QsNet^{III} an Adaptively Routed Network for High Performance Computing

Duncan Roweth, Trevor Jones
Quadrics Limited, Bristol, United Kingdom,
Email: {duncan, trevor}@quadrics.com

***Abstract*—In this paper we describe QsNet^{III}, an adaptively routed network for High Performance Computing (HPC) applications. We detail the structure of the network, the evolution of our adaptive routing algorithms from previous generations of network and new applications of these techniques. We describe other HPC specific features including hardware support for barrier and broadcast and large numbers small packets. We also describe the implementation of the network.**

I. Introduction

QsNet^{III} is the third generation of Quadrics High Performance Computing (HPC) network, its design builds on that of earlier Quadrics and Meiko products [2], [3], [6]. QsNet^{III} is a multi-stage switch network, also know as a "folded-Clos network" [19] or "fat tree" [18]. Nodes are connected via network adapters to the first stage of crosspoint routers. Routers are connected together via additional stages. Each additional stage allows a larger larger number of nodes to be connected, the number being determined by the radix of the router and the balance of links up and down. QsNet networks are constant bisection bandwidth, in that each stage has equal numbers of links down towards the nodes and up to additional stages (see figure 1).

In a fat tree network any destination can be reached from any top stage switch, so the routing decision amounts to selection of a top switch. Early work on the use of these networks for massively parallel systems [20] demonstrated that they were capable of supporting pair-wise communication without contention, provided the traffic pattern was known in advance and an appropriate set of routing tables was constructed. This type of source routing formed the basis of our first designs in 1994 [6], data packets were steered between nodes using precomputed routes supplied by the adapters. A process could select one of four routes to use in transmitting data. This choice could be made deterministically, "static" routing, or at random, "oblivious" or "dispersive" routing. This mechanism allowed an application to use default routes provided by the operating system or its own optimised routes. Much the same approach is used by Myrinet today [21]. Commodity networks such as Ethernet and Infiniband use static routing schemes with route tables held in the switches; loading application specific routes is rare.

This paper is structured as follows. In section II we describe experience from previous generations of network and its influence on the QsNet^{III} design. In section III we describe the use of adaptive routing in QsNet^{III} . In section IV we describe hardware support for HPC specific features such as barrier and broadcast. In section V we discuss the implementation. In section VI we compare QsNet^{III} to other HPC networks. We conclude with comments on the direction of future work.

II. Background

Our experience and that of our customers' previous generations of networks is that static or oblivious routing routing worked reasonably well for simple communications patterns that persisted over the runtime of the application. For example, when mesh based data was distributed over processes, the communications patterns associated with shifts in each dimension of a grid are non-contending. However, for complex but regular communication patterns, such as all-reduce or all-to-all, or for irregular communication patterns the behaviour of statically routed networks scaled badly with network size. The same problems are evident today in both Ethernet and Infiniband, performance on all-to-all scales badly with system size [8] and optimisations to the routing tables provide only modest benefit.

In 2000 Quadrics introduced QsNet [2], [4], [22], the first multi-stage switch network to perform packet by packet adaptive routing. Switches made dynamic routing decisions based on link state. QsNet was used widely in first AlphaServer SC systems including ASCI-Q [10], [20] and then large Linux clusters, notably those at Lawrence Livermore National Lab (LLNL) [5]. It demonstrated scalable performance on complex communication patterns to 1000+ nodes. The design was refined in 2004 with the introduction of QsNet^{II} . LLNL report that it achieves in excess of 85% of host adapter bandwidth on all-to-all or bisection bandwidth tests running on 1000+ node systems [5]. Their QsNet^{II} system, Thunder, has only 55% of the point-to-point bandwidth of their newer Infiniband system, Atlas[1] but when 1024 nodes all use the network at the same time the average bandwidth seen by each process is 40% higher on Thunder [12]. The worst case bandwidth, critical to many tightly coupled parallel applications, is reported to be 2.6 times better on Thunder.

III. Adaptive Routing in QsNet^{III}

The QsNet^{III} design follows that of its predecessors in that the core components are a network adapter Elan5 and a crosspoint router Elite5. Details of the adapter design can be found in [23]. In this paper we focus on the network.

[1]Atlas has a PCI-Express host interface, Thunder uses PCI-X.

1550-4794/08 $25.00 © 2008 IEEE
DOI 10.1109/HOTI.2008.31

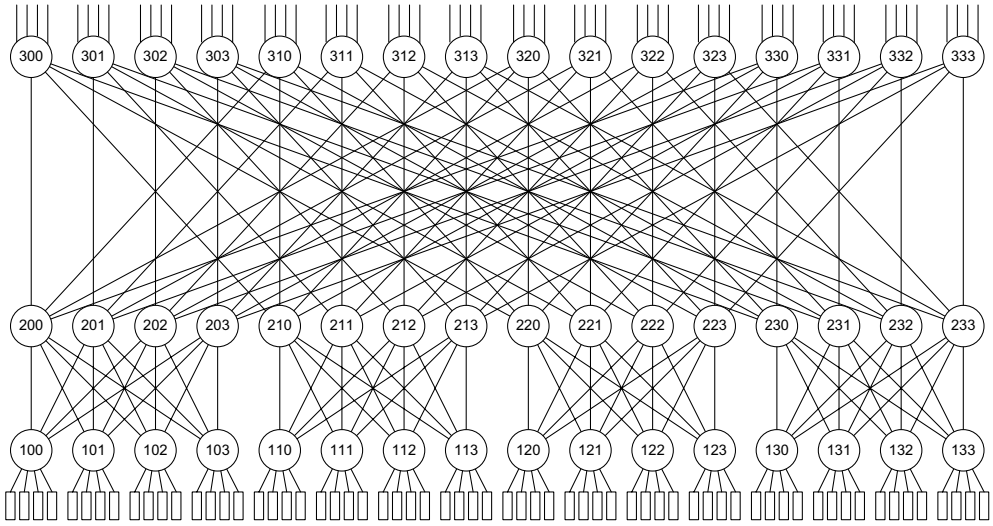

Fig. 1. Three stage switch network of radix 8. QsNet^{II} uses this design to implement its node switch, with 64 links down to the nodes and 64 links up to an additional staget of switches. The QsNet^{III} design takes the same approach with higher radix routers.

With the introduction of QsNet^{III} Quadrics has increased the radix of our routers from 8 to 32, allowing larger networks to be constructed from fewer stages of switch. For example a 2048 node network has 3 stages instead of 5. QsNet^{III} links are bi-directional, so all top switch links can connect down, a network of n stages supports up to $32 \times 16^{n-1}$ nodes. The use of high radix routers reduces the number of components required to build the network, and hence its cost and the number of routing decisions required. We have also developed the adaptive routing algorithms. A router can adaptively select from arbitrary sets of links. The decision on which link to use is made based on link state and loading. Each output link has 32 virtual channels each of which can be waiting on an acknowledgment. The number of pending acknowledgments feeds into the routing decision, allowing us to select a route from the subset of lightly loaded links. We have also implemented a variety of selection methods to allow us to experiment with different routing strategies.

Before looking at wider uses of adaptive routing it is important to consider the communication model. QsNet is fundamentally a memory based communication device, not a stream device. Every packet carries with it the virtual address of a data object. This is either the memory address to write data to (put) or read data from (get) or it is the address of a queue in the network adapter allowing us to optimise support for operations where the destination address is not known by the sender. QsNet adapters are virtual memory devices with their own MMU [3] facilitating direct one sided communication. Consider first a simple put

```
void *put(char *src, char* dest,
          size_t size, int rank)
```

Were `src` is source virtual address in the local process initiating the put and `dest` is the destination virtual address in the remote process identified by its `rank`. The put call transfers `size` bytes. It returns an opaque handle that can be tested later for completion.

If the amount of data being transferred is small it will be contained within a single packet, larger transfers will be broken into multiple packets by the sending adapter. The initiating process needs to know that the whole transfer is complete but the order in which the data is delivered is arbitrary. Similar ordering rules apply for MPI. Messages between a pair of processes must arrive in the same order as they are sent, but there is no requirement for bulk of the data to be delivered in byte order.

A large transfer will be split into many packets, which could all take different routes to the destination. They may also be replayed as a result of contention or errors. The important step, that must only be executed once, is the final one in which both sides determine that the operation has completed. QsNet^{III} distinguishes these cases, using a light weight protocol for bulk data transfer and a slightly more expensive sequence number based protocol for atomic operations.

With this approach we are free to stripe packets over multiple links (Elan5 has two, future adapters will have more), increasing bandwidth. When an error occurs[2] the network returns a NACK (not acknowledgment) to the sending adapter and it retransmits, a different route may be selected. With our model there is no requirement to wait for one packet to complete before another can be sent and no requirement to buffer packets at the destination so that data can be delivered in byte order.

On receipt of a put request the network adapter translates the user rank to a physical destination (QsNet virtualises

[2]With a link speed of 25Gbps in each direction, 2 links and a bit-error-rate of 10^{-15} we would expect to see an error once every second on a 2048 node network

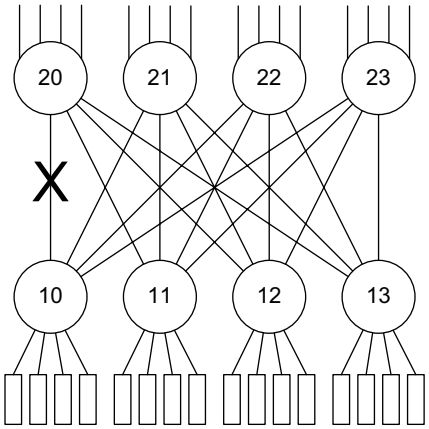

Fig. 2. Adaptive routing around a broken link. Traffic between router 10 and router 11 is automatically routed around the broken link between router 10 and router 20.

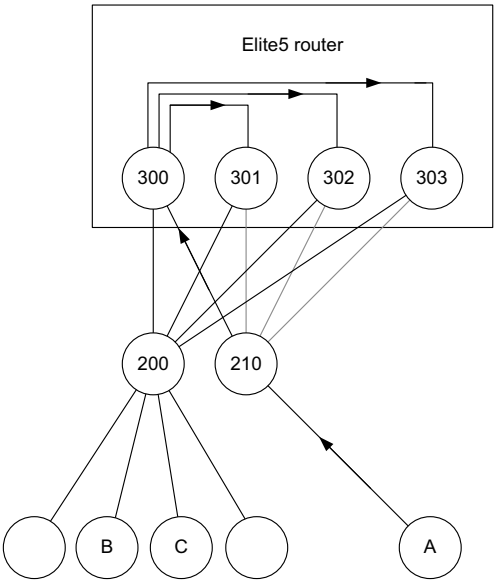

Fig. 3. Adaptive routing within the top switch of a 512-way network. A single router provides 4 top switches. Traffic from 300 can be routed via 301, 302 or 303 if the path to 200 is busy.

all addresses and destinations seen by the application) and generates packets which the network delivers, returning ACKs or NACKs. On receipt of a NACK the packet is automatically retransmitted. On the destination side data is written to user memory without intervention, there are no interrupts (provided the page is mapped) and no need for a cooperating remote process. This approach maps directly into PGAS programming models such as Shmem [14], ARMCI [15], UPC [16] and CoArray Fortran [17]. Where the destination address is not known, for example in MPI message passing [13], message headers are written to a hardware queue at a known address. Once a send and a receive have been matched the destination address is known and a DMA can be used to transfer the bulk data.

In common with previous generations of QsNet our primary use of adaptive routing is in selection of a top switch. In figure 1 router 200 is free to select from the links connecting it to routers 300, 310, 320 or 330. The decision on which link to use proceeds as follows: First we select a subset of lightly loaded links based on link activity and a configurable threshold of the number of pending operations. Then we select from this subset on a "first free", "next free" or random basis. The time to route a packet across an Elite5 is less than 40 nanoseconds[3]. This compares with 200 to 250 nanoseconds for other devices.

Our second example of adaptive routing is in the case of failed links (see figure 2). This is an important case for large production systems that need to run on in the presence of errors. If the link between routers 10 and 20 is broken, router 10 can select alternate paths via routers 21, 22 or 23 on a packet by packet basis, spreading the additional load over them. Traffic to 10 from 11, 12, or 13 might be routed via 20, giving it no means of reaching 10. In this event router 20 will generate a NACK and the source will retransmit. Router 20 will also generate an invalid destination trap. This trap and that generated by the failure of the link are caught by the management system. If the link is restored quickly it need take no action, but if the fault persists it can either reset router 20's links (so that no traffic flows this way) or update the routing tables on routers 11, 12 and 13 so that they don't use router 20 to reach the destinations connected to switch 10. On QsNet^{II} we were restricted to the first technique. On QsNet^{III} we can use either approach and the higher radix of the routers reduces the impact of the fault. The same can be achieved using oblivious or dispersive routing provided the adapters know to avoid the broken link. Static routing schemes perform badly in the presence of failures, with one of the links typically having to carry twice as much traffic. Our approach is simple and has proven to be highly effective.

QsNet^{III} provides two other opportunities for adaptive routing, within the top switches and on the final link to the destination node.

Where the required top switch size is less than that of an Elite5 router (for example a 1024 nodes requires a radix 8 switch at the top stage) each router can provide multiple top switches. This provides a further opportunity for adaptive routing; the top switches that share a router each provide an equivalent route down. The choice of which link to use can be made according to their load.

In figure 3 traffic from A to B takes the link in to top switch 300. If the link from 300 to 200 is busy with traffic to C then adaptive routing allows us to select alternate paths via 301, 302 or 303 as all four top switches are provided by the same Elite5 router.

QsNet^{III} has a final opportunity for adaptive routing, on the

[3]The time to cross an Elite5 is slightly higher than on QsNet^{II} as we have SerDes (Serializer / Deserializer) devices driving each link on the new product; the increased latency through a switch is more than compensated for by the reduced number of stages

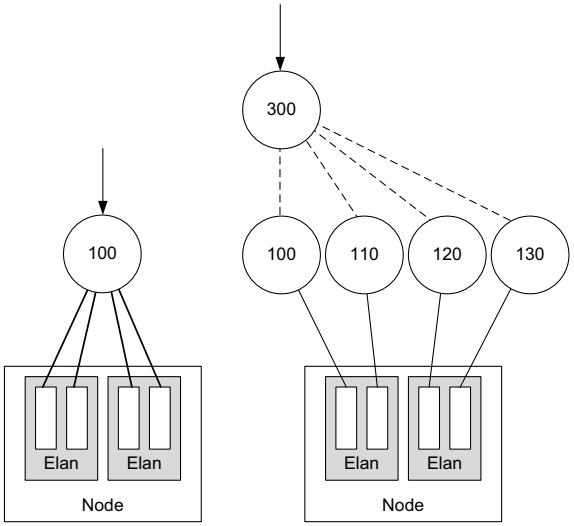

Fig. 4. Adaptive routing over the links in to a node. Two cases. In the first (left hand side) all links are connected to the same router. In the second (right hand side) the links from one node are connected to different routers.

link from the network to the destination adapter (or adapters). Where there are multiple routes in to a node the choice of which to use can be made based upon load. This allows us to reduce end-point contention by selecting a free or lightly loaded inputer.

In the simplest example two or more links in to an adapter (or a node if it has multiple adapters) are connected to the same router. The routing decision is made on the final link.

However, where a node has multiple paths to the network we may want to connect them to different physical switches so as to improve the resilience of the system. The adaptive routing decision must then be made higher in the network.

In figure 4 we consider two cases. In the first a node with two adapters each with two links is connected to 4 ports on the same switch and the adaptive routing decision is made by router 100 for traffic to this node. This is good from the point of view of minimising contention, but not optimal from a resilience standpoint. If the links are connected to 4 routers (100, 110, 120 and 130) in different switch chassis', then the we protect ourselves from cable, line card or switch chassis failure and the adaptive routing decision is made by a router in stage 3.

IV. HPC Specific Features of QsNet^{III}

QsNet^{III} has a number of other HPC specific features including hardware, support for barrier and broadcast and optimised support for small packets.

A. Barrier & Broadcast

QsNet^{III} provides hardware barrier and broadcast support in the network, an important feature for HPC applications in which the performance of collective operations controls scalability. On receipt of a broadcast packet a switch looks up the broadcast group identifier (which replaces the destination rank) and transmits the packet on all links associated with the broadcast group. Broadcast routes are set up such that packets are first sent up the network to a router high enough to span the range of destination nodes and then down the tree to these nodes. This process continues until the packets arrive at the adapters where they are acknowledged in the normal way. The switches recombine the acknowledgments, sending an ACK or NACK back towards the source when it has received a reply from each link associated with the broadcast group. A single acknowledgment is returned to the source. This approach allows data to be sent to all nodes in much the same time as it can be sent to any one. The network loading is high, but significantly less than that required to implement a conventional store and forward broadcast.

The QsNet^{II} broadcast mechanism [1], [11] supported selection of a contiguous range of destination nodes. This approach worked well when a system was running a small number of large jobs, but less so when there were many jobs and the scheduler was not able to allocate contiguous ranges of nodes. In QsNet^{III} we have enhanced the broadcast mechanism to allow the selection of an arbitrary subset of output links at each stage.

Note that the destination virtual address for all processes in a broadcast is the same. Where destination addresses cannot be guaranteed to be symmetric (MPI is a case in point) data is broadcast to a buffer allocated by the library and copied out to the user buffer. The same approach is used for multi-core nodes running multiple processes per node, with the Elan library allocating buffers in shared memory. Large broadcasts are pipelined so as to overlap the network broadcasts and the subsequent local copies.

A broadcast packet consists of a write transaction, a destination virtual address and data. Support for barriers is provided through network conditionals consisting of a read transaction, a destination virtual address and a word of data. On receipt of a network conditional the adapter reads the contents of the specified address (held in adapter memory for speed) and compares its value with that in the packet. If the two match an ACK is generated, otherwise a NACK. With this primitive in place a barrier occurs as follows: as each process enters the barrier it updates a sequence number and writes the new value to the barrier *ready* word. All processes except the root then poll on the *done* word. The root process uses a network conditional to check that all processes have entered the barrier, retrying until it receives an ACK. It then uses broadcast to set the done word. Simple extensions to this mechanism allow for non-blocking barriers and constructs of the form *wait until every process has reached sequence number n*. Polled barriers of this type are appropriate when all processes are expected to be ready at the same time. An alternate approach is to build a tree amongst the adapters, initialising a counted event and a thread in each. As each process enters the barrier it increments the event counter, as does the arrival of a message from its children. When the count reaches one plus the number of children the thread is

scheduled and its sends a message up the tree. These messages arrive at the root adapter when all processes have joined the barrier. The root broadcasts back the done word as above. This technique is appropriate when processes are expected to enter a barrier a widely varying times. This approach can be extended to perform reduction operations on data supplied by each process [7]. Both algorithms are progressed in the adapter and the root node. Neither require all processes to be scheduled in order to progress and as such scale well on large systems [10].

B. Small Packet Support

One of the most demanding challenges for HPC networks is to sustain a high proportion of link bandwidth on small packets. QsNet^{III} small packet support includes both the adapters, the switches, and the links between them. The adapters have multiple packet engines, each capable of generating network packets (see [23] for details). Each adapter can have up to 32 packets outstanding on each link at any point in time, increased from two on QsNet^{II} . QsNet^{III} switches support 32 virtual channels per link, an acknowledgment can be outstanding on each one. This provides for good overlap of data packets and acknowledgments and hence improved link utilisation.

QsNet^{III} supports a variable packet size, of up to 4K bytes. We expect to use a packet size of 1K in most systems. Packets are split into 256 byte segments. Each segment begins with either an 8-byte header followed by up to 248 bytes of data or a 4-byte continuation and up to 252 bytes of data. The segment header contains a virtual channel number, length, destination, priority, a timestamp and a 16-bit CRC calculated on the header. The adapter adds a 32-bit CRC calculated over the payload and a packet id to end of each packet (8 bytes more). For a small put the payload consists of a 64-bit virtual address and the data to be written (32 or 64 bits), 32 bytes in total.

Each Elite5 link has a 256 byte input buffer per virtual channel and a dynamically assigned pool of 16 such buffers, 8K bytes in total. This buffering allows small high priority packets to overtake on 256 byte segment boundaries. This mechanism reduces the impact of large transfers (filesystem traffic for example) on small puts or gets sharing the same link.

The DMA engine splits a large transfer into multiple packets, issuing each one as data arrives from the sending host. A 1K packet will take 40-200ns to traverse the network (32-2048 nodes) and 300-400ns to write to memory at the destination. An ACK can be returned as soon as the data has arrived or when the write to memory is complete. We issue ACKs early on all packets except the last in order to maximise bandwidth. We expect to sustain full line rate using 2 or 3 virtual channels per link. As the packet size decreases the number of pending operations required to saturate the link will increase. For small transfers the rate at which the host can update partial cache lines is likely to be the limit on performance.

V. Implementation

Elan5 and Elite5 are implemented using an 90 nanometer LSI/TSMC G90 process. They are semi-custom ASICs with 500MHz and 312 MHz system clocks respectively. Both devices use a high performance BGA package, Elan5 with 672 pins, Elite5 with 982. Maximum power consumption of Elan5 is 17W and that of Elite is 18W. QsNet^{III} links comprise 4 lanes (transmit/receive pairs) operating at 6.25GHz (25 Gbit/s in each direction). Link encoding is 8b10b. QsNet^{III} links support either Quadrics memory protocols or 10 Gbit/s Ethernet, 10GBase-CX4.

QsNet^{III} networks of up to 128 nodes are constructed from a single switch chassis. Larger "federated" networks are constructed from a two stage hierarchy of 256-port switches. Each "node" switch provides links down to the nodes and links up to a second stage of "top" switches that connect the node switches together. The general rule for construction of these networks is that $N \times M$-way node switches must be connected by $M \times N$-way top switches. With one adapter in each node and one link per adapter this network connects $M \times N$ nodes. For QsNet^{III} M is 128 and the radix of the top switch controls the number of ports, maximum network size is $128 \times N$ ports[4]. For a constant bandwidth network 128 top switches are required. Each chassis provides multiple top switches, the number of chassis required depends on the radix of the top switch as shown in table I. Where full bisection bandwidth is not required the number of top switches can be reduced.

Physical constraints on PCB and chassis designs have led us to a design in which the switch chassis houses up to 16 cards. A single slot card has 16 front panel links with QSFP connectors and 16 links to the backplane. This configuration is used for the node switch, a two stage design, with 8 cards in the lower stage providing 128 links down to the nodes, 8 cards in the upper stage providing 128 links up to the top switches and a backplane that connects the two groups of cards together. Top switches (of radix up to 32) are built using a card with all 32 links taken to the front panel; the QSFP connectors are mounted above and below the PCB. This card occupies two slots. The switch chassis houses 8 such cards.

Other chassis designs are possible, but this approach, combined with the decision to use QSFP connectors throughout allows us to construct networks of between 16 and 16384 nodes with the minimum of components, a single chassis, backplane and a choice of two switch cards. Figure 5 illustrates the construction of a 1024-way network. The numbers of components required to provide networks of a given size are summarized in table I.

Note that the top switch radix does not need to be a power of 2. For example, a single Elite5 router can provide $6 \times$ 5-way or $3 \times$ 10-way top switches, reducing the number of components required to build non power of two networks, e.g. 640 or 1280-way.

[4]The implementation is the same as that for QsNet^{II} except that each the number of ports per switch chassis has increased from 128 to 256.

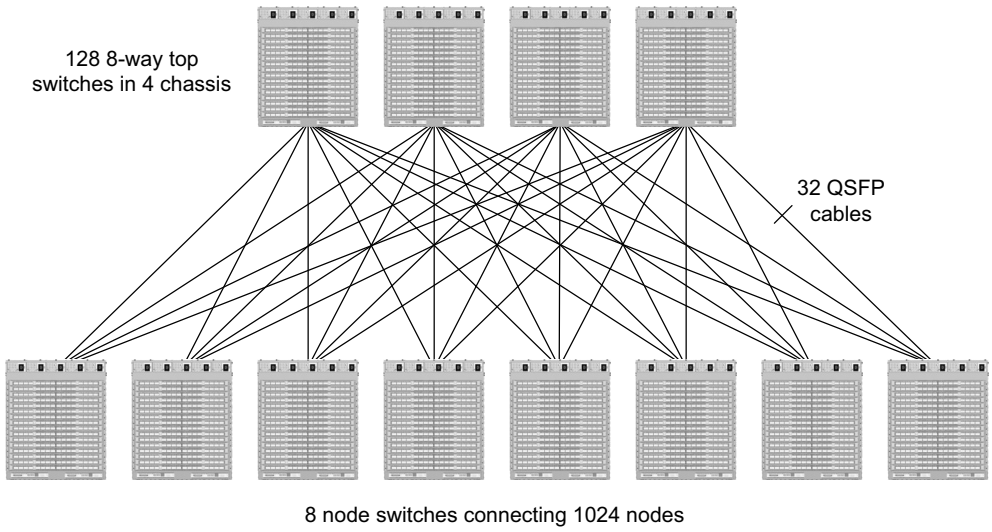

Fig. 5. Construction of a 1024-way QsNet^{III} network using 8 node switches and 128 $\times$ 8-way top switches, housed in 4 top switch chassis.

TABLE I
NETWORK SIZES AND COMPONENT COUNTS

Nodes	Node Switch Chassis	Cables	Top Switch Chassis
128	1	128	0
256	2	384	0
512	4	1024	2
640	5	1280	3
1024	8	2048	4
2048	16	4096	8
4096	32	8192	16
8192	64	16384	64
16384	128	32576	128

The choice of QSFP connectors for QsNet^{III} allows copper, active copper or optical cables to be used interchangeably. The choice of cable type can be made on grounds of length, bit-error-rate and cost. We might, for example use short copper cables on links between the nodes and the node switches and longer optical links on the links between the switches.

In a multi-rail system the network is replicated. Two networks can be connected to one adapter in each node (using both links) or to two different adapters. The performance of the host interface and the requirements of the application (bandwidth or issue rate) will determine which configuration is more appropriate.

A single top switch chassis provides 8 $\times$ 32-way top switches for a system with up to 4096 nodes. Larger networks can be constructed, up to 16384 nodes, using the node switch design to provide 64-way or 128-way top switches. These 4 stage configurations require equal numbers of node switch and top switch chassis (see table I). However, it is more likely that systems of this size will be constructed by integrating switches with the nodes, in a blade chassis for example. A 2048-way network can be constructed by connecting 128 $\times$ 16-way node switches, one in each blade chassis, with 16 $\times$ 128-way top switches. This approach has the advantage of reducing the number of cables required to build a constant bisection bandwidth network to one per node. With a larger switch chassis, 1024-way for example, this approach can be extended to system of 16384 blade servers.

QsNet^{III} components synchronise to a common clock distributed over the links. The regular structure of the network makes computing a spanning tree straightforward; the lowest numbered powered up switch in the highest stage is used. In the initial version of the firmware hardware broadcasts use the clock tree[5]; both functions move to a new top switch in the event of failure.

QsNet^{III} switches are managed out-of-band. Each switch chassis has a pair of embedded controllers. Each switch card has an embedded Freescale CPU managing one or more Elites. Switch cards are connected to the controllers via Ethernet. The controllers use I2C to monitor power supplies, fans and temperature. Switches chassis are connected together and to the system's management network via external Gbit Ethernet interfaces on each controller. Controllers and embedded CPUs run UBoot and a cut down Linux distribution.

A subset of management functions are common to QsNet^{III} and our 10 Gbit/s Ethernet switches; system responsiveness, environmental monitoring, link status, link error monitoring etc. Other functionality including clock tree determination, link connectivity checking and routing table setup is QsNet^{III} specific.

In QsNet^{II} systems links are used as soon as they are

[5]We plan to experiment with a more sophisticated routing scheme in which broadcasts and barriers occur on multiple trees but this is not supported in the current firmware.

connected, however they are connected. This approach is convenient, but not in tune with the requirements of maintaining a large production network in which links are brought in and out of use while the system is running[6]. On QsNet^{III} we use a three step link bring up process. First, we negotiate link layer connectivity, speed and protocol (CX4 or QsNet). Second we check that the link is connected to the right place, and third we test the link's ability to transfer data reliably. Elite5 has the ability to generate pseudo random packets at full line rate. We use this in the third step of link bring up. Each Elite sends and receives packets on the newly connected link. The embedded controller monitors the data rate and number of errors. If all three steps of the link bring up process complete successfully the link is added to the route tables. This approach to system maintenance and diagnostics removes the need to run tests on the nodes that interfere with the machine's workload.

If a link is disconnected, or if the error rate is unacceptably high, it is removed from the routing tables. Low rates of errors are tolerated within the network where there are multiple redundant routes. Relatively high error rates are tolerated on single links to the nodes as disconnecting them would disconnect the node. The adaptive routing hardware in each switch automatically avoids disconnected links.

VI. Comparison with other networks

QsNet^{III} is designed for high end HPC users with communication intensive applications using most or all of a large machine for a single job. QsNet^{III} has faster links that QsNet^{II} 25Gbit/s in each direction rather than 9Gbit/s total. QsNet^{III} has increased radix routers, halving the number of switches required to build a system. QsNet^{III} uses commodity connectors and cables, increasing the choice of components that can be used and reducing their cost. QsNet^{III} provides enhanced support for small packets, increasing the number of multi-core commodity nodes that can be used to execute a job efficiently.

The principle difference between QsNet^{III} and networks such as Infiniband, Ethernet or Myrinet is in the use of packet-by-packet adaptive routing. Our approach allows the hardware to dynamically find the free paths in a multi-stage network. Point-to-point bandwidths are similar. The structure of the QsNet^{III} network ensures that point-to-point bandwidths can be sustained when large numbers of nodes use the network at the same time, even for non-local communication patterns. QsNet^{III} latencies are lower (as are those for QsNet^{II}) both in the adapter and the routers. QsNet^{III} also provides hardware support for barrier, broadcast and offloaded collectives, essential features of a large HPC network provided only QsNet and the IBM Blue Gene network [25].

[6]A large federated network has two sets of cables. In general short cables are used between the nodes and the node switches and longer cables between the node and top switches. Installation and maintenance operations on the nodes and their link cables are straightforward. Installation and replacement of switch cards and the cables between the switches can result in mis-cabling. On QsNet^{II} systems the process of determining connectivity is manual. On QsNet^{III} it is an automatic part of link bring up.

VII. Future work

Our current work is focused on delivery of the first large QsNet^{III} systems. These machines use a common set of components described above. With this work complete we will introduce a low cost entry level switch and a single chassis high port count switch.

On QsNet^{II} we introduced a number of optimised collectives [7], [24] in which execution was offloaded to the network adapter. We will develop this to provide a general framework for non-blocking offloaded collectives on QsNet^{III} .

QsNet^{II} and QsNet^{III} support multi-rail systems, in which multiple links from the same node connect to distinct networks. We will also implement multi-port QsNet^{III} systems in which multiple links from the same node connect to a single network. This technique allows us to be more flexible in matching the communications resources of a node to its other characteristics, For example a large memory, high CPU count node might have 2 or 4 connections to the same network as a large number of commodity nodes with one connection each.

The use of standard link formats allows QsNet^{III} adapters to implement other network protocols, notably 10Gbit/s Ethernet. We are developing firmware and drivers for this aspect of the product, broadening its applicability.

VIII. Conclusion

QsNet^{III} provides a high performance data network for the most demanding HPC applications. We have increased link speeds significantly from QsNet^{II} and have enhanced both adapter and switch functionality. The use of high radix routers reduces the number of components required to build a network and hence its cost and the complexity of routing data across it. The use of packet-by-packet adaptive routing underwrites scalability of complex communications patterns on large numbers of nodes. Features such as hardware barrier and broadcast enhance the applicability to high end HPC applications.

Acknowledgment

The authors would like to thank Jon Beecroft, David Hewson and Moray McLaren for their invaluable contributions to the design of QsNet^{III} . We would also like to thank Dan Kidger for proof reading this paper.

References

[1] Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg and Adolfy Hoisie, *Hardware - and Software Based Collective Communication on the Quadrics Network*, in Proceedings of the 2001 IEEE International Symposium on Network Computing and Applications (NCA 2001) Cambridge, Mass, October 8-10, 2001.

[2] David Addison, Jon Beecroft, David Hewson, Moray McLaren and Fabrizio Petrini, *Quadrics QsNet^{II} : A network for Supercomputing Applications*, in Hot Chips 15, Stanford University, CA, August 2003

[3] Jon Beecroft, David Addison, David Hewson, Moray McLaren, Duncan Roweth, Fabrizio Petrini and Jarek Nieplocha, *QsNetII: Defining High-Performance Network Design*, IEEE micro July/August 2005, vol 25, No. 4 pg 34-47

[4] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali, *Adaptive source routing in multistage interconnection networks*, in IPPS '96: Proceedings of the 10th International Parallel Processing Symposium, pg 258-267, Honolulu, HW, 1996. IEEE Computer Society.

[5] Ryan L. Braby, Jim E. Garlick, and Robin J. Goldstone, *Achieving order through CHAOS: the LLNL HPC Linux Cluster Experience*, available from http://computing.llnl.gov/linux/publications.html

[6] Jon Beecroft, Mark Homewood, Moray McLaren *Meiko CS-2 interconnect Elan-Elite design*, Parallel Computing Volume 20, Issue 10-11 (November 1994)

[7] Duncan Roweth and Ashley Pittman *Optimised Global Reduction on QsNet^{II}* , Hot Interconnects 2005: 23-28.

[8] Eitan Zahavi, Gregory Johnson, Darren J. Kerbyson, Michael Lang, *Optimized Infiniband Fat-tree Routing for Shift All-to-All Communication Pattern*, in Proceedings of the International Supercomputing Conference (ISC07), Dresden, Germany, June 2007.

[9] Maria E. Gomez, Jose Flich, Antonio Robles, Pedro Lopez, Jose Duato, *Evaluation of Routing Algorithms for Infiniband Networks (Research Note)*, Proceedings of the 8th International Euro-Par Conference on Parallel Processing Publisher Springer-Verlag London, UK ISBN:3-540-44049-6, 2002 pg 775 - 780,

[10] F. Petrini, J. Fernandez, E. Frachtenberg, and S. Coll, *Scalable collective communication on the ASCI Q machine*, in Hot Interconnects 12, Aug. 2003.

[11] Quadrics Ltd *Elan Programming Manual and other documentation*. Available from http://www.quadrics.com/documentation.

[12] Matt Leininger, Mark Seager, OpenFabrics Developers Workshop, Sonoma, CA, April 30 2007.

[13] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra, *MPI: The Complete Reference. The MIT Press, 1998, Volume 1, The MPI Core*, The MIT Press, Cambridge, Massachusetts, 2nd edition, September 1998 ISBN0-262-69215-5.

[14] Cray Inc *Cray Man Page Collection: Shared Memory Access (SHMEM)*, available from the Cray http://website www.cray.com/craydoc.

[15] J. Nieplocha and J. Ju, *ARMCI: A Portable Aggregate Remote Memory Copy Interface*, October 30, 2000, available from http://www.emsl.pnl.gov:2080/docs/parsoft/armci/armci1-1.pdf

[16] UPC Consortium, *UPC Language Specifications, v1.2*, Lawrence Berkeley National Lab Tech Report LBNL-59208, 2005.

[17] Numrich and Reid, *Co-Array Fortran for Parallel Programming*, ACM Fortran Forum 1998, vol 17, no 2, pg 1-31

[18] C. Leiserson, *Fat-trees: Universal networks for hardware efficient supercomputing*, IEEE Transactions on Computer, C-34(10):892-901, October 1985.

[19] C. Clos, *A Study of Non-Blocking Switching Networks*, The Bell System technical Journal, 32(2):406 424, March 1953.

[20] F. Petrini and M. Vanneschi, *k -ary n -trees: High performance networks for massively parallel architectures*, In IPPS '97: Proceedings of the 11th International Symposium on Parallel Processing, page 87, Geneva, Switzerland, 1997. IEEE Computer Society.

[21] C. Seitz, *Myrinet: A gigabit-per-second local area network*, In Hot Interconnects II, Stanford University, Stanford, CA, August 1994

[22] Fabrizio Petrini and Wu-chun Feng and Adolfy Hoisie and Salvador Coll and Eitan Frachtenberg, *The Quadrics Network: High Performance Clustering Technology*, IEEE Micro, 2002, vol 22 pg 46-57.

[23] Duncan Roweth and Mark Homewood, *The Elan5 Network Processor*, Presented at ISC07 in Dresden, available from http://www.quadrics.com.

[24] Duncan Roweth, and David Addison, *Optimised Gather Collectives on QsNet^{II}* , Presented at PVM/MPI 2005. Lecture notes in Computer Science vol 3666/3005 pg 407-414.

[25] G. Almasi, P. Heidleberger, C.J.Archer, X.Martorell, C.Erway, J.Moreira, B.Steinmacher-Burow, Y.Zheng *Optimisation of MPI collective communication on BlueGene/L systems*, Proceedings of the 18th annual international conference on Supercomputing, 2005, pg 253-262.