Scalable Ethernet Clos-Switches

Norbert Eicker[1] and Thomas Lippert[1,2]

[1] Central Institute for Applied Mathematics, John von Neumann Institute for Computing (NIC), Research Center Jülich, 52425 Jülich, Germany
{n.eicker, th.lippert}@fz-juelich.de
[2] Department C, Bergische Universität Wuppertal, 42097 Wuppertal, Germany

Abstract. Scalability of Cluster-Computers utilizing Gigabit-Ethernet as an interconnect is limited by the unavailability of scalable switches that provide full bisectional bandwidth. Clos' idea of connecting small crossbar-switches to a large, non-blocking crossbar – wide-spread in the field of high-performance networks – is not applicable in a straightforward manner to Ethernet fabrics. This paper presents techniques necessary to implement such large crossbar-switches based on available Gigabit-Ethernet technology. We point out the ability to build Gigabit-Ethernet crossbar switches of up to 1152 ports providing full bisectional bandwidth. The cost of our configuration is at about €125 per port, with an observed latency of less than 10μsec. We were able to find a bi-directional point-to-point throughput of 210 MB/s using the ParaStation Cluster middle-ware[2].

1 Introduction

Sophisticated software accelerators enable Gigabit-Ethernet[1] to act as an alternative in the field of interconnects for Cluster-Computing[2]. Since small- and medium-sized switches are available economically priced, this technology is able to serve as an inexpensive network for Clusters with up to ~ 64 nodes – as long as the communication requirements of the applications allow to disobey high-end technologies like Myrinet, InfiniBand, InfiniPath or Quadrics. Nevertheless, in this role Gigabit Ethernet suffers from the unavailability of large, reasonably priced switches. Thus, for large Clusters one either has to purchase one expensive monolithic switch providing full bisectional bandwidth or is forced to accept the handicap imposed by cascaded, medium-sized switches. The latter configuration is afflicted with decreasing accumulated bandwidth from stage to stage.

In the early 50's Clos already proposed a way out of this dilemma[3]. Originally in the field of telephony networks he suggested to set up a special topology of cascaded crossbar-switches providing full bisectional bandwidth. This idea is widespread in the field of high-performance networks. Actually this scheme is used by *e.g.* Myrinet or InfiniBand.

In order to solve the problem discussed above at least in principle, it is possible to use a similar setup with Gigabit Ethernet switches, too. Unfortunately, some specific features of the Ethernet protocol inhibit to actually exploit the

bandwidth provided by this topology to a large extent. This work will present a way out of this dilemma.

The abilities of the switch building-blocks play an essential role for the constructions of Ethernet Clos-switches. On the one hand, they have to support virtual LANs (VLAN)[9]. On the other hand, it is necessary to modify the switches routing tables on the level of MAC addresses. Switches fulfilling these conditions are usually called to be "level 2 manageable".

This paper is organized as follows: In the next section, Clos ideas are briefly reviewed. The following two sections discuss some essential features and extensions of the Ethernet protocol, namely spanning trees, VLANs and multiple spanning trees. Based on this fundament we will sketch the setup of cascaded Ethernet crossbar switches in section 5. This includes presenting our testbed and discussing the need for explicit routing tables. Section 6 displays the results produced using the testbed. We end with conclusions and give a brief outlook on further work done in the context of the ALiCEnext project in Wuppertal.

1.1 Related Work

The idea of extending the basic fat tree Ethernet topology is wide-spread. Nevertheless, all projects targeting the improvement of the accumulated bandwidth of a Cluster's Ethernet fabric do not implement full bisectional bandwidth. Instead, special network topologies are developed which are well suited for the communication pattern of a specific class of applications the corresponding Cluster is dedicated to. Good examples of this philosophy are the network of the McKenzie Cluster[12] dedicated to astrophysics or the flat neighborhood networks[13] of the KLAT2 and KASY0 machines used for computational fluid dynamics.

Another approach getting along without any switch is the ALiCEnext mesh network dedicated to lattice QCD[14]. This network only supports pure nearest neighbor communication.

All these concepts prove to be efficient only as long as communication patterns are used they were specifically tuned for. As soon as other applications with different communication patterns are involved, either the fabric has to be re-cabled or a significant performance penalty has to be accepted. In particular such concepts are not feasible for general purpose Clusters running various applications with diverse communication patterns, using different numbers of nodes, serving many users in parallel, etc.

The Viking project[11] pursues a concept similar to the one presented in this paper. *I.e.* it uses VLANs to create many independent spanning trees. Since the main target of Viking are metropolitan area networks, a static configuration like the one proposed in the present work is not feasible. Instead, they use so called node controller and manager instances in order to adapt the configuration of the Ethernet fabric dynamically to the constraints of the current hardware configuration and the needs of the actual communication load. Furthermore, since full bisectional bandwidth is not the main goal of the Viking project, the proposed network topology is not the one presented by Clos but a grid like.

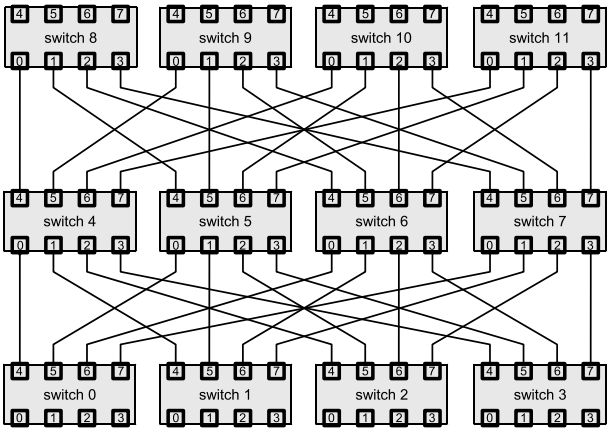

Fig. 1. Example of a full 3-stage Clos-network based on 8-port switches. The full hierarchical switch provides 8×4 ports with full bisectional bandwidth.

2 Clos-switches

In 1953 Clos[3] introduced the concept of multiple cascaded switches interconnected in a mesh like topology. Originally having telephone networks in mind the idea behind this topology was twofold: Make the network more fault tolerant, *i.e.* more robust in the case of the loss of one or more switches and increase the scalability of the accumulated bandwidth of such systems substantially.

At last, Clos' idea paved the way for multi-stage crossbar networks providing full bisectional bandwidth. The maximum size of a fully connected network is no longer limited by the number of ports offered by the biggest switch available. Of course, with increasing number of ports more switching hierarchies are necessary, each adding to the switching latency.

Today, actually all switched high performance networks (*e.g.* Myrinet[4], Quadrics[5] or InfiniBand[6]) make use of Clos' idea in order to provide full connectivity to large fabrics. This is necessary since the atomic crossbars available for these technologies typically offer not more than $\mathcal{O}(32)$ ports.

The basic topology of a 3-stage Clos-network is sketched in figure 1. It is easy to prove that at any level the same number of connections are available and full bisectional bandwidth is provided in this sense. In order to maximize the usable connectivity, an appropriate routing strategy has to be introduced. The main result of this work is a technique devising a routing strategy for a hierarchy of Gigabit-Ethernet switches. Furthermore figure 1 serves as an introduction of some terminology used in the course of this work:

- Switches connected to nodes are called *level-1 (or L1) switches*. In the example of figure 1 these are the switches 0, 1, 2, 3, 8, 9, 10 and 11.
- Switches connecting level-1 switches are called *level-2 (or L2) switches*. Switches 4, 5, 6 and 7 are the example's L2 switches.

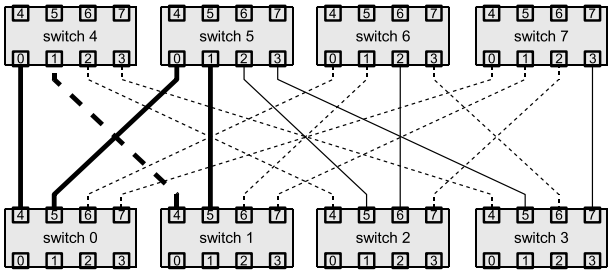

Fig. 2. Basic loop appearing in Clos-switch topologies marked by fat lines. In order to suppress such loops, STAs will switch off the dashed links.

3 Spanning Trees

Trying to construct Clos' topology based on Ethernet technology, a first limitation origins in the use of spanning trees. These are needed to avoid loops within the network fabric. While this feature is inevitable for an Ethernet fabric to work at all, it prevents from using the multiple paths between two switches in parallel. This leads to an accumulated bandwidth found to be identical to the one delivered by cascaded switches.

The very importance of the absence of loops within an Ethernet fabric lies in the fact of missing restricted lifetimes of packets on Ethernet level. This will enable packets to live forever, if the routing-information within the switches creates loops due to misconfiguration. Furthermore – even if the routing is set up correctly – the existence of broadcast packets within the Ethernet protocol provokes packet storms inside the fabric: Whenever a switch receives a broadcast packet on a given port, this packet will be forwarded to all other ports irrespective of available routing information. If there are at least two connections between two switches, a broadcast package sent from one switch to another via a first connection will be sent back to the originating one using the second connection. Once the first switch is reached again, the packet will be sent on its original way again and a loop is created.

Unfortunately, Ethernet broadcast packets play a prominent role within the Internet protocol family, since ARP messages at least on Ethernet hardware are implemented using this type of communication[7]. Thus, every time the MAC-address corresponding to a destination's IP address is unknown, broadcast messages are sent on Ethernet level. Consequently, it is almost impossible to prevent this kind of Ethernet packets in practice.

To beware an Ethernet fabric of this vulnerability, spanning trees were introduced[8]. The main idea here is the detection of loops within a given network fabric and the selective deactivation of such connections, which would eventually close loops. Unfortunately, this will happen on a quite fundamental level of the switch's functioning and thus prevent this link from carrying any data at all.

Investigating Clos-switch topology one can find loops even within the simplest example. Figure 2 sketches one loop in a setup of 2×4 switches[3]. In fact, even this simple setup hosts many loops each of them preventing it from working correctly. On the other hand, the spanning tree algorithm detects these loops and – at the same time – disables all the additionally introduced bandwidth. In figure 2 all connections deactivated by an assumed spanning tree algorithm (STA) are depicted as dashed lines.

4 Virtual LANs and Multiple Spanning Trees

Particularly important for our purposes is the concept of virtual local area networks (VLAN)[9]. This implements multiple virtually disjunct local area networks (LAN) on top of a common Ethernet hardware layer. In order to realize this feature a new level of indirection is introduced by explicitely tagging every native Ethernet packet – including broadcasts – as a member of a distinct VLAN.

This creates a twofold benefit: The topology of the network fabric can be rearranged remotely just by reconfiguration of the switches without physically touching any hardware at all. Additionally – if supported by the operating system – it is possible to assign a given computer to different VLANs at the same time without the need of extra communication hardware.

This technology is widely used in order to map a company's organization virtually onto an uniform physical network fabric in a very flexible way. Thus, it is not surprising that many so-called department switches support this feature.

In this context the idea of spanning trees has to be extended. The reason for providing each VLAN with its own spanning tree is threefold:

- For security reasons broadcast messages have to be restricted to the VLAN they were created in. Otherwise, depending on the high-level protocol[4] the possibility is given to spoof data between different VLANs.
- Within each VLAN there might be loops. These loops would compromise the functionality of the fabric as a whole if they are not eliminated.
- As long as a physical connection is available, the connectivity within each VLAN has to be guaranteed, even if the different VLANs as a whole would build loops. Configurations with physical connections inevitable for the correct functioning of one VLAN but closing a loop within another can easily be constructed. Such dichotomy can only be cured by spanning trees assigned to each VLAN separately.

In order to meet this constraints the STA discussed above was extended to the concept of multiple spanning trees (MST)[10].

[3] Actually, loops already appear in 2×2 setups. Since figure 2 also sketches the effects of STAs, the 2×4 setup was chosen.

[4] Here everything above Ethernet protocol level is seen as high-level.

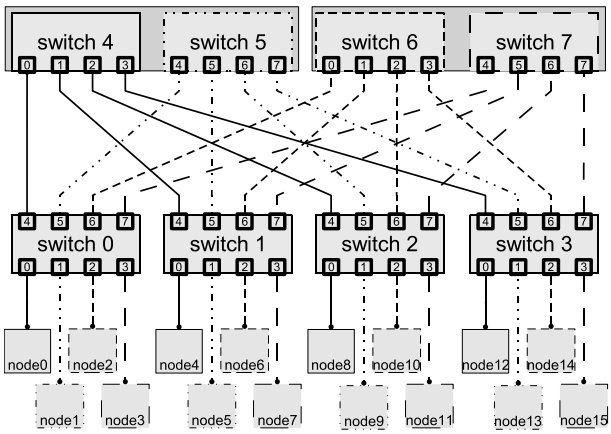

Fig. 3. Crossbar configuration with virtual switches. Each VLAN is depicted with a different line-mode. Lines connecting nodes and L1 switches are only in node → switch direction exclusively used by one VLAN; in switch → node direction each link is used by any VLAN.

5 Configuration setup

Putting together the technologies described in the last sections the problem of loops can be eliminated without loosing the additional bandwidth of the Clos-topology at the same time. We proceed as follows:

- Setup various VLANs, each forming a spanning tree.
- As many VLANs as nodes attached to a single L1 switch are needed. This fixes the number of VLANs to half the number of ports a switch provides.
- Configure node-ports (*i.e.* ports with nodes attached) to use – depending on the port – a specific VLAN whenever receiving inbound traffic. This implements the required traffic shaping.
- Configure all the node-ports to send outbound traffic from every VLAN[5]. *I.e.* data from every VLAN (and thus from every node) can be sent to any other node, irrespective of the VLAN the sending node is mapped to.

It is essential that all traffic sent from switches directly to a node is not spoiled by any VLAN information[6]. Hence, from the nodes' point of view the complex network topology is completely transparent and no modification has to be done to the nodes' configuration. Even nodes not supporting the VLAN technology at all can be used within this setup.

Figure 3 sketches the typical setup of the crossbar configuration. Here VLANs are depicted by line styles, *i.e.* L2 switches with a distinct border only carry

[5] This "port overlapping" is a feature marked as optional in the VLAN standard[9].

[6] This might introduce additional overhead on the L1 switches; in practice this proved to be negligible.

traffic sent by nodes with the same border into the corresponding VLAN. On the other hand, nodes receive data irrespectively of the sending node's VLAN. Traffic shaping is implemented as follows:

- Traffic sent from one node to another connected to the same L1 switch does not touch any other switch. *E.g.* `node6` will talk directly to `node4`.
- The sending node's switch port chooses the L2 switch used to emit data to other L1 switches. This ensures efficient use of the whole fabric.

Assuming `node0` to `node3` try to send data to the nodes connected to `switch2` concurrently, `node0` will send via `switch4`, `node1` via `switch5`, *etc.* Hence, there are 4 independent routes between any two L1 switches in the example. This assures the bisectional bandwidth of the setup.

Figure 3 shows an additional detail. `switch4` and `switch5` are only logical and assumed to share the same hardware. Because of 4 L1 switches only 4 ports of a logical L2 switch are occupied. Thus, another logical switch can use the remaining 4 ports. Again, the configuration is realized via the VLAN mechanism. This guarantees the absence of data-exchange between ports of a physical L2 switch dedicated to different VLANs. Since both – logical and physical L2 switches – have to handle the VLANs anyhow no further effort is introduced.

5.1 The ALiCEnext testbed

Our testbed used to implement this configuration consists of 144 dual-Opteron nodes of the ALiCEnext[15] Cluster located at Wuppertal University. They are connected via 10 SMC 8648T Gigabit-Ethernet switches[16]. Providing 48 port, each L1 switch serves 24 nodes leading to 24 VLANs. Thus, 6 L1 switches are needed. The other 24 ports are connected to the 4 remaining switches. Every L1 switch is connected via 6 lines to each of the 4 L2 switches. Thus, each physical L2 switch hosts 6 virtual L2 switches leading to a total of 24 logical ones as implied by the number of VLANs and the number of nodes connected to each L1 switch[7].

Unfortunately, at least the implementation of the MST algorithm the SMC 8648T provides is not robust enough to detect the – admittedly very special – setup of our Clos-switch topology correctly. In fact, the switches locked up and the network was unusable[8]. Consequently, automatic loop detection and elimination provided by the MST mechanism was switched off explicitely. This enforces special care when setting up VLANs and cabling. In particular, the default VLAN which includes all ports of the different switches and thus contains countless loops has to be eliminated from the fabric.

[7] In principle 3 L2 switches are sufficient to build a 144 port crossbar fabric. The extra ports in our setup were used to implement a connection to the outside world.

[8] Interestingly, plugging the 144 cables between L1 and L2 switches one after the other worked out. This leads to the assumption that switches cannot handle the flood of Hello BPDUs. Of course, plugging all cables whenever a switch restarts is no viable solution.

With this a first prove of concept was obtained by assuring that complete connectivity between the nodes is seen: pings were send from every node to any other node. Furthermore services like ARP[9] and multicasts work out of the box.

Nevertheless, a detailed investigation of the fabric unveiled a problem buried deeper in this setup. In fact, communication between nodes attached to the same VLAN, *i.e.* connected to the same port number of different switches, worked as expected. Communication from one VLAN to another worked in principle, too, but we observed significantly reduced performance.

5.2 Routing tables

Investigating the dynamic routing tables of the L2 switches the problem was disclosed. These are created on the fly while listening to network traffic between the nodes. Since all inbound traffic is sent via specific VLANs, a L2 switch will never see traffic of nodes sending into different VLANs. In figure 3 *e.g.* `switch5` will never see any traffic from `node0`. On receiving traffic addressed to yet unknown nodes, switches will start to broadcast to all ports. This introduces a plethora of useless traffic. Due to congestion this leads to packet loss and significantly reduced throughput.

To prevent congestion one has to harness the switches with static routing tables. They will shape the traffic addressed to a distinct node in a given VLAN to a specific port. One has to keep in mind that the size of such routing tables is proportional to both, the number of VLANs and the number of nodes connected to the fabric. Thus the tables needed for the testbed will have $24 \times 144 = 3456$ entries. Correspondingly, the routing tables of the entire ALiCEnext machine (528 ports) will contain 12672 entries. Switches providing such numbers of static entries are available; *e.g.* the SMC 8648T allows up to 16k entries[10].

The sheer number of routing entries entail that programming the switches cannot be done in a manual way. Instead, we wrote programs that collect the required data and create the corresponding configuration files. Furthermore, the collected data allow some automatic debugging of the cabling between the nodes and the L1 switches as well. The deployment of configuration files to the switches is automated, too.

6 Results

First we determined the basic parameters of the involved building-blocks. The measurements were carried out on two nodes of the ALiCEnext Cluster with their Gigabit-Ethernet ports directly connected, *i.e.* no switch in between. The very efficient ParaStation protocol was used in order to reduce the message la-

[9] Since ARP is based upon Ethernet broadcasts these work, too. Interestingly, request and response might use different VLANs and, thus, different spanning trees.

[10] This limits scalability for this building-blocks to actually ~ 680 nodes.

Table 1. Performance results

	Back-to-back	single switch	3-stage crossbar
Throughput / node [MB/s]	214.3	210.2	210.4
Latency [μsec]	18.6	21.5	28.0

tencies as much as possible[11]. As a high-level benchmark the Pallas MPI Benchmark suite (PMB)[17] was employed. We applied two tests, `pingpong` for latency determination and `sendrecv` for bandwidth measurements.

Performance numbers for directly connected ports are in the left column of table 1. Latency is half-round-trip time of 0 byte messages as determined by `pingpong`. The low latency found is due to the ParaStation protocol[12]. Throughput is for 512 kByte messages. Larger messages give slightly less throughput ($\sim$ 200MB/sec) due to cache effects when accessing the main memory. The message size for half-throughput was found to be 4096 Byte for all tests.

To determine the influence of a single switch stage the benchmark was repeated using two nodes connected to the same switch. The corresponding results are marked as "single switch" in table 1. Obviously, there is almost no influence of the switch on the throughput. Since the total latency rises from 18.6μsec to 21.5μsec, each switch stage is expected to introduce an additional latency of 2.9μsec. We anticipate a total latency of $\sim$ 27.5μsec when sending messages through all three stages of the testbed. This corresponds to a latency of $\sim$ 9μsec from the switch alone. Throughput is expected to be unaffected.

The above tests were done using a single pair of processes. In order to show bisectional bandwidth we have to concurrently employ as many pairs as possible. Furthermore, the processes have to be distributed in a way that communicating partners are connected to disjoint L1 switches, forcing all traffic to go via the L2 switches and stress the fabric to the hilt. At the time we ran our benchmarks 140 processors were accessible to us, leading to 70 pairs.

The numbers for the 3-stage crossbar presented in table 1 are worst case number. *I.e.* the result for the pair showing the least throughput is displayed there. Looking at the average value of all pairs, throughput is $\sim$ 5% bigger. The best performing pair even gives a result of $\sim 218 MB/sec$. The total throughput observed is larger than 15 GB/s.

Based on an observed latency of 28.0μsec the latency actually introduced by the crossbar-switch was found to be 9.4μsec, *i.e.* in the expected range. This is well below the numbers available for many big, monolithic Gigabit-Ethernet switches with full bisectional bandwidth – at a much lower price! We expect this number to be constant up to 1152 ports[13].

[11] ParaStation uses a fine-tuned high-performance protocol in order to reduce the overhead of general-purpose protocols like TCP.

[12] On the same hardware a fine-tuned TCP-setup will reach about 28μsec on MPI-level; out of the box the MPI latency over TCP is often in the range of $60 - 100\mu$sec.

[13] Which is a theoretical limit since size of routing tables restricts us to $\sim$ 680 ports.

7 Conclusion and Outlook

We presented a new way to set up a scalable crossbar switch based on of-the-shelf Gigabit-Ethernet technology. The switch itself is completely transparent to the node-machines. Using the ALiCEnext Cluster at Wuppertal University we showed the concept to work as expected. Full bisectional bandwidth could be achieved at a price of less than €125 per port[14] even with more expensive level 2 manageable switches[15]. In this work we demonstrated our concept to actually work for 144 ports[16].

We submitted an international patent for our approach which is pending[18].

Acknowledgments. We thank the ALiCEnext team in Wuppertal for patience and kind support.

References

1. IEEE standard 802.3z, IEEE standard 802.3ab
2. ParaStation (http://www.par-tec.com).
3. Charles Clos, "A Study of Non-blocking Switching Networks", The Bell System Technical Journal, 1953, vol. 32, no. 2, pp. 406-424
4. http://www.myri.com
5. http://www.quadrics.com
6. http://www.infinibandta.org
7. David C. Plumme, "An Ethernet Address Resolution Protocol" RFC 826, November 1982
8. IEEE standard 802.1D
9. IEEE standard 802.1Q, IEEE standard 802.3ac
10. IEEE standard 802.1s
11. S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh, "Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks" in IEEE INFOCOM, 2004.
12. J. Dubinski, R. Humble, U.-L. Pen, C. Loken, and P. Martin, "High Performance Commodity Networking in a 512-CPU Teraflops Beowulf Cluster for Computational Astrophysics", eprint arXiv:astro-ph/0305109, Paper submitted to the SC2003 conference.
13. http://aggregate.org/FNN/ and http://aggregate.org/SFNN/
14. Z. Fodor, S.D. Katz, G. Papp, "Better than $1/Mflops sustained: a scalable PC-based parallel computer for lattice QCD", Comput. Phys. Commun. 152 (2003) 121-134.
15. http://www.alicenext.uni-wuppertal.de
16. http://www.smc.com
17. Pallas MPI Benchmark now available from Intel as Intel MPI Benchmark (IMB) http://www.intel.com/software/products/cluster/mpi/mpi_benchmarks_lic.htm
18. Patent: "Data Communication System and Method", EP 05 012 567.3.

[14] Based on prices found at `http://www.pricewatch.com`.

[15] The use of more simple (and cheaper) switches lacking the possibility of defining routing tables is not feasible due to congestion.

[16] In the meantime the ALiCEnext crossbar was scaled up without problems. Now it uses 34 switches (22 L1 and 12 L2) to support 528 ports.