Dynamic Reconfiguration 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*—The FTT-SE protocol provides adaptive real-time communication on Ethernet networks. To assure a continued real-time behavior, FTT-SE integrates admission control with a quality-of-service mechanism, which screen all adaptation and reconfiguration requests, accepting only those that do not compromise the system timeliness. The adaptability and reconfigurability have been deeply studied in the case of single switch FTT-SE architectures, whereas the extension of that for the multi-hop FTT-SE architecture was not yet investigated. Therefore, in this paper we study the challenges of enabling dynamic reconfiguration in multi-hop FTT-SE networks, we propose two methods (one centralized and one distributed) and we present a qualitative comparison between them.**

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 and computations.

The limitations imposed by the simple use of COTS switches have been addressed in other works [1]. Some 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, as well as AFDX [6] optimized for quick forwarding in an event-triggered environment. However, these switches are configured in ways that are not suited for dynamic real-time systems 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 [7].

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.

Recently, three different approaches were proposed to extend the protocol for multi-hop communication. The first architecture [8] uses a single master to control the traffic transmission in the whole network. In the second architecture [9], multiple masters, each of which is attached to a switch, coordinate the traffic, whereas in the third architecture [9] several masters coexist, too, but each controls the traffic in a group of switches. The third architecture, named *hybrid architecture*, performs better in terms of bandwidth utilization [9].

The adaptability and reconfiguration in single switch FTT-SE networks is well studied and a related middleware was proposed [10] that uses a linear time-complexity online admission control [11]. Dynamic QoS management was also addressed in the context of a multimedia real-time application [12]. In this paper, we focus on the hybrid multi-hop architecture and we investigate the challenges of providing adaptability and reconfiguration therein. We propose two methods to perform the online reconfiguration. Finally, we present a qualitative comparison of the proposed methods.

II. Multi-Hop FTT-SE Protocol

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.

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*.

According to the FTT-SE protocol, 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.

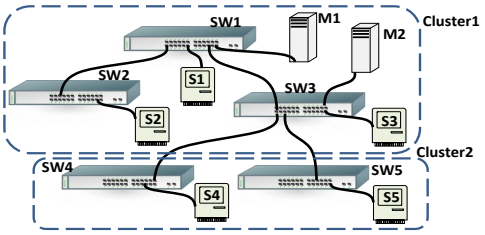

Fig. 1. The Hybrid Architecture

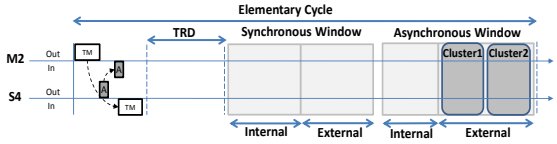

Fig. 2. The Elementary Cycle in the Hybrid Architecture

The scheduler in the master node computes the activation instants (cycle) 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).

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 [9] for more details about the hybrid architecture.

III. Problem Description

The work in [10] presented a middleware architecture that supports dynamic reconfiguration and QoS management with hard real-time guarantees in the context of the single switch 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 basic 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 is triggered by a slave node. The second step is the admission control, to verify the feasibility of the proposed changes. In this step a utilization or 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 this paper we adapt this online reconfiguration procedure to a multi-hop FTT-SE architecture. The major issues include the admission control and QoS manager, transmission of the request and update messages and the procedure for updating the respective databases.

IV. Dynamic Reconfiguration Methods

As mentioned above, the hybrid FTT-SE architecture comprises two types of traffic: internal and external. The former one is local to a cluster, scheduled solely by one master. Thanks to the traffic isolation features of FTT-SE, which dedicates disjoint and independent windows to the internal and external traffic, the reconfiguration becomes a local problem, and in so being, the solutions previously developed for the single-hop FTT-SE networks ([10], [11], [12]) can still be used.

However, the case changes radically when dealing with the external traffic. In this case the reconfiguration always involves all the clusters of the network because the external traffic window is a shared resource managed cooperatively by all masters. Therefore, the steps involved in the reconfiguration must be instantiated in all masters in parallel and with a tight synchronization, to make sure that the SRDBs of each one are consistently updated over time and therefore that the scheduling decisions are correct and consistent. In this paper we propose two methods to carry out the network reconfiguration: *centralized* and *distributed* reconfiguration. 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. In this section we describe both methods in detail and we perform a qualitative assessment of them.

A. Centralized Reconfiguration

The SRDB of all master nodes contains the information of the internal and external messages. Both internal and external messages are scheduled in parallel and independently by all nodes. A correct behavior requires a network-wide consistent scheduling of external messages, i.e., all the masters must schedule exactly the same external messages in the same external windows. The parallel scheduling of external messages creates an end-to-end path through the involved switches that allow such messages to reach their destinations in one single cycle (actually, within the respective external traffic window). Attaining this behavior requires:

- The use of a deterministic scheduling algorithm;
- Synchronization among masters;

- A consistent representation of the external traffic in the SRDB.

The first two requirements are implicitly verified, since FTT-SE uses deterministic scheduling algorithms and the whole multi-hop framework depends on a proper synchronization, for which methods have already been proposed (e.g. [9]). However, the third requirement implies that all changes to the external messages must be carried out on all master SRDBs and must take effect synchronously.

As mentioned before, a reconfiguration involves several steps. Some of these steps, like admission control and QoS re-distribution, may encompass computationally intensive operations ([11], [12]). 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*, for obvious reasons.

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, similarly to the request/update messages described in Section III, which handle the communication between slaves and their master. 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.

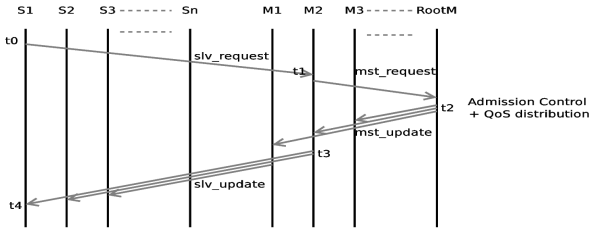

Fig. 3. Centralized Reconfiguration Event Sequence

Slave 1 requests a reconfiguration at $t = t_0$ in the same way as described in [11], 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, in the same way as in the single-hop approach, via *slv_update* messages.

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. As discussed above, 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

An alternative approach consists in distributing the responsibility of the admission control and QoS management over all master nodes. Reconfiguration requests are forwarded to all masters, which concurrently evaluate their feasibility, compute the necessary updates to the SRDBs and communicate them to the respective slaves, to synchronize the NRDBs.

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 IV-A is assumed.

S1 sends a reconfiguration request to M2 at $t = t_0$, in the same way as described in Section IV-A. 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 $t = t_4$.

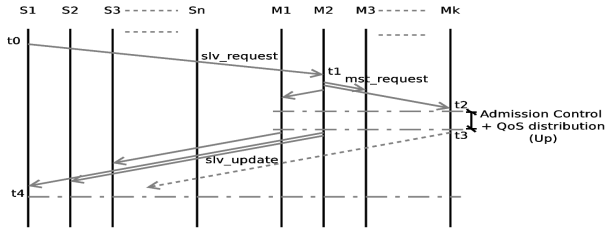

Fig. 4. Distributed Reconfiguration Event Sequence

Serialization of the requests is required, however it is rather complex. This will be addressed in future work.

C. Qualitative Comparison of the Methods

Centralized and distributed architectures are used in many contexts and their relative merits are well known. Nevertheless, the framework considered in this paper has some particularities that are discussed with some detail in this section. The overall comparison is presented in TABLE I.

Item	Centralized	Distributed
Processing Efficiency	+	-
Support Complex Admission Control and QoS	+	-
Concurrent Requests	+	-
Bandwidth	-	+
Fault Tolerance	-	+
Reconfiguration Response Time	=	=

TABLE I. QUALITATIVE COMPARISON

The centralized approach is more efficient in terms of processing, since the Admission Control and QoS algorithms are executed only in one node, while in the distributed method they are executed in all masters, producing exactly the same results. This permits e.g. having more complex admission control and QoS algorithms with a minimum system impact, since the additional complexity affects only a single node.

Reconfiguration requests occur concurrently and asynchronously and may impact on the whole message set. For this reason, they must be processed one at a time. The centralized method facilitates such serialization, which is much harder to accomplish in a fully distributed scenario (implies an additional protocol among masters to reach consensus about the processing order of reconfiguration requests).

Regarding the network bandwidth utilization, the distributed approach is more efficient than the centralized one, since there is no need to synchronize the SRDBs.

With respect to the response time, there is no clear winner. As seen above, the centralized approach requires more bandwidth which at a first glance seems worse, because it enlarges the communication time. However, there are admission control and QoS algorithms that are extremely demanding. The centralized approach allows using a more powerful central node that may reduce the execution time of those algorithms and thus reduce the overall reconfiguration time.

Although the issue of fault tolerance was not discussed in this paper, it is clear that the centralized approach has a single point of failure (the root master), while the distributed one does not suffer from such impairment.

V. CONCLUSION AND FUTURE WORK

More and more real-time distributed systems are required to operate in dynamic environments, needing to adapt and reconfigure themselves dynamically. This paper extends previous work, addressing the support for such capacity in the context of multi-hop hybrid FTT-SE networks. To guarantee a continued real-time behavior, reconfiguration requires several steps: negotiation, admission control, QoS management and mode changes. In multi-hop networks these steps must be carried out in a tightly synchronized manner, to guarantee that all masters take consistent scheduling decisions over time. This paper proposes two methods, one called centralized, which delegates the decisions to a specific node, and another one called distributed, in which all master nodes evaluate the feasibility of the requests and compute the resource allocations in parallel. Also, a preliminary qualitative evaluation of both approaches is presented. Future work consists of determining analytic bounds for the key operations, perform a deeper performance evaluation study of both solutions giving a numerical evaluation and to implement both on real hardware.

ACKNOWLEDGMENT

This work is supported by the Swedish Foundation for Strategic Research via PRESS project. Also, it is partially supported by the Portuguese Government through FCT grant Serv-CPS PTDC/EEA-AUT/122362/2010 and FCT grants CodeStream PTDC/EEITEL/3006/2012.

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] I. Land and J. Elliott, *Architecting ARNIC 664 (AFDX) Solutions*, 2011.

[7] 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.

[8] 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.

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

[10] R. Marau, L. Almeida, M. Sousa, and P. Pedreiras, "A middleware to support dynamic reconfiguration of real-time networks," in *15th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2010.

[11] 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.

[12] 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.