Abstract—Nowadays, most complex embedded systems follow a distributed approach in which a network interconnects potentially large numbers of nodes. One technology that is being increasingly used is Switched Ethernet, but real-time variants of this protocol typically limit scalability. In this paper, we focus on the scalability of the Flexible Time Triggered communication over Switched Ethernet (FTT-SE), which has been proposed to support hard real-time applications in a flexible and predictable manner. Moreover, time-triggered and event-triggered communication methods are supported in this protocol. FTT-SE has already been explored and investigated for small scale networked applications. In this paper we address the protocol scalability and suggest three different solutions with a qualitative assessment.

I. INTRODUCTION

Networked Embedded Systems (NESs) are becoming large scale distributed systems that are growing in complexity due to the increasing number of networked nodes and the rising amount of shared data [1]. Therefore, the nodes need to support more functionality and exchange higher amounts of data, which add more complexity to these systems. The situation becomes more complicated when it is also desired to have a fully configurable system at design time and the consequent need for run-time adaptation. Current solutions lack an adaptation for such changes and cannot utilize the computational and communication resources. One protocol that supports such adaptation is Flexible Time-Triggered communication over Switched Ethernet (FTT-SE) [2]. FTT-SE merges the predictability and synchronization of the time-triggered paradigm [3] with the operational flexibility needed for on-line adaptation. However, the scalability of FTT-SE networks was considered in a limited way in previous works (assuming a simple network architecture consisting of a single switch) [4], [5]. In this paper we address the scalability of FTT-SE and provide a preliminary qualitative assessment of three possible approaches. The first solution is based on a single Master scheduling the whole network. The other two approaches are based on Masters scheduling different segments. Their properties are summarized in a table at the end of the paper.

II. FTT-SE OVERVIEW

The advent of Switched Ethernet makes it possible to handle parallel switching paths, segment the network and provide isolated collision domains. However, to meet hard real-time requirements additional mechanisms are needed. One recent technique that was proposed to provide hard real-time communication is the Flexible Time Triggered Switched Ethernet (FTT-SE) [6]. This protocol provides flexible and deterministic real-time communication services with dynamic Quality-of-Service (QoS) management and a centralized scheduling mechanism.

FTT-SE uses a Master/Multi-Slave approach in which the Master node is responsible to manage and coordinate the communication activities. The Master implements the centralized scheduling concept, QoS management and on-line admission control. The Master also schedules the message transmissions on-line and broadcasts the schedule to the network elements by sending periodic control messages called Trigger Messages (TMs). The time interval between two consecutive TMs is called Elementary Cycle (EC). At the beginning of each EC, slaves decode the received TM and transmit their messages if there is any message transmission scheduled for them in the current EC. The scheduling in the Master is carried out ensuring that only messages that can be transmitted within each EC are actually scheduled for that interval. Messages that do not fit in one EC are scheduled in subsequent cycles [6].

III. SCALABILITY APPROACHES TO FTT-SE

In order to scale up the FTT-SE protocol to cope with large networks, we propose the following three different architectures based on interconnecting multiple common off-the-shelf (COTS) switches.

A. Architecture 1

The first and simpler approach is to allow the network to grow by connecting multiple COTS switches together, but still keeping a single Master that schedules the whole traffic in the network (Figure 1). This already allows increasing the number of connected nodes with respect the single switch solution, but still suffers from scalability limitations related to the use of a single Master. The fact that the Master synchronizes the whole network, facilitates consistency and increases scheduling capabilities by supporting offsets of traffic streams. However, the scheduling becomes very complex when the network grows. The EC must be long enough to tolerate the potentially long delays affecting the propagation of the TMs as well as the messages sent by the nodes (long turnaround time). This establishes a compromise situation, a longer EC...
leads to poor timing resolution and potentially poor bandwidth efficiency, while a short EC might not be enough to exchange data among the farthest nodes and also increases overhead. Moreover, the single Master is also a single point of failure and, for critical applications, a backup mechanism is needed.

In this architecture, since the switches are connected directly, a Spanning Tree Protocol (STP) must be used to eliminate cycles in forwarding the messages. As a result, the topology becomes a logical tree despite possibly being a physical mesh. Depending on the location of the Master, the network tree might be symmetric or asymmetric. The best case concerning the forwarding delays between the Master and the nodes is a completely balanced tree in which the Master is located at the root. On the other hand, the worst case occurs when the Master is on a leaf node (e.g., if the Master is connected to Switch 2 or 4 in Figure 1).

B. Architecture 2

The second approach is to assign one Master to each switch in the network (Figure 2), which schedules the traffic generated in the nodes attached to the switch. This architecture needs a mechanism to contain the dissemination of each TM to the respective switch, only, which can be done using VLANs or multicast groups.

Similarly to the previous architecture, since the switches are directly connected to each other, a spanning tree protocol must be used to resolve the existence of multiple paths.

In this architecture, two kinds of messages can be distinguished with respect to their source, as seen by each Master.

- **Local** messages generated by nodes attached to the switch and thus scheduled by the respective Master. These messages are transmitted under control of the local Master and thus following the timings of the FTT-SE protocol.
- **External** messages generated by nodes attached to other switches but directed to this switch, either crossing it or targeted to a local node. These messages are transmitted without control of the local Master and thus they may generate interference with the local traffic. On the other hand, they are transmitted as soon as possible, delayed by the local traffic, only.

Two different options might be considered in this architecture depending on whether there is synchronization among the several Masters, concerning the EC structure and message scheduling (global scheduling). These options are called synchronous architecture and asynchronous architecture, respectively.

1) **Using global synchronization:**
   - Needs setting up a synchronization mechanism;
   - Scalability is limited by the global synchronization mechanism;
   - External messages are expected and accounted for by each Master, which allows controlling interference precisely. In this case, a multi-hop message (crossing several switches) goes from sender to receiver in one single EC, similarly to Architecture 1.

2) **Without global synchronization:**
   - More scalable and more flexible (no need for any synchronization mechanism);
   - Easy to set up and to deploy;
   - Interference of external messages is unavoidable since they may arrive at any time, irrespectively of the FTT-SE windows for each kind of traffic, interfering either on the local TM as well as on the remaining local messages. The impact of this interference can be attenuated if a sufficient interval is reserved in each EC for external messages.

C. Architecture 3

The third approach (Figure 3) includes a gateway between each two directly connected switches, keeping the one Master per switch, which performs scheduling and sends trigger messages (TMs) to its connected nodes. Gateways act similarly to normal nodes concerning receiving and sending messages.

When a node is sending a message to another node connected to a different switch, a path (route) is established and the message is sent to the gateway in the path connected to the same switch as the sender node. If the path traverses multiple switches, a message must be scheduled in each of them
to transfer the information between gateways until the final destination. A routing protocol (e.g., spanning tree) is required in the network to find routes across switches if multiple paths exist. Note that, conversely to the previous 2 architectures, the spanning-tree protocol of the switches cannot be used.

A consequence of the use of gateways is that all multi-hop messages are buffered inside them while waiting to be scheduled within each switch by the respective Master. Thus, from the scheduling point of view, these messages can be seen as sequences of local messages scheduled independently. Moreover, the use of gateways also eliminates the uncontrolled interference that could occur in the previous architecture.

However, the decoupling of message forwarding in each switch within a path implies a longer waiting time in each hop leading to a longer end-to-end delay when compared to the previous architectures.

In this approach there can also be two cases depending on whether the EC and the schedulers in all Masters are synchronized or not. Again, with synchronization, the construction of the global schedule is more complex and there is an implicit limitation on scalability and flexibility. On the other hand, there is the possibility to control the relative offsets of each message in a path, allowing to reduce the end-to-end delay and the respective jitter. Without synchronization such offset control is not possible and there can be a variable delay in each hop between one EC and the message period, which accumulates through the path, leading to very long end-to-end delay and jitter.

IV. Qualitative Assessment

In this section we define the criteria that we will use to compare the referred three approaches. Beyond the obvious scalability criterion, we will also consider synchronization and isolation among Masters, Masters complexity, operational flexibility, cost, end-to-end latencies, efficiency of using the network bandwidth, routing and optimization.

In what concerns synchronization, we mean that all schedulers in all Masters are aware of all communication requirements and produce compatible schedules every EC (global scheduling). In this aspect, Architecture 1 is implicitly synchronous while architectures 2 and 3 require an appropriate protocol to achieve synchronization, if desired.

When multiple Masters are used, they can also be more or less isolated with respect to the transmission of the associated messages. While architecture 3 enforces a strict isolation by using gateways, architecture 2 just relies on the use of VLANs or multicast domains to confine the propagation of the TM of each Master but local and external messages can still interfere. This problem does not apply to architecture 1.

With respect to Master complexity, it is inherently related to the size of the scheduling problem. Therefore, architecture 1 implies the most complex Master since the scheduler is monolithic. On architectures 2 and 3 the complexity depends on whether synchronization is used. Without synchronization, the scheduling problem is broken into a set of independent smaller problems leading to lower Master complexity. With synchronization, the scheduling in each Master includes local and external messages, the latter requiring global scheduling, thus leading to an intermediate complexity.

In terms of consistent updates we consider the easiness of consistently changing operational parameters of the system at run-time. In this case, architecture 1 is the most flexible in the sense that all the system operational information, such as the communication requirements, is concentrated in one single node, the Master. In the other approaches, consistent changes in parameters that are distributed among several Masters require an agreement mechanism, which implies longer latencies and thus less operational flexibility.

For cost, architecture 1 is naturally the least expensive due to the single Master. Architecture 2 presents a higher cost due to the multiple Masters while architecture 3 is the most expensive one due to the addition of the gateways.

Concerning the end-to-end latencies there are two main aspects to consider, the communication load and the synchronization of the multiple Masters when they exist. Architecture 1 is essentially dependent on the load. Architecture 2 is also dependent mostly on the load, with a minor impact of the Masters synchronization. On the other hand, this last aspect becomes the key factor in architecture 3 in which synchronization with non-optimized message offsets or the lack of synchronization may lead to rather long end-to-end latencies because of the chained transmissions across the Masters, independently of the load.

With respect to bandwidth efficiency we consider the effective usage of the network bandwidth during each EC. This is particularly influenced by the turnaround time since messages are scheduled in a given TM until they are disseminated through the network. Architectures 1 and 2 are
the ones that suffer the most from the turnaround because of the messages that cross several switches, thus being the least efficient. In architecture 3, the turnaround considered in the Masters scheduling is always within 1 switch, thus leading to a potentially better use of the bandwidth.

About *routing*, architectures 1 and 2 do not need it and can rely on the lower level spanning-tree protocol embedded in the switches. Conversely, architecture 3 needs a routing protocol.

Finally, in what concerns *scheduling optimization*, we consider the impact of message offsets. In the presence of global synchronization, all approaches support this kind of optimization. However, in the absence of synchronization, in both architectures 2 and 3, we lose the ability to control the relative offsets of the messages that cross multiple switches reducing the scheduling optimization capabilities.

Table 1 shows a summary of the main features of each architecture.

### Table 1

<table>
<thead>
<tr>
<th></th>
<th>Archit. 1</th>
<th>Architecture 2</th>
<th>Architecture 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sync</td>
<td>Async</td>
<td>Sync</td>
</tr>
<tr>
<td><strong>Synchronization</strong></td>
<td>Implicit</td>
<td>Needed</td>
<td>No</td>
</tr>
<tr>
<td><strong>Isolation among Masters</strong></td>
<td>Single Master</td>
<td>VLANs</td>
<td>VLANs</td>
</tr>
<tr>
<td><strong>Masters complexity</strong></td>
<td>High</td>
<td>Med.</td>
<td>High</td>
</tr>
<tr>
<td><strong>Consistent updates</strong></td>
<td>Easier</td>
<td>Harder</td>
<td>Harder</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Lower</td>
<td>Med.</td>
<td>High</td>
</tr>
<tr>
<td><strong>End-to-end latencies</strong></td>
<td>$\propto$ load</td>
<td>$\propto$ load</td>
<td>$\propto$ load</td>
</tr>
<tr>
<td><strong>Efficiency of using the network bandwidth</strong></td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Routing</strong></td>
<td>Switch level</td>
<td>Switch level</td>
<td>FTT sched.</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Possible</td>
<td>Possible</td>
<td>No</td>
</tr>
</tbody>
</table>

#### V. Conclusions

In this paper, we presented our ongoing work on extending the scalability model for FTT-SE networks. We have proposed three different architectures for integrating largely distributed applications and FTT-SE protocol. We discussed the main features of each solution and we carried out a qualitative assessment and comparison of these solutions concerning different factors such as scalability, operational flexibility, Master complexity and end-to-end latencies, among other. We found that none of the 3 presented architectures is optimal, i.e., all have advantages and disadvantages, thus a hybrid solution including combinations of two or three of the presented approaches might give better results for certain applications depending on the respective system requirement. For example, we can replace the switch between each pair of gateways in architecture 3 by a group of several switches still controlled by a single domain Master. Thus, in such domains we obtain an architecture similar to architecture 1. This approach will aggregate more nodes in each domain, leading to lower number of domains and thus gateways. Consequently, the number of hops needed to traverse the network is also reduced. This will contribute to decrease the end to end latencies when compared to architecture 3. On the other hand, compared to architecture 1, this approach still increases scalability and decreases the complexity of the single Master.

Further evaluation of hybrid architectures as well as a quantitative comparison of those presented in this paper is part of on-going work. Moreover, future work will also address the specific case of support for resource reservation, which we expect to be more complex in architectures 2 and 3, given the distributed network resource managers, i.e., the Masters.

#### VI. Acknowledgment

This work was partially supported by the iLAND project, call 2008-1 of the EU ARTEMIS JU Programme and by the European Community through the ICT NoE 214373 ArtistDesign.

#### References


