to generate a user virtual address in the remote process.Virtual PEs (ranks in MPI parlance) used by the application are translated on output by the Node Translation Table (NTT) to obtain the physical PE. Constraints on physical resources limit the size of the NTT, only the top 12 bits of the PE are translated. Very large jobs are layed out in a regular fashion with low bits of the virtual and physical PEs being equal, or they can use physical PEs.

## 2.1 Fast Memory Access (FMA)

Fast Memory Access (FMA) is a mechanism whereby user processes generate network transactions, puts, gets and atomic memory operations (AMO), by storing directly to the NIC.The FMA block translates stores by the processor into fully qualified network requests. FMA provides both low latency and high issue rate on small transfers. On initialization the user process is allocated one or more FMA descriptors and associated FMA windows. Writes to the FMA descriptor determine the remote processing element and the remote address associated with the base of the window. A write of up to 64 bytes to the put window generates a remote put. Storing an 8 byte control word to the get window generates a get of upto 64 bytes or a fetching AMO. FMA supports scattered accesses by allowing the user to select which bits in an FMA window determine the remote address and which the remote PE. Having set the FMA descriptor appropriately one can, for example, store a unique word of data to each process in a parallel job by simply storing a contiguous block of data to the FMA window. The DMAPP library provides a light-weight wrapper around this functionality for the Cray compilers and libraries.

FMA supports source-side synchronization methods for tracking when put requests have reached a *globally ordered* point at the target and when responses to get requests have reached a globally ordered point in the local node. It is also possible to issue puts that generate destination-side synchronization events at the target node, enabling a process on that node to be notified of new data, or to poll a single completion queue for its arrival.

## 2.2 Block Transfer Engine (BTE)

The Block Transfer Engine (BTE) supports asynchronous transfer between local and remote memory. Software writes block transfer descriptors to a queue and the Gemini hardware performs the transfers asynchronously. The BTE supports memory operations (put/get) where the user specifies a local address, a network address and a transfer size. In addition the BTE supports channel operations (send) where the user specifies a local address and a target, but no target address. Channel semantics require the user to have pre-posted a receive buffer with the target BTE. By default there is no guarantee of

completion ordering in block transfers issued by a given Gemini. Fence operations are used where necessary to ensure that one transfer is completed before another starts.

In general FMA is used for small transfers and BTE for large. FMA transfers are lower latency. BTE transfers take longer to start, but once running can transfer large amounts of data (upto 4GB) without CPU involvement.

## 2.3 Completion Queue (CQ)

Completion queues provide a light-weight event notification mechanism. The completion of a BTE or FMA transaction can generate an event in a user (or kernel thread) specific queue. Completion events can be generated on either the source or the target node. They include both user data and transaction status information.

## 2.4 Atomic Memory Operation (AMO)

Gemini supports a wide range of atomic operations, those with put semantics such as atomic add and those with get semantics such as conditional swap. Gemini maintains an AMO cache, reducing the need for reads of host memory when multiple processes access the same atomic variable. Host memory is updated each time the variable is updated (lazy update mechanisms are also provided to reduce load on the host interface), but network atomics are not coherent with respect to local AMD64 memory operations - all processes must use a Gemini application interface to update an atomic variable.

## 2.5 Synchronization Sequence Identification

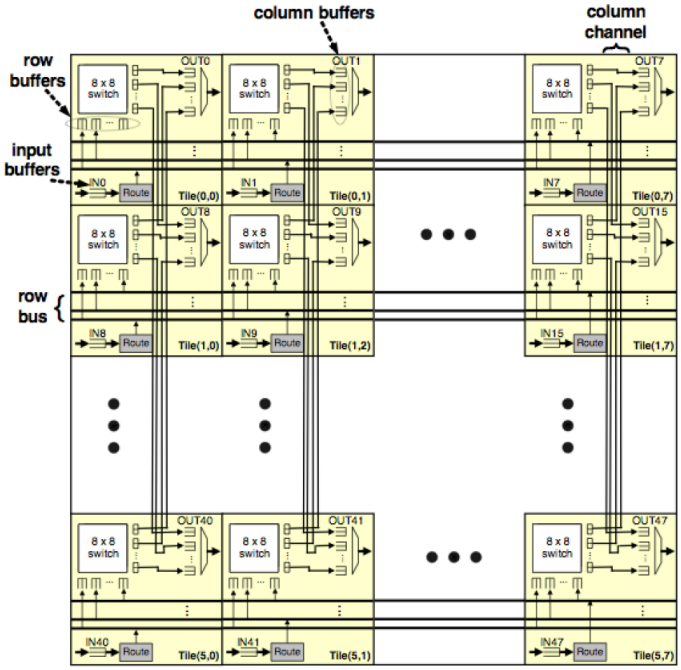
Gemini uses a mechanism known as *Sequence Identification* to track the set of packets that make up a transaction. Every packet in the sequence contains the same Synchronization Sequence Identification (SSID). Packets can be delivered in arbitrary order; each contains a network address and can be committed to memory as soon as it arrives - there is no need for reorder buffering. The sequence as a whole completes and CQ events are generated when all packets have been delivered. This mechanism is implemented using the SSID and Output Request Buffer (ORB) blocks on the output side and the Receive Message Table (RMT) block on the input side. The RMT caches active SSID state avoiding a network round trip for performance critical operations. It also matches BTE send requests to queued receive descriptors.

# 3 Gemini Router

The building block for the Gemini router is the *tile* (see Figure 5). Each tile contains all of the logic and buffering associated with one input port, one output port, an 8×8 switch

and associated buffers. In Gemini, each tile's switch accepts inputs from six row buses that are driven by the input ports in its row, and drives separate output channels to the eight output ports in its column. Using a tile-based microarchitecture facilitates implementation, since each tile is identical and produces a very regular structure for replication and physical implementation in silicon.

Figure 5 Gemini 48-port tiled router



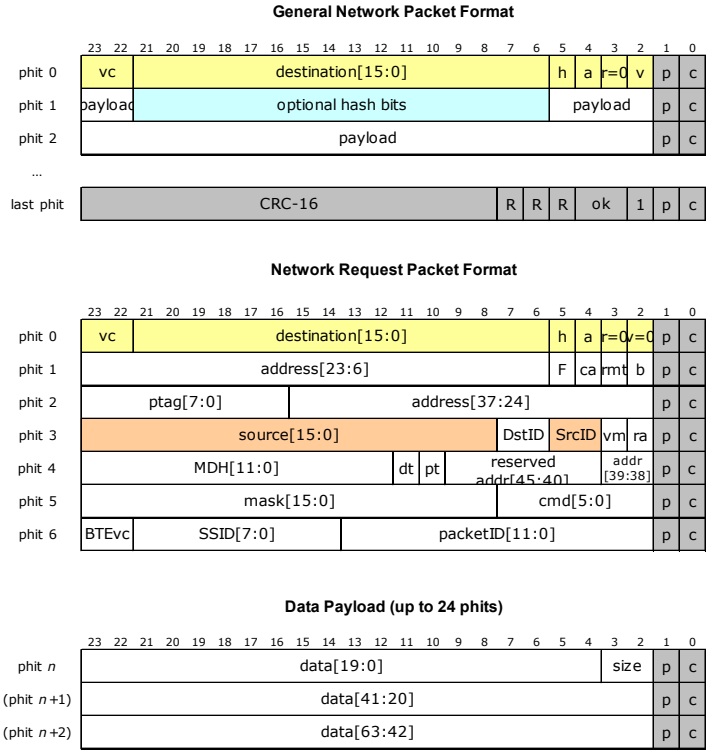

The tile-based design is best understood by following a packet through the router. A packet arrives in the input link of a tile. When the packet reaches the head of the input buffer, a routing decision is made to select the output column for the packet. The packet is then driven onto the row bus associated with the input port and buffered in a row buffer at the input of the 8×8 switch at the junction of the packet's input row and output column (at the crosspoint tile). At this point the routing decision must be refined to select a particular output port within the output column. The switch then routes the packet to the column channel associated with the selected output port. The column channel delivers the packet to an output buffer (associated with the input row) at the output port multiplexer. Packets in the per-input-row output buffers arbitrate for access to the output port and, when granted access, are switched onto the output port via the multiplexer.

Gemini uses virtual cut-through flow control across the network links, but uses wormhole flow control internally due to buffer size constraints. Network link input buffers are deep enough to account for credit round-trip latency and the maximum size packet.

Packets are variable size and divided into 24-bit *phits* (physical units) for transmission over network links. Write request packets have a 7-phit header, up to 24 phits of data and a single phit end-of-packet that denotes the last phit of a packet and contains status bits for error handling. A 2-phit response packet is generated for each request (3 phits on error). Get responses include a payload of up to 24 phits.

The head phit controls routing, it specifies the destination, the virtual channel and details of how the packet is to be routed (see Figure 6).

Figure 6 Gemini Packet Format




Each Gemini has a unique 16-bit identifier, specified by the *destination* field within each packet. NICs and hence Opteron nodes are specified using a 2-bit identifier for the *source* (SrcID) and *destination* (DstID). The combined 18-bit address uniquely identifies every node in the system. The *v* field specifies the virtual channel; Gemini uses one virtual channel for requests and another for responses. The *r, a* and *h* fields control routing. If the *r* bit is set, then the packet will be *source routed* and must contain a routing vector in the payload. Source routing is only used for diagnostics. If the *a* (adapt) bit is set, the packet is routed adaptively, otherwise the packet is routed using a deterministic hash constructed from the source and destination ids and (optionally if the *h* bit is set) the remote address. The fields shaded gray in Figure 6 contain side band data used by the link control block.

An 8-byte write requires an 11-phit request (7 header, 3 data and 1 EOP) and a 2-phit response. A 64-byte cache-line write requires 32 request phits (7 header, 24 data, and 1 EOP) and 2 response. A 64-byte get comprises an 8-phit request (7 header plus EOP) and a 27-phit response (2 header, 24 data and EOP)

## 3.1 Gemini Fault Tolerance

Gemini provides a 16-bit packet CRC, which protects up to 64-bytes of data and the associated headers (768 bits max). Within each Gemini, major memories are protected using ECC.

Gemini links provide reliable delivery using a sliding window protocol. The receiving link checks the CRC as a packet arrives, returning an error if it is incorrect. The sending link retransmits on receipt of an error. The link block includes a send buffer of sufficient size to cover the round trip.

The CRC is also checked as a packet leaves each Gemini and as it transitions from the router to the NIC, enabling detection of errors occurring within the router core. If the checksum is incorrect, the packet is marked as bad and passed on; it will be dropped by the destination Gemini.

Completion events include details of the status of each transaction allowing software to recover from errors. HSS/OS interfaces allow the reporting of any of these errors at the point of occurrence.

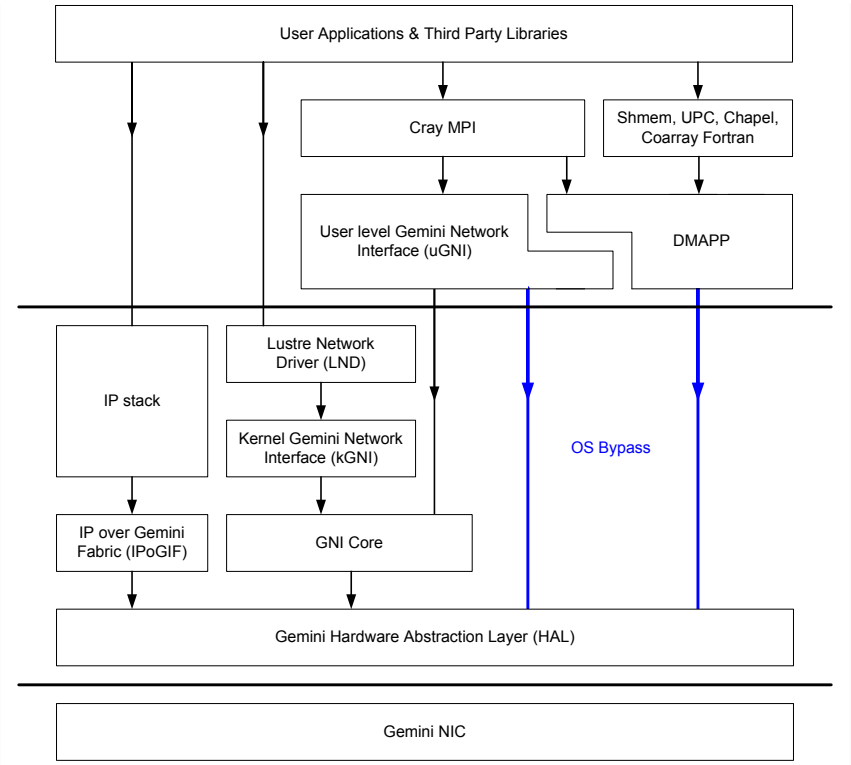
Each 3D Torus link is made up of 4 Gemini links. The Gemini adaptive routing hardware will spread packets over the available links. If a link fails, the adaptive routing hardware will mask it out. In the event of losing all connectivity between two Gemini, it is necessary to route around the problem (this also happens when a board is removed). The management software quiesces the network, computes new routing tables and then re-enables the network.

# 4 Gemini Software

Gemini supports both kernel communication, through a Linux device driver and direct user space communication, where the driver is used to establish communication domains and handle errors, but can be bypassed for data transfer. Parallel applications typically use a library such as Cray MPI or SHMEM in which the programmer makes explicit communication calls.

Alternatively they may uses a programming model such as UPC, Chapel or Co-Array Fortran in which the compiler automatically generates the inter-process communication calls. Both approaches are layered over the user level Gemini Network Interface (uGNI) and/or the Distributed Memory APPlications (DMAPP) library (as shown in Figure 7) which perform the Cray network specific operations. Kernel modules such as the Lustre communicate via the kernel Gemini Network Interface (kGNI).

Figure 7 Gemini Software Stack




Gemini is a connectionless design, rather than creating point to point connections between processes, a parallel application begins by establishing a communications domain. Connectionless designs are preferred in High Performance Computing as they scale well to large system sizes - in particular their memory usage remains constant as the size of the job increases. Each communication domain specifies a set of processes (or kernel threads) that are allowed to communicate. Creating a communications domain is a privileged operation that results in a protection tag being allocated. The hardware adds this tag to each packet ensuring isolation of communication domains between users. Each process is also allocated a virtual process id (or rank). The Gemini NIC performs virtual to physical translations, converting ranks to physical locations. Any attempt to communicate with a process outside of the communication domain generates an error. Program initialization follows a sequence of steps in which the placement scheduler (ALPS) first creates a communication domain and then starts the user processes. The processes sign on to the communication domain, create their completion queues and register memory with the communication domain. Having completed this sequence of steps, the processes in a parallel job can initiate either put/get or send/receive style communication. These operations are all asynchronous, with completion events being generated when an operation or sequence of operations has been completed. This approach promotes programming models in which communications is initiated early so as to hide latency. Gemini optimizes for this, supporting both large numbers of pending operations and high issue rates.

## 4.1 Programming Environment

Cray MPI uses the MPICH2 distribution from Argonne. The MPI device for Gemini is layered over uGNI (point-to-point) and DMAPP (collectives). Use of FMA gives MPI applications the ability to pipeline large numbers of small, low latency transfers; an increasingly important requirement for strong scaling on multi-core nodes. MPI message matching is progressed by each call or using a helper thread. The Gemini block transfer engine is used to provide high bandwidth and good overlap of computation and communication. Cray MPI uses dedicated mailboxes (for frequently used communication paths) combined with a per-node message queue ensuring that memory requirements are modest, even on the largest jobs. FMA and AMO transactions are combined to implement high performance collective operations.

SHMEM provides an explicit one-sided communication model. Each process executes in its own address space but can access segments of the memory of other processes (typically the static data segment and symmetric heap) through a variety of put/get calls and collectives. Cray systems since the T3D have supported SHMEM. Its implementation for Gemini provides the application programmer with fine grain control of communication with a minimum of overhead.

Gemini is designed to provide efficient support of emerging Partitioned Global Address Space (PGAS) programming models as well as MPI. Cray's Chapel, UPC and Co-Array Fortran compilers use a common runtime implemented with DMAPP. The DMAPP library includes blocking, non-blocking and indexed variants of put and get together with scatter/gather operations and AMOs. This interface is also provided for third parties developing their own compilers, libraries and programming tools. Compiler generated communication typically results in large numbers of small irregular transfers. Gemini's FMA mechanism minimizes the overhead of issuing them and its high packet rate maximizes performance.

# 5 Gemini Packaging

Cray systems are constructed from dual socket Opteron nodes. Four nodes are packaged on a blade card, eight blades in a chassis and three chassis in a cabinet, for a total of 96 nodes (192 processor sockets) per cabinet. A network mezzanine card is fitted to each blade. This card contains a pair of Gemini ASICs connected such that each blade provides a 1×4×1 element of the overall 3D torus. The chassis backplane provides connections in the 'z' direction so that the eight cards in a chassis form a 1×4×8 element of the torus.

The remainder of the links (88 per chassis) are taken to connectors on the rear of the chassis. A range of cabling options support different sizes of systems with cabinets layed out in a 2D grid as illustrated in Table 1, where N is the number of cabinets per row and R is the number of rows. Each row must have the same number of cabinets.

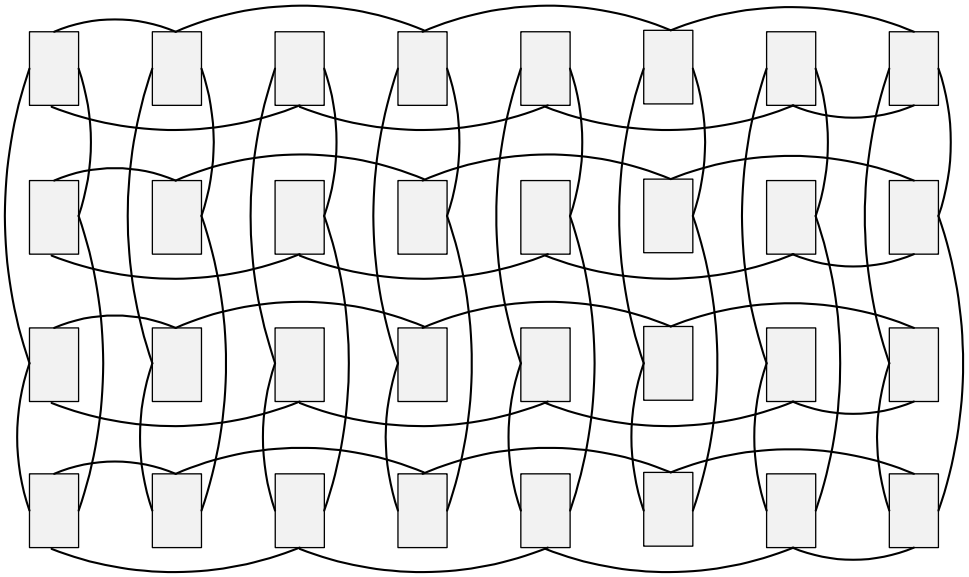
Table 1 Gemini Network Topologies

| Class | Cabinets | Geometry | Connectivity |
|---|---|---|---|
| 0 | 1-3 | 3N×4×8 | x between chassis<br>y within chassis<br>z within chassis |
| 1 | 4-16 in one row | N×12×8 | x between cabinets<br>y within cabinet<br>z within chassis |
| 2 | 16-48 in two rows | N×12×16 | x between cabinets in a row<br>y within cabinet<br>z connects the rows |
| 3 | > 48 | N×4R×24 | x between cabinets in a row<br>y between rows<br>z within cabinet |

Restrictions on cable lengths make it impractical to close the torus by running cables from one end of a row or column to the other. Instead alternate cabinets are cabled together as illustrated in Figure 8 for a 32 cabinet system. Cables close the ‘x’ and ‘z’ dimensions. An even number of rows is strongly preferred.

Jaguar, the largest system shipped to date, has 200 cabinets arranged in 8 rows of 25 cabinets. The Gemini design, like that of SeaStar, its predecessor, allows large systems such as Jaguar to be constructed from a single scalable unit, one cabinet, using copper cables. No external routers are required

Figure 8 Inter-cabinet cabling for 32 cabinets in 4 rows



## 5.1 Gemini in ‘m’ series systems

Cray’s entry level ‘m’ series systems also use the Gemini interconnect. These systems comprise a single row of 1-6 cabinets connected in a 2D torus. Single cabinet systems use a 4×8, 8×8 or 12×8 torus. Multi-cabinet systems use a 4N×24 torus where N is the number of cabinets.

## 5.2 Upgrading SeaStar to Gemini

The Gemini mezzanine card is pin compatible with the SeaStar network card used on Cray XT5 and XT6 systems allowing them to be upgraded to Gemini. The upgrade is straightforward, each blade is removed and its network card is replaced. There are no changes to the chassis, cabinets or cabling.

## 5.3 Cray Models

Cray uses XT*n* names to denote SeaStar models and XE*n* names to denote Gemini.

Table 2 Cray Model names

| Name | Year | Node | Gemini Name |
|---|---|---|---|
| Cray XT3 | | Single socket Opteron nodes | |
| Cray XT4 | | Single socket Budapest nodes | |
| Cray XT5 | 2008 | Dual socket Barcelona/Shanghai/Istanbul | XE5 |
| Cray XT5m | 2009 | Dual socket Barcelona/Shanghai/Istanbul | XE5m |
| Cray XT6 | 2010 | Dual socket Magny Cours | XE6 |
| Cray XT6m | 2010 | Dual socket Magny Cours | XE6m |

# 6 Gemini Performance

## 6.1 Clock Speed

The Gemini NIC operates at 650 MHz, the router at 800MHz and the link SERDES at 3.125GHz. The speed of the HyperTransport interface ranges from 1600MHz to 2600Mhz depending on the node type.

## 6.2 Latency & Bandwidth

End-to-end latency in a Gemini network is determined by the end point latencies and the number of hops. On a quiet network, the end-point latency is 1.0μs or less for a remote put, 1.5μs or less for a small MPI message. The per hop latency is 105ns on a quiet network.

The Gemini NIC can transfer 64 bytes of data in each direction every 5 cycles. Maximum bandwidth per direction is 64 × 650 / 5 = 8.3 GB/s. Injection bandwidths depend on the speed of the HyperTransport interface and the method of transfer. The interface is 16 bits wide and transfers data on both edges of the clock, giving a raw bandwidth of 9.6 GB/sec in each direction at 2400 MHz. On FMA put the HyperTransport overheads are 12 bytes on up to 64 bytes of data, limiting the peak bandwidth to 8 GB/sec. For BTE transfers there is a 12-byte read request followed by a 76-byte posted write for every 64-byte data packet. For symmetric BTE traffic the peak bandwidth of the host interface is 7 GB/sec in each direction after protocol. The netlink block injects packets into the router, distributing traffic across the 8 processor tiles. Each 64-byte write is transferred as 32 × 24-bit request phits (7 header, 24 data and 1 end of packet) with a 2-phit response (3 on error), with each processor tile transfering one 64-byte packet in each direction every 32 cycles.

Each of the 10 Gemini torus links compises 12 channels in each direction operating at 3.125GHz. Link bandwidths are 4.68GB/sec per direction. Efficiency on 64-byte transfers is 63%, significantly higher than competing products. Bandwidth after protocol is 2.9 GB/sec per link per direction. The torus network provides multiple links between nodes. Packets are adaptively distributed over all of the available links, enabling a single transfer (a point-to-point MPI message for example) to achieve bandwidths of 5GB/sec or more. Higher bandwidths can be achieved between processes on the same node or processes on nodes connected to the same Gemini. Where multiple user processes or kernel threads are sending data at the same time (the common case for multi-core nodes), the Gemini NIC can inject packets at host interface bandwidth, with the router spreading traffic out over all of the available links. Note that the bandwidth at which a node can send data to multiple destinations or receive data from multiple destinations exceeds the point-to-point bandwidth between any pair of nodes. The former is limited by injection bandwidth and the latter is limited by link bandwidth.

### 6.2.1 Bisection Bandwidth

To determine the bisection bandwidth of a Cray Gemini system, consider the dimensions of the torus X×Y×Z. Next determine whether the ‘y’ dimension of the torus is closed. The ‘x’ and ‘z’ dimension are closed in standard configurations. Now consider the three products

$$Lz = X \times Y \times 2,\ Ly = X \times Z \times Cy,\ Lx = Y \times Z \times 2$$

Where Cy is 1 if the ‘y’ dimension is open and 2 if it is closed. Select the smallest of the three products. This is the number of links crossing the worst case bisection. Gemini links are bidirectional; so to calculate the bisection bandwidth, multiply the worst case number of links by twice the link speed.

$$\text{bandwidth} = 2 \times \text{links} \times 4.68 \text{ GB/s}$$

For example, in a 40 cabinet system arranged as 4 rows of 10 racks the torus is 10×16×24

Lz = 10×16×2=320, Ly=10×24×2=480, Lx=16×24×2=768

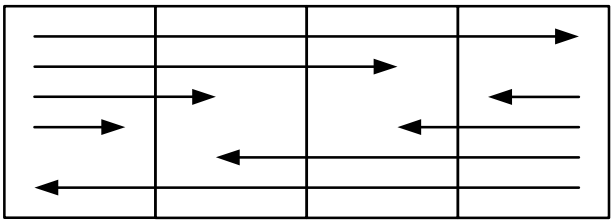
The ‘z’ dimension has the fewest links and so the bisection bandwidth is

2×320×4.68 = 2995 GB/s

### 6.2.2 Global Bandwidth

Bisection bandwidth is frequently used to characterise the performance of a network, but it is rare that all processes in one half of the machine communicate simultaneously with all of the processes in the other half. A more common case is that half of the communication crosses the bisection and half remains local (see Figure 9), for example in a global exchange (or all-to-all) or purely random communication.

Figure 9 Global communication pattern



The global bandwidth of a torus network is twice its bisection, 5990GB/s in the 40 cabinet example above.