Evaluation of Dynamic Reconfiguration Architecture in Multi-Hop Switched Ethernet Networks

Mohammad Ashjaei[1], Paulo Pedreiras[2], Moris Behnam[1], Luis Almeida[3], Thomas Nolte[1]
[1] MRTC/Mälardalen University, Västerås, Sweden
[2] DETI/IT/University of Aveiro, Aveiro, Portugal
[3]IT/DEEC/University of Porto, Porto, Portugal

***Abstract*—On-the-fly adaptability and reconfigurability are recently becoming an interest in real-time communications. To assure a continued real-time behavior, the admission control with a quality-of-service mechanism is required, that screen all adaptation and reconfiguration requests. In the context of switched Ethernet networks, the FTT-SE protocol provides adaptive real-time communication. Recently, we proposed two methods to perform the online reconfiguration in multi-hop FTT-SE architectures. However, the methods lack the experimental evaluation. In this paper, we evaluate both methods in terms of the reconfiguration time.**

I. Introduction

The interest of using Ethernet switches in real-time distributed applications is rapidly increasing due to its features such as wide availability, low cost and high throughput. However, using commercially available (COTS) switches in time critical applications may hinder the ability to provide real-time guarantees. In addition, operating conditions may change, for example triggered by changes in the environment that may lead to increased communication requirements. In turn, this calls upon adequate dynamic adaptation and reconfiguration policies that ensure continued timeliness in the communications.

The limitations imposed by the simple use of COTS switches have been addressed in [1]. Some relatively old solutions are based on enhanced switches such as EtheReal [2] and the EDF Scheduled Switch [3], both using reserved channels for traffic transmission. Some other solutions made it to the market, such as PROFINET-IRT [4] and TTEthernet [5], both optimized for time-triggered operation. However, these switches are configured in ways that are not suited for dynamic real-time systems that are operating in dynamic environments. despite the performance improvements offered by using these enhanced switches, their usage result in a high cost and a lower availability compared to COTS switches.

A more effective solution is to control the traffic submitted to the COTS switch avoiding queue build up and then to use adequate traffic scheduling policies. This can be achieved with a master-slave technique which is the case of the FTT-SE protocol [6].

The FTT-SE (Flexible Time-Triggered Switched Ethernet) protocol is a bandwidth-efficient master-slave protocol that handles all types of message streams including real-time periodic, real-time sporadic and non-real-time traffic. The protocol provides temporal isolation between the message types by defining specific reserved bandwidth for each type of message streams. Moreover, it caters for requirements of dynamic reconfiguration and adaptability.

The multi-hop communication over the FTT-SE protocol was addressed in [7] and [8], where three architectures were studied. It turned out that, among those architectures, the one where several master nodes coexist to control the traffic in a group of switches performs better in terms of bandwidth utilization [8]. This architecture is called *hybrid architecture*.

Recently, we proposed two different methods for on-line reconfiguration in the hybrid architecture [9]: centralized and distributed. However, the methods lack the experimental evaluation. In this paper, we evaluate both methods in terms of the reconfiguration time.

The paper organizes as follow. The next section describes the hybrid FTT-SE architecture. Section III presents the reconfiguration methods. Section IV depicts the evaluation of the methods and Section V concludes the paper and shows future directions.

II. The Hybrid Architecture

An example of the hybrid architecture is depicted in Fig. 1. In this architecture, a group of switches along with their associated nodes that have the same parent switch form a *cluster* (e.g., Cluster2 in Fig. 1). The traffic within a cluster is controlled by one master node connected to the parent switch of the cluster (e.g., M2 is a master node for Cluster2). Note that, the master of the root cluster (M1 in Fig. 1) is included in its cluster since it cannot be accounted as one cluster itself.

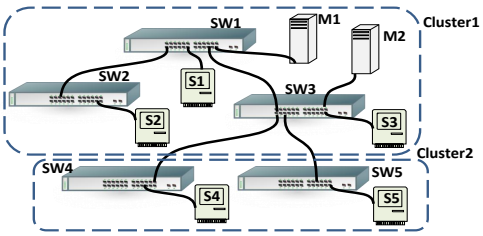

Fig. 1. The Hybrid Architecture

Considering different clusters, the message types are categorized as follows. A message that is transmitted within a cluster is called *internal*, while a message that is transmitted across clusters is called *external*.

The master nodes schedule the respective traffic on-line according to any desired scheduling policy (e.g., Fixed Priority Scheduling), on a cyclic basis. The basic cycle has a fixed duration of time and it is called Elementary Cycle (EC). In the hybrid architecture, each EC is partitioned among the traffic types, i.e., internal/external and synchronous/asynchronous traffic (Fig. 2). The external asynchronous window is further split into cluster sub-windows.

The scheduler in the master node computes the activation instants of the synchronous messages and schedules them EC

by EC, ensuring that they fit in the respective window. Master nodes schedule both internal and external messages in parallel and communicate the scheduled messages in each EC to the nodes through a message sent in the beginning of each cycle called the Trigger Message (TM).

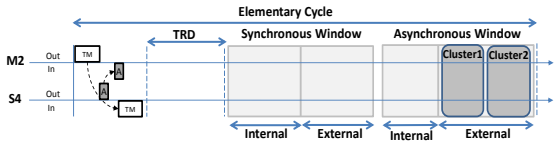

Fig. 2. The Elementary Cycle in the Hybrid Architecture

The activation of asynchronous messages is unknown in advance. Therefore, a Signaling Message (SIG) is transmitted in parallel with the TM but in the opposite direction (the links are full duplex) informing the master node of the respective cluster about the pending requests (e.g., A from S4 to M2 in Fig. 2). The master then schedules the asynchronous messages adequately and inserts them into the TM.

The slave nodes initiate message transmissions after receiving the TM and decoding it, which takes an amount of time called the Turn Around Time (TRD) (Fig.2). The reader is referred to [8] for more details about the hybrid architecture.

III. Reconfiguration Methods

According to the FTT-SE protocol, in the master node the *System Requirements Data Base (SRDB)* contains the traffic parameters. The scheduler scans the information in the SRDB to schedule the traffic for the following ECs. In addition, in each slave node, the *Node Requirements Data Base (NRDB)* holds the traffic parameters related to that specific node. Upon each TM reception, the node checks the NRDB to decide whether it is the transmitter of any of the scheduled messages in that cycle. These databases, SRDB and NRDB, must be synchronized, i.e., the messages and their parameters need to be identical in both databases.

The online reconfiguration includes four steps. The first step is the negotiation between the slave nodes and the master node. The negotiation may refer to removing/adding streams, or changing the parameters of the streams and it may be triggered by slave or master nodes, as well as by dedicated QoS broker node. The second step is the admission control, to verify the feasibility of the proposed changes. In this step a response time analysis is used to check the available resources. The third step is the resource reservation in which the resources, after being distributed by the QoS management, are allocated to the message streams. The final step is the mode-change where the SRDB and NRDBs are updated. This transition is done gradually, yet in a bounded time, in order to provide a safe mode-change in the system.

In order to negotiate and perform the mode-change two real-time asynchronous messages are provided, one for sending the request from the slave node to the master node, and the other to send the update for the slave nodes. Note that the reconfiguration procedure is fully deterministic to achieve timeliness guarantee. Therefore, the mentioned steps are carried out in a bounded time including the request and update message transmission between slave nodes and the master node, i.e., the response time of the assigned asynchronous messages for request and update is bounded and known.

In [9] we proposed two reconfiguration methods in the hybrid architecture, the *centralized* and the *distributed* reconfigurations. The former method uses one master node to perform all the decisions regarding the change requests, while the latter method is fully distributed and all masters process the requests in parallel. Herein, we briefly review the methods.

A. Centralized Reconfiguration

Some of the mentioned reconfiguration steps, like admission control and QoS re-distribution, may encompass computationally intensive operations ([10], [11]). Providing short response times to reconfiguration requests in those cases may require a considerable amount of processing power, much higher than the one necessary to carry out the "regular" master operations (e.g. scheduling, control messages handling). In this scenario, it may be more resource-efficient having a single node, embodied with a higher processing capacity, in charge of processing all the reconfiguration requests. The results are then communicated to the network master nodes, which only need to carry out a minimum extra processing. This architecture is designated *centralized* method.

Without loss of generality, lets assume that the root master is the one responsible for deciding about the reconfiguration requests. The centralized method requires that all requests made by the slave nodes reach the root master. It is also necessary to allow the root master to communicate its decisions to the other masters. Thus, it is necessary to create a two-way channel between the root master and each one of the other masters, to convey the reconfiguration requests and replies. The global event sequence involved in a reconfiguration is depicted in Fig. 3, where Slave 1 (S1), belonging to the cluster managed by Master 2 (M2), issues a reconfiguration request that also affects a message that is produced/consumed by Slaves 2 (S2) and 3 (S3). S2 also belongs to the cluster managed by M2, while S3 belongs to the cluster managed by M1.

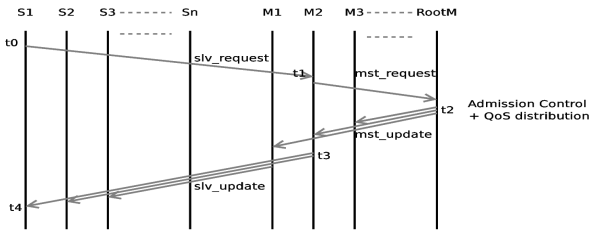

Fig. 3. Centralized Reconfiguration Event Sequence

Slave 1 requests a reconfiguration at $t = t_0$, issuing a *slv_request* message to its cluster head (M2). M2 forwards the request to the root master (RootM), via a *mst_request*, at time $t = t_1$. RootM carries out the Admission Control and eventually the QoS redistribution. The results of these operations are then communicated to the other master(s) at time $t = t_2$. If the change request is denied, RootM sends a *mst_update* message to M2, notifying the decision, and no SRDB updates are required. Conversely, if the decision is positive, RootM sends a *mst_update* message to all masters, specifying which changes must be made to the respective SRDBs. For both cases the masters inform the slaves of the result of the reconfiguration, to synchronize the SRDBs and the NRDBs.

Note that a single request may cause changes in several messages, due to the QoS redistribution. This may happen e.g. after deleting a message in a highly loaded system. The bandwidth that becomes free may be distributed, by the QoS

manager, among other messages that may take advantage of it. As a side effect, for systems with many external messages, *mst_update* and *slv_update* messages may have to be fragmented in several Ethernet frames.

Updating the SRDB and NRDB may take several ECs to complete. It is necessary that all the masters instantiate the updates at the same time, in order to schedule the external traffic consistently. Therefore, the *mst_update* messages encode the EC in which the changes should take effect. The RootM node is able to compute a safe upper bound to this value because the messages involved in the communication of its decisions (*mst_update* and *slv_update*) use asynchronous real-time (thus predictable) channels and RootM has global network knowledge.

B. Distributed Reconfiguration

This approach is possible because the masters have a consistent view of the shared resources (external messages are known by all masters and external windows have the same size in all clusters) and the admission control and QoS management algorithms are deterministic. Under these circumstances, despite operating quasi-independently, masters will reach consistent decisions. The only problem that remains is guaranteeing that the reconfiguration decisions are applied synchronously by all masters. Assuring this implies obtaining an upper bound to the execution time of the Admission Control and QoS tasks in all masters, in addition to an upper bound to the delivery of management messages. These constraints can also be met, since masters must have some sort of real-time support (executive or RTOS), as they have to carry out several real-time tasks (e.g. scheduling, TM dispatching), thus it is possible to bound the execution time of those algorithms. Additionally, as in the centralized approach, the reconfiguration is supported by real-time channels, thus the communication time can also be bounded. Fed with these two bounds, the master that receives a reconfiguration request may compute a time bound and encode it in the reconfiguration requests that it sends to its peers, permitting the synchronization of the instantiation of the reconfiguration results.

The global event sequence involved in a distributed reconfiguration is depicted in Fig. 4, where a network configuration similar to the one considered in Section II is assumed.

S1 sends a reconfiguration request to M2 at $t = t_0$. Subsequently M2 computes an upper bound to the processing time required by the reconfiguration request and forwards this information, together with the actual reconfiguration request, to all other masters (*mst_request* messages). Such requests are delivered until $t = t_2$ and then processed. $U_P = t_3 - t_2$ represents the upper bound of the processing time of all masters. Then each master notifies its slaves about the eventual changes to the NRDB, via the *slv_update* messages. Since these messages use a real-time channel, they are delivered on time by each master, thus they take effect at the predefined time $t = t_4$.

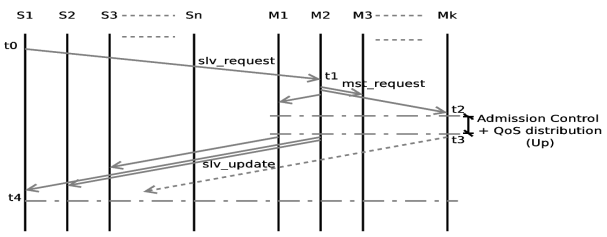

Fig. 4. Distributed Reconfiguration Event Sequence

It may occur that two conflicting requests are issued from two slave nodes, i.e., the consequence changes of the requests are in conflict. In order to handle this situation, some sort of agreement among the master nodes are required to be established. Therefore, a priority for each request based on the priority of the affecting stream is considered. The master nodes in case of receiving two or more conflicting requests, discard the lower priority ones, and in turn, apply only the highest priority request.

IV. Evaluation

In this work, we evaluated the reconfiguration time which is divided into two parts: the computational time and the reconfiguration signaling time. The computational time is the time required to compute the response time of all the messages in a set by one master node. The reconfiguration signaling time is the time required for negotiation and update steps. In order to evaluate these two parameters we considered two network examples, one small network with three switches and the other, a larger network with five switches. The two networks are depicted in Figure 5 and 6.

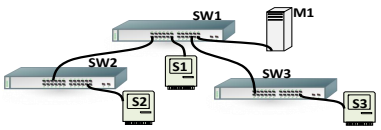

Fig. 5. A Network with Three Switches

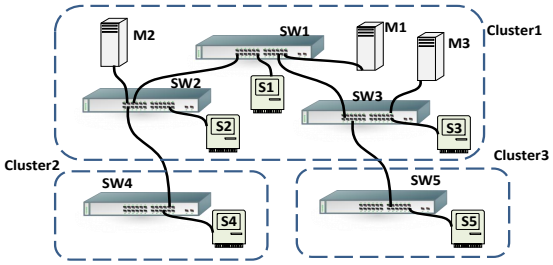

Fig. 6. A Network with Five Switches

A. Computational Time

In this evaluation, we generated 1000 sets of messages randomly. The periods of the messages are selected within $[2,10]EC$ and their priorities are assigned based on the Rate Monotonic algorithm within $[1,4]$, where 1 represents the highest priority. Note that, the messages can share a priority level in this evaluation when their periods are equal. Moreover, the transmission time of the messages are chosen within $[80,123]\mu s$, where $123\mu s$ is the transmission time of a frame with 1542KB and it is the maximum Ethernet packet size. Moreover, the network capacity is set to $100Mbps$ and the EC size is set to $2ms$.

As a master node in the network, we considered an Atmel evaluation kit EVK1100 with an Atmel 32-bit AVR UC3A microcontroller. Remember that, this node is responsible to compute the response time for the messages. The response time calculation is done based on the analysis presented in [8]. Note that, in both centralized and distributed reconfiguration methods, all master nodes should calculate the response time of all messages.

The computational time for the generated sets containing different number of messages is presented in Table I and Table II, for the networks depicted in 5 and 6, respectively.

No. of Msg	5	10	15	20	30	40
Min (μs)	2666	9717	21066	56566	60215	80854
Mean (μs)	2671	9724	21073	56571	60223	80857
Max (μs)	2677	9728	21077	56577	60225	80865

TABLE I. COMPUTATIONAL TIME FOR SMALL NETWORK

No. of Msg	5	10	15	20	30	40
Min (μs)	2677	9770	21408	57142	61527	81435
Mean (μs)	2682	9778	21412	57151	61535	81532
Max (μs)	2688	9781	21418	57153	61537	81604

TABLE II. COMPUTATION TIME FOR LARGE NETWORK

As it can be seen in both tables, the computational time is increasing sharply by increasing the number of messages. Considering the EC size of 2*ms* for this evaluation, the average computational time for 10 messages in both small and large networks is 2 ECs and it is increasing up to 41 ECs for 40 messages. This shows that using a centralized reconfiguration method is more efficient for the networks with high number of messages as we can have a high capacity node to compute the reconfiguration parameters and the rest of the master nodes can be regular capacity nodes.

Moreover, the size of network (number of clusters and nodes) does not affect significantly the computational time as in both small and large network examples, for the same number of messages, the computational time is the same (2EC).In fact, the only part that changes in the response time calculation by increasing the size of a network is finding the interference in all routes which is much faster than computing the response time in several iterations.

B. Reconfiguration Signaling Time

In this evaluation, we assumed a change request from node S2 in Figure 5. Then, we measured the response time of the negotiation and update signaling, i.e., the response time of the request and update messages. As the reconfiguration messages are asynchronous, we filled up the asynchronous window in the EC with other asynchronous data messages. However, the reconfiguration messages have the priority of 1 (the highest), hence they are scheduled before the data messages. Note that, the data messages are generated randomly with the same setting as the previous evaluation.

Table III shows the reconfiguration signaling time for the small network example considering different number of messages in the set. Note that, the response time is represented in number of ECs.

Number of Messages	10	20	30	40	50
Centralized (EC)	4	4	4	4	4
Distributed (EC)	4	4	4	4	4

TABLE III. THE RECONFIGURATION SIGNALING TIME

As it can be seen increasing the number of data messages does not affect the reconfiguration signaling time. The reason is that the reconfiguration messages have the highest priority and their transmission times are very small, hence they are scheduled always before the data messages. Also, in both centralized and distributed methods, the signaling time of reconfiguration is the same. This is because the requests, in both methods, are crossing only one hop.

Table IV shows the reconfiguration signaling time for the network example depicted in Figure 6.

By increasing the size of the network to 3 clusters, the centralized method requires more time for reconfiguration, as shown in Table IV. This is due to passing the requests to the root master and sending back the decision to the masters. However, in the distributed method, the master sends the requests to all other masters and the decision is sent to the slave nodes directly.

Number of Messages	10	20	30	40	50
Centralized (EC)	8	8	8	8	8
Distributed (EC)	6	6	6	6	6

TABLE IV. THE RECONFIGURATION SIGNALING TIME

Although the reconfiguration signaling time is less when using the distributed method, considering the computational time that has bigger effect, it is still more efficient to use the centralized method. Moreover, handling the concurrent requests in centralized method is much easier due to the serialization of the requests in that method.

V. CONCLUSION AND FUTURE WORK

In this paper, we evaluated the reconfiguration methods in terms of the computational time and the reconfiguration signaling time. The results show that the centralized reconfiguration method is more efficient for networks with high number of messages, even though the reconfiguration signaling time is less when using the distributed method. Moreover, in the centralized method, there is no need to consider the concurrent handling of the requests in as the requests are handled in serial with one master node. The ongoing work aims at implementing the methods on the hardware.

ACKNOWLEDGMENT

This work is supported by the Swedish Foundation for Strategic Research via the PRESS project. Also, it is partially supported by the Portuguese Government through FCT grants Serv-CPS PTDC/EEA-AUT/122362/2010.

REFERENCES

[1] P. Pedreiras and L. Almeida, "Approaches to enforce real-time behavior in ethernet," in *CRC Press*, Feb. 2005.

[2] S. Varadarajan and T. Chiueh, "Ethereal: a host-transparent real-time fast ethernet switch," in *6th Int. Conference on Network Protocols*, 1998.

[3] H. Hoang and M. Jonsson, "Switched real-time ethernet in industrial applications - deadline partitioning," in *9th Asia-Pacific Conference on Communications (APCC)*, 2003.

[4] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet io irt message scheduling," in *21st Euromicro Conf. on Real-Time Systems (ECRTS)*, 2009.

[5] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "Ttethernet dataflow concept," in *8th IEEE International Symposium on Network Computing and Applications*, 2009.

[6] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over cots ethernet switches," in *6th IEEE International Workshop on Factory Communication Systems (WFCS)*, June 2006.

[7] R. Marau, M. Behnam, Z. Iqbal, P. Silva, L. Almeida, and P. Portugal, "Controlling multi-switch networks for prompt reconfiguration," in *9th Int. Workshop on Factory Communication Sys. (WFCS)*, May 2012.

[8] M. Ashjaei, M. Behnam, L. Almeida, and T. Nolte, "Performance analysis of master-slave multi-hop switched ethernet networks," in *8th IEEE Int. Symposium on Ind. Embedded Systems (SIES)*, June 2013.

[9] M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte, "Dynamic reconfiguration in multi-hop switched ethernet networks," in *6th Workshop on Adaptive and Reconfigurable Embedded Systems*, April 2014.

[10] R. Marau, L. Almeida, P. Pedreiras, K. Lakshmanan, and R. Rajkumar, "Utilization-based schedulability analysis for switched ethernet aiming dynamic qos management," in *15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2010.

[11] J. Silvestre-Blanes, L. Almeida, R. Marau, and P. Pedreiras, "Online qos management for multimedia real-time transmission in industrial networks," *IEEE Trans. on Ind. Electronics*, vol. 58, no. 3, Mar 2011.