A Scalable Ethernet Clos-Switch

Norbert Eicker and Thomas Lippert

published in

NIC Symposium 2006 ,
G. Münster, D. Wolf, M. Kremer (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **32**, ISBN 3-00-017351-X, pp. 307-314, 2006.

A Scalable Ethernet Clos-Switch

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
E-mail: {n.eicker, th.lippert}@fz-juelich.de

[2] Department C, Bergische Universität Wuppertal, 42097 Wuppertal, Germany

The Scalability of Cluster Computers with Gigabit-Ethernet interconnect is limited by the unavailability of scalable Gigabit-Ethernet switches that can achieve full bisectional bandwidth. Clos' idea of connecting small crossbar-switches to a large, non-blocking crossbar is not applicable in a straight-forward manner on Ethernet fabrics. This paper presents techniques to realize large crossbar-switches built up on standard Gigabit-Ethernet switches as components. We show how to build Gigabit-Ethernet crossbar switches with up to 1152 ports, achieving full bisectional bandwidth at a cost of about € 125 per port. The latency of our Clos-switch is less than 10 μsec. These numbers are superior to any monolithic switch on the market. Using the ParaStation cluster middle-ware[2], a full bisectional bandwidth is achieved and a bi-directional point-to-point throughput of 220 MB/s is found.

1 Introduction

Sophisticated software accelerators render Gigabit-Ethernet[1] a true alternative as an interconnect for Cluster Computers. Since small- and medium-sized switches are available at attractive prices, this technology is able to serve as an inexpensive network for clusters with up to 64 nodes—as long as the applications do not require high-end communication technologies like *e.g.* Infiniband[6]. However, Gigabit Ethernet networking technology suffers from the unavailability of larger, reasonably priced switches. In order to build larger clusters one either has to purchase an expensive monolithic switch or one is forced to use a tree of cascaded switches with decreasing accumulated bandwidth from stage to stage.

A way out of this dilemma was proposed by Clos in the early 50's[3] in the field of telephony-networks by setting up a special topology of cascaded crossbar-switches providing full bisectional bandwidth; this scheme is used for instance by Myrinet[4] or Infiniband[6].

In principle it is also possible to build a similar setup with Gigabit Ethernet switches. Unfortunately, some specific features of the Ethernet protocol to a large extent inhibit the exploitation of the bandwidth offered by this topology.

This paper shows how to solve the problem. The capabilities of the building-blocks play an essential role for the construction of efficient Ethernet Clos-switches. On the one hand, they have to be able to support virtual LANs (VLAN)[9]. On the other hand, it must be possible to perform a modification of the routing tables on the MAC level. Switches that fulfill these conditions often are characterized as "level 2 manageable".

The paper is organized as follows: In the next section, the concept of Clos-networks is briefly reviewed. Sections 3 and 4 discuss spanning trees, virtual LANs and multiple spanning trees, followed by a sketch of the the setup of cascaded Ethernet crossbar switches. In section 6, we present the testbed used for prove of concept and give results in section 8. We conclude and give a short outlook on further work to be done in the context of the ALiCEnext project at Wuppertal University.

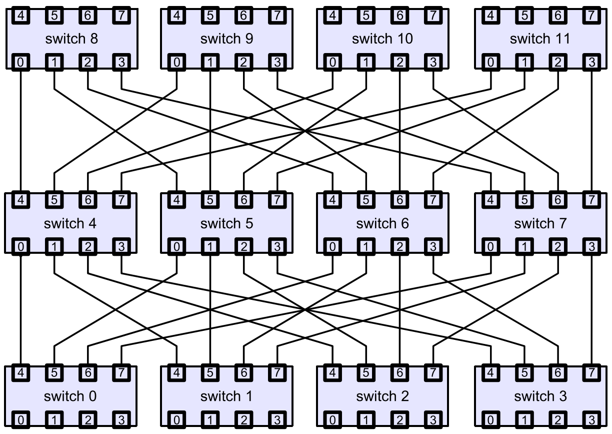

Figure 1. Example of a full 3-stage Clos-network based on 8-port switches. The full hierarchical switch provides 8×4 ports with full bisectional bandwidth.

2 Clos-Switches

In 1953 Clos[3] introduced the idea of multiple cascaded switches interconnected in a mesh-like topology originally addressing telephone networks. The main idea behind this topology was twofold: On the one hand, it should render the network more fault tolerant, *i.e.* more robust in case of failure of one or more switches. On the other hand, scalability of the accumulated bandwidth of such systems is by far superior than that of centralized systems.

Eventually, Clos' topology paved the way to set up multi-stage crossbar networks with full bisectional bandwidth. The maximum size of a fully connected network is no longer limited by the numbers of ports offered by the biggest switch available. With increasing numbers of ports more switch levels are necessary, each adding to the latency.

Today, all available switched high performance networks (*e.g.* Myrinet[4], Quadrics[5] or InfiniBand[6]) make use of Clos' idea in order to provide a full connectivity for large networks. This is necessary since the switch boxes available with 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 show that at any level the same number of connections is provided and that full bisectional bandwidth is guaranteed in this manner. In order to make maximal usage of the connectivity, an appropriate routing strategy has to be introduced. The main result of the present work is a technique devising such routing on a hierarchy of Gigabit-Ethernet switches. Furthermore, figure 1 can serve us to introduce the nomenclature used in the following:

- Switches connected to nodes are called *level-1 switches*. In figure 1, these are the switches 0, 1, 2, 3, 8, 9, 10 and 11.

- Switches connecting level-1 switches only are called *level-2 switches*. Hence, switches 4, 5, 6 and 7 are the example's level-2 switches.

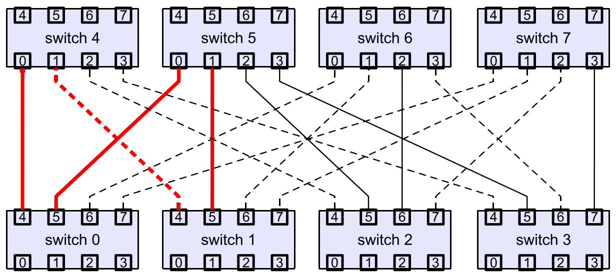

Figure 2. Basic loop appearing in Clos-switch topologies marked red. In order to suppress such loops, STAs will switch off the dashed links.

3 Spanning Trees

The major problem setting up a Clos-switch topology via Ethernet technology lies in the fact that Ethernet switches build spanning trees in order to avoid closed loops within the network. While this feature is mandatory for an Ethernet fabric to function at all, it prevents the parallel exploitation of more than one path between two switches. Therefore, the accumulated bandwidth of the network is the same as the bandwidth of cascaded switches.

The very importance of spanning trees to avoid loops within an Ethernet fabric is due to the fact that, on the Ethernet level, packets don't have a restricted life time. On the one hand, this will enable packets to live forever, if the routing-information within the switches creates loops due to misconfiguration. On the other hand—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, it will forward this packet to all other ports irrespective of any available routing information. *E.g.*, if there are two connections between two switches, a broadcast package sent from one switch to another via one of the connections will be sent back to its originating switch via the second connection. Once the originating switch is reached again, the packet will be sent along its former way once more, and a loop is created.

Unfortunately, Ethernet broadcast packets play a very important role within the Internet protocol family, since ARP messages on Ethernet hardware are implemented using this type of communication[7]. Every time the MAC-address corresponding to a destination's IP address is unknown, broadcast messages are sent on Ethernet level.

In order to avoid this extreme vulnerability of the Ethernet concept, spanning trees were introduced[8]. The main idea behind this concept is the detection of loops within a given network fabric and the selective deactivation of such connections which would close possible loops. Unfortunately, the deactivation happens on a quite fundamental level of the switch's functioning and thus it does not allow the link to carry any data at all.

For a Clos-switch topology one can find closed loops even for the simplest possible example. Figure 2 sketches a loop in a setup of 2×4 switches[a]. Many loops are found preventing the fabric from working correctly.

[a] In fact, closed loops already appear in 2×2 setups. Since figure 2 also illustrates the effects of STAs, the 2×4 setup was chosen.

4 Virtual LANs and Multiple Spanning Trees

In the context of Ethernet fabrics the concept of virtual local area networks (VLAN)[9] is particularly important for our purposes. The main idea is to implement multiple, virtually disjunct local area networks (LAN) on top of a common hardware layer. This is realized through an additional level of indirection marking every native Ethernet packet with a tag identifying the VLAN it belongs to. The benefit is twofold: it is possible to rearrange the topology of the network's fabric just by reconfiguration of the switches without touching any hardware physically. Furthermore, with the support of the operating system, it is possible to assign a computer to different virtual networks via a single network interface card. As this technology is very useful for mapping a company's organization virtually onto a single physical network fabric it is available in many so-called department switches.

For VLANs, the idea of spanning trees has to be adapted. Every VLAN requires its own spanning tree for three reasons:

- For the sake of security, broadcast messages shall only be visible within the VLAN they where created. Otherwise, as far as the the high-level protocol is concerned[b], it would be possible to spoof data transmitted within one VLAN from another.

- Within each VLAN there might be loops. The loops would compromise the functionality of the fabric as a whole if they are not eliminated.

- Even if the various VLANs as a whole might build loops, the connectivity within each separate VLAN has to be guaranteed as long as a physical connection is available. It is possible to encounter situations where a connection has to be shut down for a given VLAN but is mandatory for the correct functioning of another VLAN. This dichotomy can only be cured by spanning trees separately assigned to each VLAN.

In order to meet these needs of the VLAN technology, the spanning tree algorithm discussed above has been extended to the concept of multiple spanning trees (MST). Like the STA, the MST is standardized[10].

In practice, this mechanism does not seem to help, however. For the experimental setup described below, we found that the implementation of the MST algorithm we have tested is not robust enough to detect the—admittedly very special—setup of our Ethernet Clos-switch correctly. In fact, the switches locked up and the network was no longer usable.

We came to the conclusion, that one has to be very careful when setting up a Clos-network based on Ethernet technology. One has to avoid any loop within the various VLANs, because the automatic loop detection and elimination provided by the MST mechanism has to be be switched off explicitely. In particular, the default VLAN which usually includes all the ports of the different switches and thus contains many loops has to be eliminated from the fabric.

5 Ethernet Clos-Network Configuration

Putting together the technologies described in the last sections it is possible to avoid the problem of packet-flooding by loops in Ethernet fabrics. We proceed as follows:

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

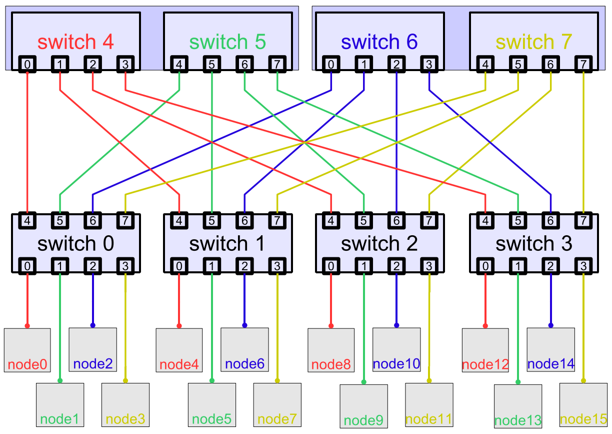

Figure 3. Crossbar configuration with virtual switches. Each VLAN is depicted with a different color. The lines between nodes and level-1 switches are only in node $\longrightarrow$ switch direction used exclusively by one VLAN; in switch $\longrightarrow$ node direction each link is used by all VLANs.

- Various VLANs are configured, each forming a spanning tree.
- In total, as many VLANs are needed as nodes are attached to a single level-1 switch.
- The node-ports (*i.e.* ports with nodes attached) are configured to use a specific VLAN (depending on the port) whenever they receive inbound traffic. This implements the required traffic shaping.
- All the node-ports are configured to send outbound traffic from every VLAN. *I.e.* data from every VLAN (and thus from every node) can be sent to any other node, independent of the VLAN the sending node is mapped to.

It is essential that traffic sent from any switch directly to a node is not spoiled by VLAN information. Hence from the nodes' point of view the network is completely transparent and no modification of the configuration of the nodes has to be carried out.

Figure 3 sketches the setup of the crossbar configuration. Here VLANs are depicted by the same colors, *i.e.*, colored switches only carry traffic sent by nodes of the same color into the corresponding VLAN. On the other hand, nodes receive data irrespectively of the sending node's color. The traffic shaping is implemented as follows:

- Traffic sent from a node to another one connected to the same level-1 switch does not touch any other switch. Thus `node6` will talk directly to `node4`.
- With respect to the color of a node the level-2 switch of the same color will be used in order to talk to nodes connected to other level-1 switches. This ensures the efficient usage of the complete network fabric.

Lets assume the machines `node0` to `node3` try to concurrently send data to the nodes on `switch2`. The traffic from `node0` will be sent via `switch4`, `node1` will use `switch5`, *etc.* Hence there are 4 independent routes between any two level-1 switches in this example. Thus, the full bisectional bandwidth of the setup is guaranteed.

Furthermore, figure 3 shows another important detail. In this setup, `switch4` and `switch5` are only virtual switches, *i.e.* they are assumed to use the same hardware. As only 4 ports of a virtual level-2 switch are occupied, one virtual switch can use the first 4 ports of a physical switch while another virtual switch can use the remaining 4 ports. Again the configuration is realized via the VLAN mechanism. It guarantees that there is no exchange of data between ports of a level-2 switch dedicated to different VLANs.

Note that no further effort is introduced since both—virtual and physical level-2 switches—have to handle the corresponding VLAN anyhow. The same arguments hold for `switch6` and `switch7`.

6 Testbed ALiCEnext

Our testbed consists of 144 nodes of the ALiCEnext[11] cluster located at Wuppertal University, Germany. The dual-Opteron nodes are connected via 10 SMC 8648T Gigabit-Ethernet switches[12]. They are attached to 6 of the 48-port switches, and each level-1 switch serves 24 nodes. The other 24 ports are connected to the 4 remaining switches. Every level-1 switch is connected via 6 lines to each of the 4 level-2 switches. Therefore, each physical level-2 switch hosts 6 virtual ones with VLANs in total corresponding to the number of nodes connected to a level-1 switch[c]. This setup delivers the full bisectional bandwidth.

In a first experiment the fabric was configured in such a way that only 24 VLANs were created for all the switches. Unfortunately, this approach introduced major problems which resulted in a total inaccessibility of both the nodes and the switches. The main reason is that the switches were unable to find a stable MST setup with all the loops introduced by our cabling. The only solution we found was to expliciteley switch off the MST algorithm.

After switching off MST a first proof of concept was obtained by confirmation of complete connectivity between the nodes; therefore every node was pinged from any other node. Nevertheless, a detailed investigation of the fabric unveiled a deeper problem lurking 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. But while communication from one VLAN to another worked in principle, we observed a significantly reduced performance.

7 Routing Tables

A detailed investigation of the dynamic routing tables unveiled the underlying problem: they are created on the fly while the switch is listening to the network traffic between the nodes. Since all inbound traffic is sent via specific VLANs, most switches will never see traffic sent by a given node. For the example of figure 3, `switch5` will never see any traffic from `node0`. As soon as a switch receives a packet addressed to a node not yet in the routing tables, it will start to broadcast this packet to all ports. The broadcast introduces a plethora of useless traffic within the fabric, leading to packet loss and a significantly reduced performance.

[c]In principle, 3 level-2 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.

In order to prevent the switches from creating unnecessary traffic one has to harness them with static routing tables. These tables explicitly 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 (with 512 nodes) will contain 12288 entries. Fortunately, switches providing this number of static entries are available, *e.g.* the SMC 8648T switch allows up to 16k entries.

8 Results

First we determined the basic parameters of the building-blocks. These measurements where carried out using two nodes of the ALiCEnext cluster with their Gigabit-Ethernet ports connected by a twisted pair cable and no switch in between. We used the very efficient ParaStation protocol for low level communication in order to reduce the latencies as far as possible[d]. As a high-level benchmark the Pallas MPI Benchmark suite[13] (PMB) was employed. Two tests have been applied, `pingpong` to determine the latency and `sendrecv` for bandwidth measurements.

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

Table 1. PMB. Throughput: `sendrecv` for 512 kByte. Latency: `pingpong` for 0 byte messages.

The performance numbers of the directly connected ports can be found in the left column of table 1. The latency presented there is the half-round-trip time for 0 byte length messages as determined via the `pingpong` test. The low latency found in our setup is due to the ParaStation protocol[e]. On the same hardware a fine-tuned TCP-setup will reach a latency of about $28\,\mu$s on MPI-level; out of the box the MPI latency over TCP is often in the range of $60 - 100\,\mu$s. The throughput numbers are for 512 kByte messages. Larger messages give slightly less throughput of about $200\,$MB/s. This is due to cache effects when the main memory has to be accessed. The message size for half-throughput was found to be 4096 Byte for all setups.

In order to determine the influence of a single switch stage on the network performance the same benchmark was carried out communicating via a switch, *i.e.* using two nodes connected to the same level-1 switch. The corresponding results marked as "single switch" can also be found in table 1. It can be seen that there is almost no influence of the switch on the throughput. Since the latency rises from $18.6\,\mu$s to $21.5\,\mu$s each switch stage is expected to introduce a penalty of only $2.9\,\mu$s. The total latency when sending messages through all three stages of our testbed is anticipated to be about $27.5\,\mu$s which would correspond to a latency of about $9\,\mu$s from the switch alone. The throughput is not affected.

The above tests where done with a single pair of processes. Of course in order to show the full bisectional bandwidth of the crossbar switch configuration one has to use as many

[d]ParaStation uses a fine-tuned high-performance protocol reducing the overhead of protocols like TCP.
[e]In fact on other hardware latencies as small as $10\,\mu$s were found.

pairs as possible concurrently. Furthermore, the processes have to be distributed in such a way that a pair's ones are connected to different level-1 switches.

We have run our benchmarks on 140 processors accessible to us, with 70 pairs of processes. The first 24 pairs where distributed over the first two level-1 switches, *etc.* For each pair the two processes were placed on different level-1 switches. Therefore, all the traffic had to be sent via the level-2 switches. Of course this is a worst case communication pattern since real applications normally have traffic running also within a level-1 switch.

The numbers for the test case presented in table 1 are worst case numbers. *I.e.* the result for the pair giving the least throughput is displayed there. If one would take the average value of all pairs, the throughput is about 5% larger, the best performing pair even gives a result of about 218 MB/s. The total throughput is larger than 15 GB/s.

Based on the observed latency of $28.0\,\mu$s the actual latency introduced by the crossbar-switch was found to be $9.4\,\mu$sec. This result is far below the values of big Gigabit-Ethernet switches with full bisectional bandwidth—at a much lower price! We expect this number to be constant for up to 1152 ports.

9 Conclusion and Outlook

We have constructed a scalable crossbar switch using off-the-shelf Gigabit-Ethernet components. We ran our performance benchmarks on the Wuppertal ALiCEnext cluster system. Full bisectional bandwidth could be achieved at a price of less than € 125 per port[f]. So far we demonstrated our concept to work for 144 ports. As a next step we will scale up the ALiCEnext cabling to 528 processors and include additional mesh topologies. An international patent for our approach was submitted and is pending[14].

Acknowledgments

We thank the ALiCEnext team in Wuppertal for patience and help.

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, Nov. 1982.
8. IEEE standard 802.1D.
9. IEEE standard 802.1Q, IEEE standard 802.3ac.
10. IEEE standard 802.1s.
11. `http://www.alicenext.uni-wuppertal.de`.
12. `http://www.smc.com`.
13. Pallas MPI Benchmark now available from Intel as Intel MPI Benchmark (IMB) `http://www.intel.com/software/products/cluster/mpi/mpi_benchmarks_lic.htm`.
14. Patent: "Data Communication System and Method", EP 05 012 567.3.

[f] `http://www.pricewatch.com`.