Task- and Network-level Schedule Co-Synthesis of Ethernet-based Time-triggered Systems

Licong Zhang, Dip Goswami, Reinhard Schneider, Samarjit Chakraborty
Institute for Real-Time Computer Systems, TU Munich, Germany
licong.zhang@rcs.ei.tum.de, dip.goswami@tum.de, reinhard.schneider@rcs.ei.tum.de, samarjit@tum.de

***Abstract*—In this paper, we study time-triggered distributed systems where periodic application tasks are mapped onto different end stations (processing units) communicating over a switched Ethernet network. We address the problem of application level (i.e., both task- and network-level) schedule synthesis and optimization. In this context, most of the recent works [10], [11] either focus on communication schedule or consider a simplified task model. In this work, we formulate the co-synthesis problem of task and communication schedules as a Mixed Integer Programming (MIP) model taking into account a number of Ethernet-specific timing parameters such as interframe gap, precision and synchronization error. Our formulation is able to handle one or multiple timing objectives such as application response time, end-to-end delay and their combinations. We show the applicability of our formulation considering an industrial size case study using a number of different sets of objectives. Further, we show that our formulation scales to systems with reasonably large size.**

I. Introduction

Recently, several Ethernet based protocols are developed and adopted in real-time safety-critical domains. The examples are EtherCAT and Profinet [1] in industrial automation, AFDX [2] and TTEthernet [3] in avionics and the AVB networks [4]. The majority of these protocols are based on the original Ethernet standard IEEE802.3 and employ a *switched* network paradigm. That is, switches are used to connect the *end stations* via *full-duplex* links. A full-duplex link consists of two independent directed links and allows simultaneous transmission in both directions. Fig. 1 shows an example of such a network with four end stations and two switches.

In an Ethernet-based network, the end stations communicate by exchanging messages as Ethernet *frames*. The frames are forwarded link by link via the switches from the sender to receiver end station. In general, when the frames are forwarded to the same link simultaneously, they are stored in a queue and sent according to an output scheduler (e.g., strict priority). Thus, the queueing in switches results in a queueing delay which further depends on many factors such as the topology, traffic load and often introduces a considerably large delay. Such non-deterministic temporal behavior makes the traditional Ethernet-based networks unsuitable for safety-critical applications with stringent timing requirements. To address this problem, most of such protocols offer mixed-criticality services by dividing the traffic into different classes, applied to communication with different timing requirements. For example, Profinet has IRT, RT and NRT services in decreasing timing guarantee. Similarly, the traffic in TTEthernet is divided into TT, RC and BE traffic and in AVB networks Class A/Class B are differentiated with normal prioritized frames.

In the above context, the traffic class designed to provide most strict timing guarantee is the time-triggered traffic. Examples of this traffic class include the IRT in Profinet, the TT traffic in TTEthernet and the time-triggered traffic in the IEEE802.1Qbv standard still in development. The actual implementation of the time-triggered traffic class may vary from one protocol to another, but the basic idea is to schedule the

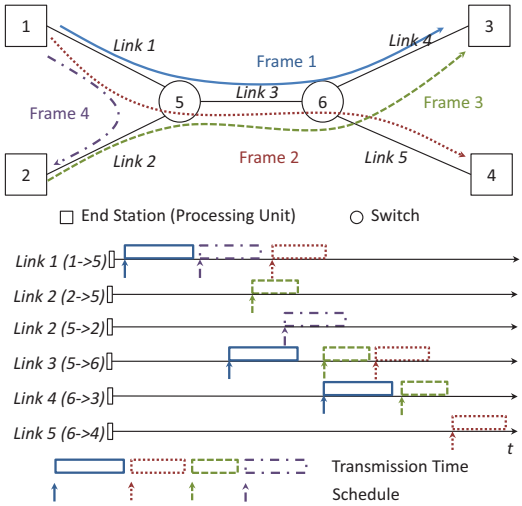

Fig. 1. An example of time-triggered traffic in switched Ethernet network.

frames so that the delays introduced by queueing in switches can be minimized or eliminated, thus achieving low network latency and jitter. Certain mechanisms are used to remove the influence from other traffic classes so that the frame transmission follows the pre-defined schedule. For example, Profinet uses a dedicated time period in a cycle to transmit IRT frames. All the frames in this traffic class are sent and forwarded in the network according to a pre-configured schedule. A schedule defines the exact time points when the end stations and switches send the frames on the Ethernet links. As shown in Fig. 1, the schedules should be chosen such that the frame transmission times do not collide with each other and thus ensure deterministic temporal behavior with low latency and jitter. In this paper, we consider the time-triggered traffic. Communication of other traffic classes can be added to the system without influencing the timing properties of time-triggered communication. We present our method considering the time-triggered traffic in a general switched Ethernet network. Our method can be tailored to fit into specific protocols such as Profinet and TTEthernet.

In a distributed system, a number of functions (or applications) are performed by a combination of *tasks* running on the end stations and data exchange by the communication over the network. For example, a distributed control system might have its *sensing*, *computing* and *actuating* tasks mapped onto different end stations and sensing/actuation data is sent via the network. In such cases, the performance of an application depends on schedules of both tasks and communication. Since optimal communication schedules may not necessarily result in optimal schedules at the application level, the above problem can not be simplified to that of the communication schedules alone. Moreover, the representation of design objectives is often itself a problem. For example, the design objective can be to minimize the overall latency of all applications while placing emphasis on certain ones. Such design requirements need multi-objective formulation.

978-1-4799-2816-3/14/$31.00 ©2014 IEEE

In this work, we aim to optimize the schedules for the tasks running on the end stations and for communication over the network considering one or multiple optimization objectives.

Related work: The related work in the above direction can be classified into three groups: (i) general time-triggered architecture [6] (ii) application level schedule synthesis and optimization of time-triggered (static) segment of FlexRay bus [7], [8], [9] (iii) optimization of time-trigged communication schedules in Ethernet-based network [10], [11], [12], [13]. Clearly, the works in direction (i) present general techniques [6] to deal with time-triggered systems without taking into account the Ethernet-based details which is the focus of our work. The recent works [7], [8], [9] in direction (ii) deal with both the network- [7] and application-level [8], [9] schedules taking into account the FlexRay specific details. However, FlexRay as a communication protocol differs considerably from a switched Ethernet network and works in [7], [8], [9] can not be applied in a straightforward manner. In direction (iii), [12] proposed a model formulation for the schedule synthesis for the Profinet IRT which is the time-triggered traffic pattern in Profinet. Similarly, [10] formulated the complete model of the time-triggered traffic in TTEthernet in SMT, which is later complemented by [11] for the bandwidth reservation of RC traffic. Further, [13] proposed an alternative schedule synthesis method of TTEthernet based on Tabu search, such that the deadlines of TT and RC messages are satisfied and end-to-end delays of RC messages are minimized.

Contributions: We propose a method for application-level schedule synthesis of distributed systems using time-triggered traffic in Ethernet communication. In this method, we formulate the scheduling problem as a Mixed Integer Programming (MIP) model. We further consider different optimization objectives and formulate a multi-objective optimization problem. We use Gurobi [16] to obtain a solution for the MIP formulation. Although there are a few recent works [10], [11] towards optimizing communication schedules in this context, our work focuses on application level optimization. Using an industrial size case-study, we illustrate various design aspects of such Ethernet-based time-triggered systems by considering different optimization objectives. To the best of our knowledge, this is the first work to consider the interplay between the task and communication schedules of a time-triggered Ethernet-based distributed system and optimize the schedules according to the (multi-)objectives at the application level.

II. Ethernet-based time-triggered systems

We consider a distributed system whose physical topology consists of a number of *end stations* connected by a switched Ethernet network. We denote the system as a graph $G(\mathcal{V}, \mathcal{E})$, where vertices $\mathcal{V}$ denote the set of network *nodes* (end stations and switches) and edges $\mathcal{E}$ denote the *full-duplex* Ethernet links. To differentiate between different types of nodes, we denote an end station as $v_i^e \in \mathcal{V}$ and a switch as $v_i^s \in \mathcal{V}$. A full-duplex link connecting two nodes v_m and v_n is denoted by $l_{m,n} \in \mathcal{E}$ and $l_{n,m} \in \mathcal{E}$, each representing a *directed* link from v_m to v_n and from v_n to v_m. In the distributed setup under consideration, we consider the following components which are described using an example shown in Fig. 2.

Application tasks: We define a task running on an end station as an application task. We consider periodic application tasks τ_i which are characterized by a tuple $\tau_i = \{\tau_i.p,\ \tau_i.o,\ \tau_i.e\}$ consisting of the period, offset and the WCET respectively. In the example shown in Fig. 2, there are five applications tasks τ_1 to τ_5 which are running in five different end stations.

Communication tasks: A communication task c_i can be defined as $c_i = \{f_i,\ c_i.tr,\ c_i.o,\ c_i.p\}$. Each communication

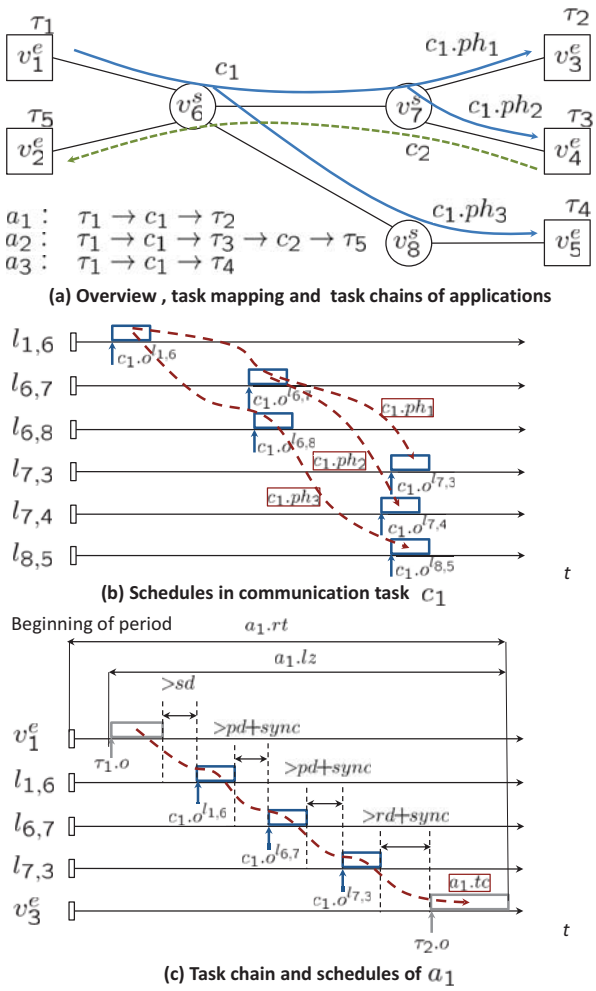

Fig. 2. Example of a distributed system of applications a_1 to a_3, application tasks τ_1 to τ_5 and communication tasks c_1 and c_2.

task is associated with an unique Ethernet frame with frame length $f_i.fl$. We define a *path* as a route from the sender to one receiver. For example, $c_1.ph_1$ is one path from v_1^e to v_3^e. Further, $c_i.tr$ denotes the *path tree* which consists of all the *paths* from a particular sender station to the receiver stations. A path tree may contain one path $c_i.tr = c_i.ph_1$ or multiple paths $c_i.tr = \{c_i.ph_1,\ c_i.ph_2,\ \ldots,\ c_i.ph_{\alpha_i}\}$, depending on whether the uni-cast or multi-cast operation is applied. For example, in Fig. 2 (a) and (b), path tree of c_1 contains three paths $c_1.ph_1$, $c_1.ph_2$ and $c_1.ph_3$. Furthermore, $c_i.o$ denotes the set of sending schedules of frame f_i on all the Ethernet links in the path tree. A single schedule on link $l_{m,n}$ is represented as $c_i.o^{l_{m,n}}$. In Fig. 2, the communication task c_1 involves six Ethernet links and there is one schedule for each link as indicated in Fig. 2 (b). Finally, $c_i.p$ denotes the period of the communication task. Note that as oppose to the traditional task, the communication task here represents the whole process of sending, forwarding and receiving a frame over the network.

Applications: An *application* a_i is a collection of certain application tasks and communication tasks that perform an independent function. An application is characterized by the tuple $a_i = \{a_i.tc,\ a_i.p,\ a_i.rt,\ a_i.lz\}$. Here, $a_i.tc$ is the *task chain*, containing the application and communication tasks that constitute the application in the temporal order. Three applications a_1, a_2 and a_3 are shown in Fig. 2 (a). We further denote the tasks contained in the task chain as $a_i.t_j$, where $1 \leq j \leq \beta_i$ with $a_i.t_1$ and $a_i.t_{\beta_i}$ denoting the first and last

task respectively. Each task in the task chain either represents an application or a communication task. Fig. 2 (c) shows the details of task chain $a_1.tc$. Since there are three tasks in a_1, $\beta_1 = 3$ with $a_1.t_1 = \tau_1$, $a_1.t_2 = c_1$ and $a_1.t_3 = \tau_2$. $a_i.p$ denotes the period of the application and all the application and communication tasks share the same period with the application. That is, $\tau_1.p = c_1.p = \tau_2.p = a_1.p$ in the above example. Finally, $a_i.rt$, $a_i.lz$ are the *response time* and the *end-to-end latency*, which represent the time between the start of the period and the end of last task and the time between the start of the first task and end of last task. The significance of these two parameters will be discussed in details in the following sections.

Additional timing restrictions: In this paper, we consider all the end stations as single processors running under a time-triggered non-preemptive scheduling scheme. Such scheduling scheme is common in safety-critical domains in particular. If an application task finishes its execution, it needs some time before the data can be packed in frames and sent on the network. This time interval has an upper bound which is denoted by sd. Similarly, once a frame arrives at an end station, it needs some time to get unpacked and processed before the data can be utilized by the corresponding application task. We assume that this time has an upper bound denoted as rd. On the network, we use the general case where each Ethernet switch possesses a *dispatcher* and each frame arriving at a switch is *forwarded* according to a schedule. The maximum processing delay of a frame in a switch is bounded by pd. That is, pd is the time between reception of the last bit on the *input port* and the earliest possible transmission of the first bit on the *output port*. It should be noted that *store-and-forward* and *cut-through* mechanism [14] can be modeled by setting pd an appropriate value. We further denote the *bandwidth* as bw – time to send one bit on the link and *interframe gap* as ifg – minimal link idle time between the transmission of two consecutive frames. The *precision*, i.e., the maximum difference between the any two clocks in the system is denoted as $sync$. In this paper, all the schedules are referenced to the local time of the network nodes and we pad a $sync$ in the constraints with schedules from different nodes.

III. Constraints formulation

In this work, we present a framework to co-synthesize schedules for all the application tasks τ_i and communication tasks c_i such that some high-level objectives are optimized. We characterize the distributed system described in the previous section by the following set of constraints.

(C1) Collision free application tasks: This condition ensures that on every single processor, one application task is triggered only when the processor is idle, i.e., after the last task is finished. We define the set of all application tasks as $\mathcal{T}$ and that of those mapped on an end station v_m^e as $\mathcal{T}(v_m^e)$. This condition can be formulated as:

$$\begin{aligned} &\forall m, v_m^e \in \mathcal{V}, \forall i, j, i \neq j, \tau_i, \tau_j \in \mathcal{T}(v_m^e) \\ &\tau_i.p \times k_i + \tau_i.o + \tau_i.e < \tau_j.p \times k_j + \tau_j.o \\ &\textbf{or} \\ &\tau_j.p \times k_j + \tau_j.o + \tau_j.e < \tau_i.p \times k_i + \tau_i.o \end{aligned} \tag{1}$$

where

$$\forall k_i \in \left[0, \frac{LCM(\tau_i.p, \tau_j.p)}{\tau_i.p} - 1\right]$$

$$\forall k_j \in \left[0, \frac{LCM(\tau_i.p, \tau_j.p)}{\tau_j.p} - 1\right].$$

and $LCM(\tau_i.p, \tau_j.p)$ denotes the least common multiple of periods $\tau_i.p$ and $\tau_j.p$. In Fig. 2, $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ and $\mathcal{T}(v_1^e) = \tau_1$ for example. Although there is only one application task mapped on each end station in this example, there can be multiple applications tasks running on an end station as we will see in the experimental section. The constraint (1) is applicable to those scenarios.

(C2) Collision free communication tasks: To ensure there is no collision of frames being sent on a single directed Ethernet link, a frame can only start its transmission ifg time units after the last frame is finished. We define $\mathcal{C}$ as the set of all communication tasks and $\mathcal{C}(l_{m,n})$ as the set whose path trees contain link $l_{m,n}$. This condition can be formulated as:

$$\begin{aligned} &\forall m, n, l_{m,n} \in \mathcal{E}, \forall i, j, c_i, c_j \in \mathcal{C}(l_{m,n}) \\ &c_i.p \times k_i + c_i.o^{l_{m,n}} + f_i.fl/bw + ifg < c_j.p \times k_j + c_j.o^{l_{m,n}} \\ &\textbf{or} \\ &c_j.p \times k_j + c_j.o^{l_{m,n}} + f_j.fl/bw + ifg < c_i.p \times k_i + c_i.o^{l_{m,n}} \end{aligned} \tag{2}$$

where

$$\forall k_i \in \left[0, \frac{LCM(c_i.p, c_j.p)}{c_i.p} - 1\right]$$

$$\forall k_j \in \left[0, \frac{LCM(c_i.p, c_j.p)}{c_j.p} - 1\right].$$

In Fig. 2, $\mathcal{C} = \{c_1, c_2\}$, $\mathcal{C}(l_{6,8}) = \{c_1\}$ and similarly, $\mathcal{C}(l_{7,6}) = \{c_2\}$. Although there is only one communication task mapped on each link $l_{m,n}$ in this example, there can be multiple communication tasks sharing the same link. The constraint (2) is applicable to those scenarios.

(C3) Path dependency in communication: In a communication task, a frame can only be forwarded along the paths in the correct temporal order. We further represent the schedule $c_i.o^{l_{m,n}}$ on the link $l_{m,n}$ as $c_i.o^{l_{m,n}}[ph_j, q]$ or simply $c_i.o[ph_j, q]$, i.e., the q^{th} schedule in path ph_j. Here, $q = 1$ and $q = \gamma_{i,j}$ represent the first and last schedule. The data dependency of the communication can be formulated as:

$$\begin{aligned} &\forall i, c_i \in C, \forall j \in [1, \alpha_i], \forall q \in [2, \gamma_{i,j}] \\ &c_i.o[ph_j, q-1] + f_i.fl/bw + pd + sync < c_i.o[ph_j, q] \ . \end{aligned} \tag{3}$$

It should be noted that $f_i.fl/bw$ represents the transmission time of the frame f_i for a given bandwidth bw. In Fig. 2, let us consider the path $c_1.ph_1$. Here, $\gamma_{1,1} = 3$ and the three schedules $c_1.o^{l_{1,6}}[ph_1, 1]$, $c_1.o^{l_{6,7}}[ph_1, 2]$ and $c_1.o^{l_{7,3}}[ph_1, 3]$ should be in the correct temporal order satisfying the constraint (3). Note that $c_1.o^{l_{1,6}}[ph_1, 1]$, $c_1.o^{l_{1,6}}[ph_2, 1]$ and $c_1.o^{l_{1,6}}[ph_3, 1]$ represent the same schedule since all three paths share the same link $l_{1,6}$.

(C4) Data dependency in applications: Due to data dependency, the application and communication tasks in the task chain of an application have to be executed in the correct temporal order. In a task chain, two consecutive tasks must be one of the following cases: (i) both application tasks, (ii) an application task followed by a communication task (iii) a communication task followed by an application task. We define the set of all applications as $\mathcal{A}$. This condition can be formulated as follows:

$$\begin{aligned} &\forall i, a_i \in \mathcal{A}, \forall j \in [1, \beta_i - 1] \\ &\textbf{if } a_i.t_j = \tau_h \in \mathcal{T} \wedge a_i.t_{j+1} = \tau_g \in \mathcal{T} \textbf{ then} \\ &\quad \tau_h.o + \tau_h.e < \tau_g.o \end{aligned} \tag{4}$$

$$\begin{aligned} &\textbf{if } a_i.t_j = \tau_h \in \mathcal{T} \wedge a_i.t_{j+1} = c_g \in \mathcal{C} \textbf{ then} \\ &\quad \tau_h.o + \tau_h.e + sd < c_g.o[ph_u, 1] \end{aligned} \tag{5}$$

$$\begin{aligned} &\textbf{if } a_i.t_j = c_h \in \mathcal{C} \wedge a_i.t_{j+1} = \tau_g \in \mathcal{T} \textbf{ then} \\ &\quad c_h.o[ph_v, \gamma_{h,v}] + f_h.fl/bw + sync + rd < \tau_g.o \ . \end{aligned} \tag{6}$$

where ph_u is any path within the path tree and ph_v represents the path which is utilized by the corresponding application.

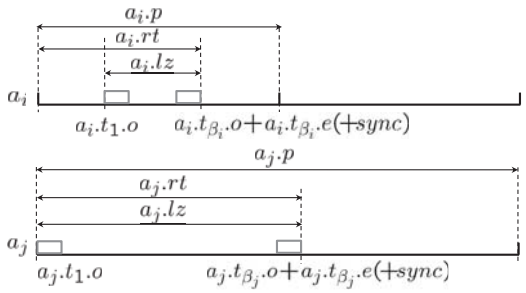

Fig. 3. Response time and end-to-end latency.

For illustration, let us consider the example in Fig. 2. For application a_1 in Fig. 2 (c), the application task τ_1 is followed by c_1 and the schedules $\tau_1.o$ and $c_1.o^{l_{1,6}}$ are constrained by (5). Similarly, c_1 is followed by τ_2 and the schedules $c_1.o^{l_{7,3}}$ and $\tau_2.o$ are constrained by (6).

(C5) Application response time constraint: The response time of an application a_i is given by:

$$a_i.rt = a_i.t_{\beta_i}.o + a_i.t_{\beta_i}.e(+sync) \ . \tag{7}$$

We denote the response time of an application as the time when the last task in the corresponding task chain is finished from the beginning of period. If observed from the local time of the sender station, the $sync$ can be omitted. Fig. 3 shows an example of response time of two tasks a_i and a_j. In a_i, $a_i.t_{\beta_1}.o \neq$ *beginning of period* possibly because of unavailability of (computation or network) resources. In many real-life time-triggered systems, the platform status information needs to be updated within a maximum tolerable time bound which is reflected by response time of an application. A hard constraint on maximum tolerable response time $a_i.rt_{max}$ can be enforced by adding the condition

$$a_i.rt < a_i.rt_{max}, \forall i, a_i \in \mathcal{A} \ . \tag{8}$$

(C6) Application end-to-end latency constraint: End-to-end latency of an application can be formulated as:

$$a_i.lz = a_i.t_{\beta_i}.o + a_i.t_{\beta_i}.e - a_i.t_1.o(+sync) \ . \tag{9}$$

Clearly, the end-to-end latency of an application is the time between the start of the first task and the finishing of the last task of its task chain. Similar to the response time, a maximum tolerable end-to-end latency can be enforced by the condition

$$a_i.lz < a_i.lz_{max}, \forall i, a_i \in \mathcal{A} \ . \tag{10}$$

Fig. 3 illustrates the difference between the response time and the end-to-end latency. When $a_i.t_{\beta_1}.o =$ *beginning of period*, $a_i.rt = a_i.lz$. The application a_2 in Fig. 3 shows such an example. On the other hand, when $a_i.t_{\beta_1}.o \neq$ *beginning of period*,

$$a_i.rt = a_i.lz + a_i.t_1.o \ .$$

One such example is the application a_1 in Fig. 3.

A. Constraint formulation as Mixed Integer Programming

Here, we discuss how the constraints and optimization objectives described in the previous section can be converted into a MIP problem. Towards this, each constraint should be represented as a single inequity or equation. The constraints (C3) to (C6) are already in the form of a single inequity. Since the constraints (C1) and (C2) are *either-or* conditions, they have to be converted by introducing a binary decision variable [15]. Thus, (C1) can be represented as

$$\begin{aligned} &\forall n, v_n^e \in V, \forall i, j, i \neq j, \tau_i, \tau_j \in T(v_m^e), \forall q \in [1,\ \lambda] \\ &\tau_i.p \times k_i + \tau_i.o + \tau_i.e < \tau_j.p \times k_j + \tau_j.o + y_q \times M_q \\ &\tau_j.p \times k_j + \tau_j.o + \tau_j.e < \tau_i.p \times k_i + \tau_i.o + (1 - y_q) \times M_q \ . \end{aligned} \tag{11}$$

where y_q denotes the introduced binary variable, λ represents the total number of possible collisions between application tasks and M_q is a sufficiently large constant. Similar conversion can be applied to (C2).

IV. Multi-objective optimization

Given the above representation of a time-triggered system, we consider three different classes of objectives. Let us consider a set of N applications $\mathcal{A}(obj) \in \mathcal{A}$.

(O1) Max/avg response time:

$$\begin{aligned} &\forall i, a_i \in \mathcal{A}(obj) \\ &\mathcal{A}(obj).rt_{max} = max(a_i.rt) \\ &\mathcal{A}(obj).rt_{avg} = \sum a_i.rt/N \end{aligned} \tag{12}$$

(O2) Max/avg end-to-end latency

$$\begin{aligned} &\forall i, a_i \in \mathcal{A}(obj) \\ &\mathcal{A}(obj).lz_{max} = max(a_i.lz) \\ &\mathcal{A}(obj).lz_{avg} = \sum a_i.lz/N \end{aligned} \tag{13}$$

(O3) Multi-objective optimization

$$\begin{aligned} &\forall i, obj_i \in \mathcal{OBJ} \\ &obj_M = \sum obj_i \times \omega_i \end{aligned} \tag{14}$$

where $\mathcal{OBJ}$ denotes the set of objectives in the form of O1 or O2 and obj_M denotes the weighted multi-objective function.

A. Objective formulation as Mixed Integer Programming

The implementation of objective function as the response time or end-to-end latency of a single application or the average value of multiple applications is straight forward. When the objective is the maximum value of several applications, the *minimax* problem can be formulated in MIP by introducing a continuous variable z [15]. Thus the minimax objective functions can be converted to

$$obj = z, \forall i, a_i \in \mathcal{A}(obj), a_i.rt \leq z, a_i.lz \leq z \ . \tag{15}$$

Such an objective will introduce an extra continuous variable z and N inequities, which must be added to the constraints. Further, a multi-objective function can be formulated as (14).

V. Experimental Results

In this section, we present the experimental results to show the applicability of our approach. We present an industrial size case study of a distributed system to illustrate the various design aspects considering different optimization objectives. Further, we present a scalability analysis and show that our formulation scales to systems with reasonably large size. We used Gurobi 5.10 [16] for solving the MIP model and all experiments are carried out on a computer with 1.87GHz CPU and 4GB memory.

A. Case Study

In the case study, we consider a distributed system consisting of 12 end stations connected via a switched Ethernet network. We explore four different network topologies as shown in Fig. 4. Further, 53 application tasks are mapped on the processors and 23 frames are sent on the network amongst which 7 are multi-cast. The application tasks and communication tasks constitute a total of 30 applications, with 5 of them having a task chain length of 5. The system parameters are bw = 100Mbps, ifg = 0.96us (12byte), $sd = pd = rd$ = 10us, $sync$ = 5us. The details of the system configuration is shown in Table I to Table III.

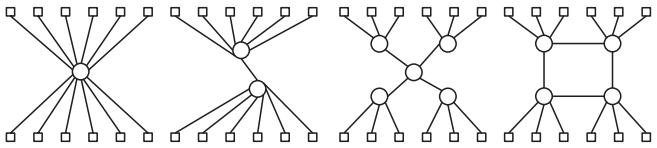

Fig. 4. Network topologies explored (from left to right): (a) Star, (b) Twin-star, (c) Tree, (d) Ring.

v_i^e	τ_i	$\tau_i.e$[us]	send c_i	receive c_i
v_1^e	τ_1,τ_2,τ_3,τ_4,τ_5	200	c_1,c_2,-,-,-	-,-,c_3,c_5,c_7
v_2^e	τ_6,τ_7,τ_8,τ_9,τ_{10}	500	c_3,c_4,-,-,-	-,-,c_{10},c_{17},c_{21}
v_3^e	τ_{11},τ_{12},τ_{13},τ_{14}	550	c_5,c_6,-,-	-,-,c_2,c_4
v_4^e	τ_{15},τ_{16}	350	c_7,c_8	c_1,c_{13}
v_5^e	τ_{17},τ_{18},τ_{19},τ_{20},τ_{21}	500	c_9,-,-,-,-	-,c_6,c_{11},c_{17},c_{18}
v_6^e	τ_{22},τ_{23},τ_{24},τ_{25},τ_{26},τ_{27}	400	c_{10},c_{11},c_{12},-,-,-	-,-,-,c_{14},c_{15},c_{18}
v_7^e	τ_{28},τ_{29},τ_{30},τ_{31},τ_{32}	300	c_{13},c_{14},-,-,-	-,-,c_8,c_{12},c_{19}
v_8^e	τ_{33},τ_{34},τ_{35},τ_{36},τ_{37}	500	c_{15},c_{16},-,-,-	-,-,c_9,c_{23},c_3
v_9^e	τ_{38},τ_{39},τ_{40},τ_{41}	500	c_{17},-,-,-	-,c_{16},c_{21},c_9
v_{10}^e	τ_{42},τ_{43},τ_{44},τ_{45}	300	c_{18},c_{19},-,-	-,-,c_{17},c_{21}
v_{11}^e	τ_{46},τ_{47},τ_{48},τ_{49},τ_{50}	500	c_{20},-,-,-,-	-,c_3,c_{19},c_{22},c_{18}
v_{12}^e	τ_{51},τ_{52},τ_{53}	600	c_{21},c_{22},c_{23}	c_3,c_{10},c_{20}

TABLE I. CONFIGURATION OF APPLICATION TASKS.

a_i	$a_i.p$[ms]	$a_i.tc$	a_i	$a_i.p$[ms]	$a_i.tc$
a_1	5	τ_1,c_1,τ_{15},c_7,τ_5	a_{16}	4	τ_{28},c_{13},τ_{16},c_8,τ_{30}
a_2	10	τ_2,c_2,τ_{13}	a_{17}	4	τ_{29},c_{14},τ_{25}
a_3	5	τ_6,c_3,τ_3	a_{18}	20	τ_{33},c_{15},τ_{26}
a_4	5	τ_6,c_3,τ_{37}	a_{19}	20	τ_{34},c_{16},τ_{39}
a_5	5	τ_6,c_3,τ_{47}	a_{20}	10	τ_{38},c_{17},τ_9
a_6	5	τ_6,c_3,τ_{51},c_{21},τ_{45}	a_{21}	10	τ_{38},c_{17},τ_{20}
a_7	10	τ_7,c_4,τ_{14}	a_{22}	10	τ_{38},c_{17},τ_{44}
a_8	10	τ_{11},c_5,τ_4	a_{23}	10	τ_{42},c_{18},τ_{21}
a_9	10	τ_{12},c_6,τ_{18}	a_{24}	10	τ_{42},c_{18},τ_{27}
a_{10}	10	τ_{17},c_9,τ_{35}	a_{25}	10	τ_{42},c_{18},τ_{50}
a_{11}	10	τ_{17},c_9,τ_{41}	a_{26}	10	τ_{43},c_{19},τ_{32}
a_{12}	5	τ_{22},c_{10},τ_8	a_{27}	10	τ_{43},c_{19},τ_{48}
a_{13}	5	τ_{22},c_{10},τ_{52},c_{22},τ_{49}	a_{28}	5	τ_{46},c_{20},τ_{53},c_{23},τ_{36}
a_{14}	10	τ_{23},c_{11},τ_{19}	a_{29}	5	τ_{51},c_{21},τ_{10}
a_{15}	10	τ_{24},c_{12},τ_{31}	a_{30}	5	τ_{51},c_{21},τ_{40}

TABLE II. CONFIGURATION OF THE APPLICATIONS.

In the experiment, we define the following objectives to be minimized (response time and latency are observed from local time of the first sender station):

- obj_1: Maximal response time of all applications $\mathcal{A}$.
- obj_2: Maximal response time of applications a_1 to a_5.
- obj_3: Maximal response time of applications a_1 to a_{10}.
- obj_4: Average response time of all applications $\mathcal{A}$.
- obj_5: Maximal end-to-end latency of all applications $\mathcal{A}$.

Depending on the higher-level goals one or more of the objectives above can be optimized. For platform/system status applications, the response time is an important parameter since they need to be completed as early as possible. Towards this, obj_1 and obj_4 are useful objectives. Often, a subset of applications are system status applications while the rest are usual applications. In such cases, obj_2 and obj_3 allows to minimize the response time of the relevant applications. Moreover, for many applications such as feedback control loops, the maximum end-to-end latency plays an important role. Therefore, obj_5 also have application level relevance. In general, it is possible to form various combinations of response time and end-to-end latency as an objective depending on the exact design requirements. Further, in many real-life scenarios, a single objective might not suffice. In such cases, our approach offers a multi-objective formulation. In the case-study, we explore the results considering each single objective and several multi-objective cases. In the case of multi-objective optimization, we assign equal weights ω_i. The optimization results are shown in Table IV for four different topologies as shown in Fig. 4.

c_i	$f_i.fl$[B]	c_i	$f_i.fl$[B]	c_i	$f_i.fl$[B]	c_i	$f_i.fl$[B]
c_1	64	c_7	100	c_{13}	100	c_{19}	100
c_2	80	c_8	100	c_{14}	150	c_{20}	64
c_3	64	c_9	64	c_{15}	150	c_{21}	64
c_4	80	c_{10}	64	c_{16}	100	c_{22}	64
c_5	80	c_{11}	64	c_{17}	100	c_{23}	64
c_6	100	c_{12}	64	c_{18}	100		

TABLE III. FRAME LENGTH OF COMMUNICATION TASKS.

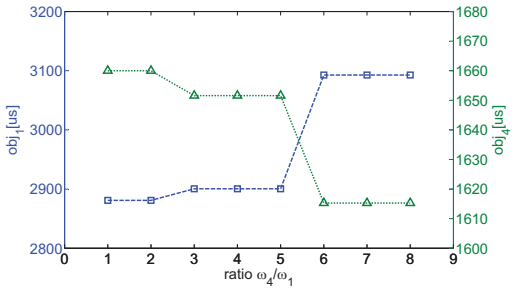

Fig. 5. Results for obj_1 and obj_4 in a multi-objective optimization with different weight ratio $\frac{\omega_4}{\omega_1}$.

Discussions: (i) Using our approach, it is possible to co-synthesize task and communication schedules in Ethernet-based time-triggered systems considering different application level objectives. For example, the definitions of obj_2 and obj_3 imply that $obj_2 \leq obj_3 \leq obj_1$, which is also reflected in our results. Similarly, the definition of obj_4 implies $obj_4 \leq obj_1$. (ii) In the case of single objective optimization, the first five rows of Table IV show the results for four different topologies. Clearly, the optimality was achieved for each individual objective (in bold font). For example, the result for obj_1 for tree topology is $obj_1 = 2880.96$us, which is the minimum among all the cases in first column. Naturally, such single objective cases often lead to non-optimal results for the others. E.g., in the case of tree topology, single objective optimization according to obj_2 and obj_3 leads to the minimal value of $obj_2 = 1342.48$us and $obj_3 = 2200$us respectively, but results in the undesirable results for obj_1 as 12395us and 18995us respectively. (iii) In the case of multi-objective optimization, it is often possible to achieve results close to the optimal one for all the objectives under consideration. The example of such optimization is illustrated using the combination of obj_1, obj_2 and obj_3 as shown in the 8^{th} row of Table IV. For the tree topology, we obtain $obj_1 = 3092.72$us, $obj_2 = 1342.48$us, $obj_3 = 2252$us. Clearly, all three obj_i are close or equal to the results in the single objective cases. It should be noted that the results of the objectives that are not optimized (not underlined) can vary under different solver configurations. This is because it is possible that there are multiple solutions to such an optimization problem and the results in the table show only one of them. However, the results of objectives that are considered for optimization are independent of the solver configuration.

Further, we investigate the influence of the weights on the optimization results. Fig. 5 shows the result in the case of multi-objective optimization of obj_1 and obj_4 for different weight ratios ω_4/ω_1. As expected, the weight ratio of the objective can influence the optimization result. A higher ω_4/ω_1 should imply a relatively lower obj_4 which is also reflected in Fig. 5.

From the experimental results we can see that our formulation allows the designer to generate schedules according to various application level objectives. It also offers the possibility to combine single objectives to form multi-objective optimization problems that could meet more complicated higher-level requirements. Moreover, our approach is applicable

obj	STAR					TWINSTAR				
	1 [us]	2 [us]	3 [us]	4 [us]	5 [us]	1 [us]	2 [us]	3 [us]	4 [us]	5 [us]
1	**2800.48**	2800.48	2800.48	2164.52	2800.48	**2820.60**	2820.60	2820.60	1920.03	2820.60
2	18995.00	**1256.24**	9995.00	5445.84	8944.76	11126.36	**1256.24**	9995.00	5147.35	6027.00
3	12800.00	2200.00	**2200.00**	4601.62	6795.00	11200.00	1650.00	**2200.00**	4600.77	8188.52
4	3006.48	**1256.24**	2206.00	**1590.88**	2550.24	3006.48	1570.36	2206.00	**1597.34**	2570.36
5	10236.48	7785.76	7785.76	4485.34	**1700.48**	12504.96	9166.24	9166.24	6295.11	**1700.48**
1+2	3006.48	**1256.24**	3006.48	2157.80	3006.48	3006.48	**1256.24**	2906.48	1801.53	3006.48.
1+3	**2800.48**	2200.00	**2200.00**	2141.23	2800.48	**2820.60**	2200.00	**2200.00**	1979.30	2820.60
1+2+3	3006.48	**1256.24**	2206.00	1996.64	3006.48	3006.48	**1256.24**	2260.00	2009.12	3006.48
1+4	**2800.48**	1650.24	2602.80	1630.01	2800.48	**2820.60**	1670.36	2602.80	1638.19	2820.60
1+5	2900.48	2900.48	2900.48	2037.11	**1700.48**	2900.48	2900.48	2900.48	1961.06	**1700.48**
1+2+3+4	3006.48	**1256.24**	2206.00	**1590.88**	2550.24	3006.48	**1256.24**	2206.00	**1597.34**	2467.56
1+2+3+4+5	3056.48	1356.00	2206.00	1656.46	**1700.48**	3056.48	1356.00	2206.00	1689.81	**1700.48**
obj	TREE					RING				
	1 [us]	2 [us]	3 [us]	4 [us]	5 [us]	1 [us]	2 [us]	3 [us]	4 [us]	5 [us]
1	**2880.96**	2880.96	2880.96	1954.47	2880.96	**2840.72**	2840.72	2840.72	2093.28	2840.72
2	12395.00	**1342.48**	9995.00	6087.15	9995.00	12395.00	**1299.36**	9995.00	5915.79	8186.04
3	18995.00	2200.00	**2200.00**	5104.39	10601.48	18995.00	2200.00	**2200.00**	5569.66	11904.52
4	3092.72	**1342.48**	2252.00	**1615.28**	2590.48	3049.60	**1299.36**	2229.00	**1604.08**	2570.36
5	12962.20	8624.68	8624.68	6010.64	**1740.72**	14464.28	6170.60	6170.60	4703.60	**1720.60**
1+2	3092.72	**1342.48**	2902.00	2062.89	3092.72	3049.60	**1299.36**	3049.60	2290.29	3049.60
1+3	**2880.96**	2200.00	**2200.00**	2027.25	2880.96	**2840.72**	2200.00	**2200.00**	2009.24	2840.72
1+2+3	3092.72	**1342.48**	2252.00	2077.51	3092.72	3049.60	**1299.36**	2229.00	1915.68	3049.60
1+4	**2880.96**	1690.48	2602.80	1659.99	2880.96	**2840.72**	1670.36	2602.80	1651.58	2840.72
1+5	2902.00	2703.28	2902.00	2021.08	**1740.72**	2900.48	2683.16	2900.48	1972.93	**1720.60**
1+2+3+4	3092.72	**1342.48**	2252.00	**1615.28**	2487.68	3049.60	**1299.36**	2229.00	**1604.08**	2570.36
1+2+3+4+5	3132.96	1356.00	2252.00	1684.95	1750.00	3076.60	1256.00	2229.00	1682.22	**1720.60**

TABLE IV. OPTIMIZATION RESULTS ACCORDING TO DIFFERENT OBJECTIVES.

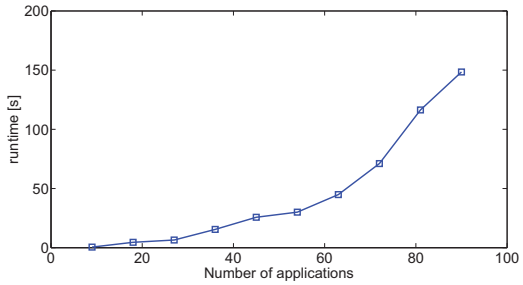

Fig. 6. Number of applications vs. runtime.

to any network topology and configuration of applications, application tasks and communication tasks.

B. Scalability analysis

To show the scalability of the proposed approach, we use a series of synthetic test systems in increasing size and measure the time required to solve the model. We construct 20 synthetic systems with different configurations for each system size from 9 to 90 applications. Initially, the systems employ a tree topology with three end stations and two switches. Subsequently, we add new applications and for each additional 9 applications, we add three end stations and one switch. All the applications consist of two application tasks and one uni-cast communication task. The periods of applications $a_i.p$ are chosen randomly from 4, 5, 10 and 20ms and the WCET $\tau_i.e$ are between 100 and 300us. Further, the frame lengths range from 80 to 200 bytes. Fig. 6 shows the average runtime of the overall system. Clearly, our approach scales up to 90 applications in a reasonable amount of time. In practice, the presented values are enough for many application domains such as automotive.

VI. CONCLUDING REMARKS

In this work, we presented a MIP formulation for the application-level schedule optimization problem for Ethernet-based time-triggered systems. Although the presented work mainly deals the application-level timing properties, our approach can be utilized to optimize a number of functional properties such as extensibility and sustainability. We plan to investigate in this direction in our future works.

REFERENCES

[1] R. Pigan, M. Metter, *Automating with PROFINET, 2nd edition*, Publicis Publishing, 2008.

[2] "664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network", ARINC, 2009.

[3] "AS6802: Time-triggered Ethernet," SAE International, 2011.

[4] "Media access control (MAC) bridges and virtual bridge local area networks," IEEE Computer Society, 2011.

[5] "http://www.ieee802.org/1/pages/802.1bv.html"

[6] H. Kopetz, G. Bauer, "The time-triggered architecture," *Proceesings of the IEEE*, vol. 91, iss. 1, 112–126, 2003.

[7] M. Lukasiewycz, M. Glass, J. Teich, P. Milbredt, "FlexRay schedule optimization of the static segment," *CODES+ISSS*, 2009.

[8] H. Zeng, M. Di Natale, A. Ghosal, A. Sangiovanni-Vincentelli, "Schedule optimization of time-triggered systems communicating over the FlexRay static segment," *Industrial Informatics, IEEE Transactions on*, vol. 7, iss. 1, 1–17, 2011.

[9] D. Goswami, M. Lukasiewycz, R. Schneider, S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," *DATE*, 2012.

[10] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," *RTSS*, 2010.

[11] W. Steiner, "Synthesis of static communication schedules for mixed-criticality systems," *ISORCW*, 2011.

[12] Z. Hanzalek, P. Burget, P. Sucha, "Profinet IO IRT message scheduling with temporal constraints," *Industrial Informatics, IEEE Transactions on*, vol. 6, iss. 3, 369–380, 2010.

[13] D. Tamas-Selicean, P. Pop W. Steiner, "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," *CODES+ISSS*, 2012.

[14] R. Seifert, J. Edwards *The All-New Switch Book: The Complete Guide to LAN Switching Technology,* Wiley Publishing, Inc., 2008.

[15] H. P. Williams, *Model Building in Mathematical Programming 4th edition,* John Wiley and Sons, LTD, 1999.

[16] "www.gurobi.com"