Performance Evaluation of Real-Time Communication Services on High-Speed LANs under Topology Changes*

Juan Fernández[1], José Manuel García[1], and José Duato[2]

[1] Dpto. Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30071 Murcia (Spain)
{peinador, jmgarcia}@ditec.um.es
[2] Dpto. Informática de Sistemas y Computadores
Universidad Politécnica de Valencia, 46071 Valencia (Spain)
jduato@gap.upv.es

Abstract. Topology changes, such as switches being turned on/off, hot expansion, hot replacement or link re-mapping, are very likely to occur in NOWs and clusters. Moreover, topology changes are much more frequent than faults. However, their impact on real-time communications has not been considered a major problem up to now, mostly because they are not feasible in traditional environments, such as massive parallel processors (MPPs), which have fixed topologies. Topology changes are supported and handled by some current and future interconnects, such as Myrinet or Infiniband. Unfortunately, they do not include support for real-time communications in the presence of topology changes.
In this paper, we evaluate a previously proposed protocol, called *Dynamically Re-established Real-Time Channels (DRRTC) protocol*, that provides topology change- and fault-tolerant real-time communication services on NOWs. We present and analyze the performance evaluation results when a single switch or a single link is deactivated/activated for different topologies and workloads. The simulation results suggest that topology change tolerance is only limited by the resources available to establish real-time channels as well as by the topology connectivity.

1 Introduction

In the past few years, networks of workstations (NOWs) and clusters, based on off-the-shelf commodity components like workstations, PCs and high-speed local area networks (LANs), have emerged as a serious alternative to massive parallel processors (MPPs) and are the most cost-effective platform for high-performance servers. However, distributed real-time processing on NOWs and clusters is still a pending issue.

Distributed real-time applications impose strict conditions on traffic such as bounded delivery time (deadline) or guaranteed bandwidth [1]. In order to

* This work has been supported in part by the Spanish CICYT under grant TIC2000-1151-C07-03.

provide real-time communications, real-time channels [2] establish unidirectional connections among source and destination hosts. Once a real-time channel has been established, that is, resource reservation has finished, maximum delivery time and bandwidth are guaranteed.

Faults have been traditionally considered a major problem in distributed real-time processing since they may interrupt real-time communications. Thus, several researchers have proposed efficient solutions to this problem. The backup channel protocol (BCP), developed by Shin *et al.* [3] and based on real-time channels [2], performs recovery from faults by means of additional resources (backup channels).

On the other hand, topology changes, such as switches being turned on/off, hot expansion, hot replacement or link re-mapping, are very likely to occur in NOWs and clusters. Moreover, topology changes are much more frequent than faults. However, their impact on real-time communications has not been considered a major problem up to now, mostly because they are not feasible in traditional environments, such as massive parallel processors (MPPs), which have fixed topologies. Topology changes degrade network ability to establish new real-time channels because the routing tables are not up-to-date. Therefore, not all the available resources can be fully exploited. To make the best use of resources, every time a topology change or fault occurs, routing tables must be updated to reflect the new configuration. Dynamic reconfiguration, recently proposed by Duato *et al.* [4, 5], assimilates topology changes by updating routing tables without stopping traffic. Note that dynamic reconfiguration by itself provides neither quality of service nor real-time services, but it provides support for an additional mechanism designed to meet real-time requirements. Finally, topology changes are supported and handled by some current and future interconnects, such as Myrinet [6] or Infiniband [7]. Unfortunately, they do not include support for real-time communications in the presence of topology changes.

In this paper, we evaluate a previously proposed protocol [8], called Dynamically Re-established Real-Time Channels (DRRTC) protocol, to provide topology change- and fault-tolerant real-time communication services on NOWs. The novelty of this protocol primarily relies on the ability to assimilate hot topology changes and faults while still providing real-time communication services as well as best-effort ones. To do this, our protocol is based on real-time channels with single backup channels and dynamic reconfiguration. Real-time channels provide real-time communications, single backup channels provide single-fault tolerance, and dynamic reconfiguration provides topology change tolerance and tolerance to additional faults. We present and analyze the performance evaluation results when a single switch or a single link is deactivated/activated for different topologies and workloads. All interrupted real-time channels are re-established after a topology change when two real-time connections per host are established. As the workload increases, channel recovery guarantees decrease. In this way, the simulation results suggest that topology change tolerance is only limited by the resources available to establish channels as well as by topology connectivity.

The rest of this paper is organized as follows. Next section presents our protocol. Network model is depicted in Sect. 3. In Sect. 4, the performance evaluation results are shown. Sect. 5 describes related work. Finally, we present our conclusions and feasible ways of future work.

2 Dynamically Re-established Real-Time Channels

This section describes the protocol previously proposed in [8] in an informal way. In order to support real-time communications, we set up a primary channel and a single secondary channel[1] for each real-time channel. Once a real-time channel has been established, real-time messages flow through the primary channel from source host to destination host until the real-time channel is closed or the primary channel is broken down. When either a hot topology change or a fault breaks down a primary channel, real-time messages are redirected through its secondary channel. At the same time, the dynamic reconfiguration algorithm described in [4, 5] is triggered. The dynamic reconfiguration process updates routing tables in such a way that secondary channels could be re-established for the affected real-time channels as long as the network topology still provides an alternative physical path. The dynamic reconfiguration process does not affect the deadline of real-time messages flowing through real-time channels because traffic is allowed during reconfiguration. After reconfiguration, a new secondary channel is allocated for each affected real-time channel regardless of whether the old secondary channel has become the new primary channel or the old secondary channel was broken down. The procedure for interrupted secondary channels is the same as for primary ones. However, in this case, real-time traffic is not affected. On the other hand, if several topology changes or faults concurrently occur, the dynamic reconfiguration protocol combines them in a single process [5], and so, our protocol remains valid. Next, we are going to describe the different stages of our protocol.

2.1 Real-time Channel Establishment

A *real-time channel* is a unidirectional connection between a pair of hosts H_1 and H_2. A real-time channel consists of a primary channel and a single secondary one. Each of them is a set $\{H_1, H_2, B, D, T\}$ where H_1 is the source host, H_2 is the destination host, B is the required bandwidth, D is the maximum admissible latency or deadline for real-time messages, and T is the type of channel, that is, primary or secondary. Enough resources (a virtual channel and enough bandwidth) are assigned to each channel in each switch along its path to meet its bandwidth and deadline requirements before real-time messages are transmitted. Real-time messages flow through the primary channel from H_1 to H_2 until the real-time channel is closed or the primary channel is broken down. In the meantime, the secondary channel remains idle and does not consume link

[1] The terms secondary and backup are used interchangeably throughout this paper.

bandwidth but just a virtual channel in each switch. In order to maximize topology change tolerance, the primary and the secondary channels must not share physical resources. In this way, disjoint physical paths are essential.

Before transmitting real-time messages, H_1 must reserve the necessary resources for both the primary and the secondary channels. A best-effort message, called RTC_REQUEST, is sent from H_1 to H_2 to set up the primary channel. In each switch, the request message is processed to check for the the availability of resources and to reserve them as we will see later. If there are not enough resources, a best-effort message, called RTC_RESPONSE(False), is returned to H_1. When a request message arrives at H_2, the primary channel is accepted if, and only if, latency of request message is shorter than channel deadline. Then, H_2 sends a best-effort message, called RTC_RESPONSE(True), back to H_1. The path followed by the response messages may not be the same used by the request to establish the channel. If H_1 receives a false response message or channel timeout expires, resources are released by means of a RTC_MSG(Release) message that flows through the channel. Channel timeout allows H_1 to release resources when no response message is received. After a few cycles, H_1 will try to establish the primary channel again until it is established or a maximum number of attempts is reached. If H_1 receives a true response message, the secondary channel will be established likewise. A request is sent from H_1 to H_2 to set up the secondary channel. In each switch, the request message is processed to check for the the availability of resources, to reserve them, and also to verify that the primary channel does not go through that switch. This is because the primary and the secondary channels must not share resources in order to maximize fault tolerance.[2] Once both the primary and the secondary channels have been successfully established, the real-time channel establishment process has finished. Otherwise, resources are released, and the real-time channel is rejected. Note that several channels could be concurrently established.

Finally, we are going to analyze the processing of requests in detail. First, for each possible output port provided by the routing function for a request message, we check for the availability of resources, that is, a free virtual channel and enough bandwidth. Output ports without enough resources are no longer considered. If the request corresponds to the secondary channel, it must be also verified that the primary channel does not go through the switch. To do this, each switch has a channel table that keeps track of all channels that go through it. If there are not enough resources or the primary channel goes through the switch, a false response message is returned to H_1. Once an appropriate output port has been found, the channel is added to the channel table, resources are reserved (a virtual channel and enough bandwidth), and the request is forwarded to the next switch. Note that selection of channel routes is distributed among all switches, that is, global information is not necessary to establish channels.

[2] Initially, two NICs per host are assumed so that the primary and the secondary channels have to share neither switches nor links.

2.2 Real-time Channel Operation

After a real-time channel has been successfully established, H_1 begins to inject real-time messages, called RTC_MSG messages, through the primary channel. Real-time messages flow from H_1 to H_2 through the primary channel until the real-time channel is closed or the primary channel is broken down. In the former case, resources are released by means of two release messages. In the latter case, real-time messages are redirected through the secondary channel and a new secondary channel will be allocated if possible. Note that if the secondary channel is broken down, real-time traffic is not affected.

2.3 Real-time Channel Recovery

Once a real-time channel has been set up and is transmitting real-time messages, we have to deal with the problem of channel recovery while still satisfying real-time requirements. Let us assume that a single link fails or is turned off. Next, the two adjacent switches detect the fault and determine the broken channels looking up their real-time channel tables. For each channel whose output port matches with the broken link, a best-effort message, called RTC_REPORT, is sent to its corresponding source host. The switch remains in the releasing state until the corresponding release message is received. For each channel whose input port matches with the broken link, a release message is sent to the destination host through the channel. Every time a report arrives at a source host for the primary or the secondary channels, a release message releases resources from that source host up to the previous switch. In the former case, real-time traffic is redirected through the secondary channel, that is, the secondary channel becomes the new primary channel. In the latter case, real-time messages continue flowing through the primary channel. In any case, after reconfiguration, the secondary channel will be re-established if possible.

At the same time that the adjacent switches detect the fault, the dynamic reconfiguration protocol described in [4, 5] is triggered. This process performs sequences of partial routing table updates to avoid stopping traffic, and trying to update routing tables in such a way that a secondary channel could be re-established for each affected real-time channel. After reconfiguration, a new secondary channel is allocated, if possible, for each affected real-time channel regardless of whether the old secondary channel has become the new primary channel or the old secondary channel was broken down.

2.4 Modified Switch Architecture

Although a detailed hardware design is out of the scope of this paper, switch architecture is depicted to help readers to understand our proposal (see [8]). As shown in Fig. 1, our approach uses an input-buffered switch with virtual channels. Virtual cut-through is used because it may replace wormhole in the near future in NOWs [9]. Physical link bandwidth is 1.24 Gbps, and links are 8-bit wide. Each output port has sixteen virtual channels: thirteen RTC virtual

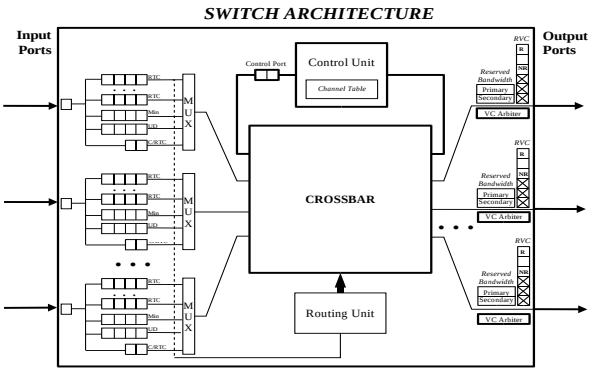


Fig. 1. Modified Switch Architecture

channels that can be reserved for real-time channels, Min and UD (Up*/Down* routing) virtual channels that are used by best-effort traffic, and C/RTC that is used by DRRTC control messages and all control messages generated by reconfiguration. To build a deadlock-free routing function we use the methodology described in [10]. Primary and Secondary store the amount of bandwidth reserved by primary and secondary channels respectively. RVC stores the reserved virtual channels. Virtual Channel Arbiter implements the link scheduling algorithm used to forward messages. The scheduling algorithm is based on that of Infiniband [7]. The control unit processes requests and all control messages generated by reconfiguration. The Channel Table keeps track of all channels that go through the switch. The Control Port allows switch to inject messages.

3 Network Model

Network is composed of a set of switches connected by point-to-point links and hosts attached to switches through two network interface cards (NICs). In this way, we eliminate single points of failure. Network topologies were randomly generated and are completely irregular. However, some restrictions were applied for the sake of simplicity. First, all switches have eight ports, four ports connected to other switches and four ports connected to hosts. Therefore, the number of hosts is two times the number of switches. Second, two switches are not connected by more than one link. Finally, the assignment of hosts to switches was made so that the intersection of the sets of hosts connected to two switches is one host maximum. This is because we want to avoid bottlenecks when re-establishing channels after reconfiguration.

Simulation was used instead of analytical modeling. Our simulator models the network at the packet level. Simulation results were generated from three irregular topologies (T1, T2 and T3) consisting of 64 switches and 128 hosts each other (2 NICs per host). Workload is composed of CBR real-time channels of 1 Mbps and deadline is equal to inter-arrival time (IAT). Packet length of real-

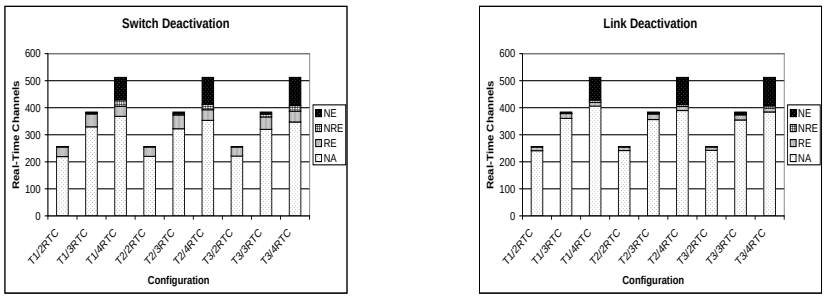
time messages is 16 Bytes and packet length of best-effort messages is 256 Bytes. All hosts try to establish the same number of real-time channels. Nevertheless, the number of real-time channels per host varies between two and four (2RTC, 3RTC and 4RTC). Therefore, the total number of real-time channels is 256, 384 or 512 according to the number of real-time channels per host. Destinations of channels are randomly chosen among all hosts in the network. Real-time channels are initially established during an interval of time proportional to the number of real-time channels per host.

4 Simulation Results and Analysis

4.1 Switch Deactivation and Activation

In Fig. 2(a), we show the evolution of real-time channels for different configurations when a single switch is turned off. Each column represents the total number of real-time channels that hosts are trying to establish, that is, 256, 384 and 512 for 2RTC, 3RTC and 4RTC respectively. *NE* corresponds to initially non-established real-time channels and the rest corresponds to successfully established real-time channels. As illustrated in Fig. 2(a), for all topologies, as the number of channels per host increases, the number of initially non-established channels increases too. Whereas 100% of channels are established for 2RTC per host, the percentage of non-established channels varies varies from 1.3% (T1) to 1.8% (T3) for 3RTC per host, and from 16.2% (T1) to 20.3% (T3) for 4RTC per host. For each topology, we successively simulate the deactivation of eight switches, one by one, that is, each switch is deactivated while the rest remain activated. Switches are randomly chosen among all switches in the network. Next, mean values are represented. *NA* corresponds to the average of non-affected real-time channels, and the rest corresponds to the average of interrupted real-time channels. *RE* is the average of re-established real-time channels after reconfiguration, and *NRE* is the average of non re-established real-time channels because a new secondary channel could not be allocated after reconfiguration. The average of affected real-time channels by switch deactivation is very similar for all configurations (it varies from 13% for T1/3RTC to 15% for T3/4RTC). However, the averages of re-established channels differ considerably from each other according to the number of real-time channels that hosts are trying to establish. As shown in Fig. 2(a), while all interrupted channels are re-established for 2RTC, the average of re-established channels varies from 81% (T3) to 94% (T1) for 3RTC per host, and from 61% (T1) to 66% (T2) for 4RTC per host. Finally, note that switch activation does not affect any already established real-time channel. Consequently, no further analysis is needed.

Now, we analyze why results get worse when increasing the number of real-time channels per host. In Fig. 3(a) and Fig. 3(b) we show the reserved virtual channels for all switch-to-switch links before reconfiguration for 2RTC and 4RTC per host respectively. For each switch, four bars represent the reserved virtual channels of its four ports connected to other switches. As we can observe, for 2RTC per host, only a few ports have no free virtual channels (only 47% of



(a) (b)

Fig. 2. Evolution of real-time channels for different configurations when a single switch/link is turned off. Configurations correspond to topologies T1, T2 and T3 when hosts try to establish 2RTC, 3RTC and 4RTC per host

virtual channels are reserved). However, for 4RTC per host, most ports have consumed all its virtual channels (87% of virtual channels are reserved). Hence, the average of established channels decreases as the channels per host increase because free virtual channels are used up in most ports. After reconfiguration, the reasoning is same as the previous one (49% versus 87% of reserved virtual channels for 2RTC and 4RTC respectively).[3]

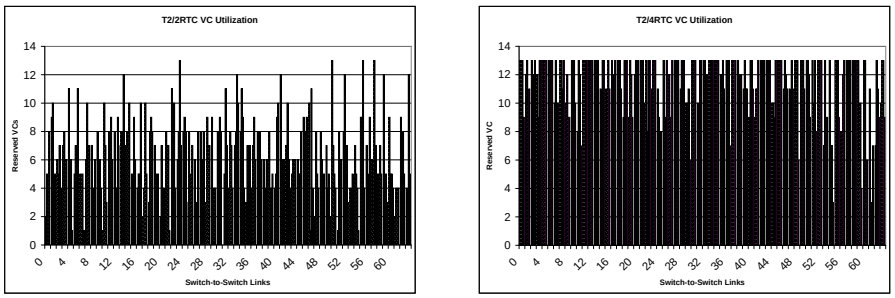
4.2 Link Deactivation and Activation

In Fig. 2(b), we show the evolution of real-time channels for different configurations when a single link is turned off. For each topology, we simulate the deactivation of the four switch-to-switch links, one by one, of the same switches used to generate the results in Fig. 2(a). As expected, the average of affected real-time channels by link deactivation is very similar for all configurations (it varies from 4% for T1/3RTC to 5% for T1/3RTC) but lower than the one by switch deactivation. Apart from that, results keep the same proportions as in the case of switch deactivation. Finally, note that link activation does not affect any already established real-time channel so that no further analysis is needed.

5 Related Work

Shin *et al.* have proposed the *Backup Channel Protocol* (BCP) [3] to achieve fault-tolerant real-time communications. In this approach, the maximum number of admissible faults depends on the maximum number of alternative paths provided by the routing function to establish the backup channels. Moreover, topology change tolerance is not provided.

[3] Figures have been omitted for the sake of brevity.



(a) (b)

Fig. 3. VC usage before/after switch deactivation when establishing 2RTC/4RTC

Static reconfiguration techniques (Autonet [11] and Myrinet with GM [12]) stop user traffic to update routing tables. Although reconfigurations are not frequent, they can degrade performance considerably [11] and real-time constraints can not be met because of traffic disruption. Duato *et al.* [4, 5] have proposed a dynamic reconfiguration algorithm, called *Partial Progressive Reconfiguration*, to minimize the negative effects of static reconfiguration on network performance. The protocol guarantees that the global routing algorithm remains deadlock-free at any time. Pinkston *et al.* [13] have developed a simple but effective strategy for dynamic reconfiguration in networks with virtual channels. Lysne *et al.* [14] aim at reducing the scope of reconfiguration identifying a restricted part of the network, the *skyline*, as the only part where a full reconfiguration is necessary. Avresky *et al.* have recently presented a new dynamic reconfiguration protocol, called *NetRec*, for high-speed LANs using wormhole routing [15].

6 Conclusions and future work

In this paper, the DRRTC protocol has been evaluated. The novelty of this protocol primarily relies on the ability to assimilate hot topology changes and faults while still providing real-time communication services. We have evaluated its behavior when a single switch or a single link is turned on/off for different topologies and workloads. Simulation results suggest that the DRRTC protocol provides topology change tolerance that is only limited by the resources available to establish real-time channels as well as by the topology connectivity.

Using the ideas presented in this paper, future work involves: a quantitative characterization of the DRRTC protocol under multiple topology changes, an analysis of the optimal assignment of hosts to switches within a bounded distance, and identifying of an upper bound to the protocol in order to ensure channel recovery. We are also planning to develop an InfiniBand [7] version of DRRTC.

Acknowledgments

The authors thank *Francisco J. Alfaro* at the University of Castilla La Mancha for his comments during the course of this work. This work has been supported in part by the Spanish CICYT under grant TIC2000-1151-C07-03.

References

[1] ATM Forum: ATM Forum Traffic Management Specification. Version 4.1. (1995)
[2] Kandlur, D.D., Shin, K.G., Ferrari, D.: Real-Time Communications in Multi-hop Networks. IEEE Transactions on Parallel and Distributed Systems **5** (1994) 1044–1056
[3] Han, S., Shin, K.G.: A Primary-Backup Channel Approach to Dependable Real-Time Communication in Multi-hop Networks. IEEE Transactions on Computers **47** (1998)
[4] Alfaro, F.J., Bermudez, A., Casado, R., Quiles, F.J., Sanchez, J.L., Duato, J.: Extending Dynamic Reconfiguration to NOWs with Adaptive Routing. In: Proceedings of Communications, Architecture, and Applications for Network-based Parallel Computing Workshop. (2000)
[5] Avresky, D., ed.: 10, Dynamic Reconfiguration in High Speed Local Area Networks. José Duato, Rafael Casado, Francisco J. Quiles and José L. Sánchez. In: Dependable Network Computing. Kluwer Academic Press (1999)
[6] Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N., Su, W.K.: Myrinet: A Gigabit-per-Second Local-Area Network. IEEE Micro **15** (1995) 29–36
[7] InfiniBand Trade Association: InfiniBand Architecture Specification Volume 1. Release 1.0. (2000)
[8] Fernández, J., García, J.M., Duato, J.: A New Approach to Provide Real-Time Services in High-Speed Local Area Networks. In: Proceedings of Workshop on Fault-Tolerant Parallel and Distributed Systems. (2001) Held in conjunction with IPDPS'01, San Francisco, CA.
[9] Duato, J., Robles, A., Silla, F., Beivide, R.: A Comparison of Router Architectures for Virtual Cut-Through and Wormhole Switching in a NOW Environment. In: Proceedings of International Parallel Processing Symposium. (1998)
[10] Silla, F., Duato, J.: Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology. In: Proceedings of International Conference on High Performance Computing. (1997)
[11] Rodeheffer, T., Schroeder, M.D.: Automatic Reconfiguration in Autonet. Proceedings of ACM Symposium on Operating System Principles (1991)
[12] Myricom, Inc.: Myrinet GM documentation. (1999) http://www.myri.com/GM.
[13] Pang, R., Pinkston, T., Duato, J.: The Double Scheme: Deadlock-free Dynamic Reconfiguration of CutThrough Networks. In: Proceedings of International Conference on Parallel Processing. (2000)
[14] Lysne, O., Duato, J.: Fast Dynamic Reconfiguration in Irregular Networks. In: Proceedings of International Conference on Parallel Processing. (2000)
[15] Avresky, D.R., Varoglu, Y.: Dynamically Scaling Computer Networks. In: Proceedings of Workshop on Fault-Tolerant Parallel and Distributed Systems. (2001) Held in conjunction with IPDPS'01, San Francisco, CA.