Generation of Communication Schedules Using Component Interfaces

Akramul Azim, Rodolfo Pellizzoni, Sebastian Fischmeister
Department of Electrical and Computer Engineering
University of Waterloo, Canada
{*aazim,rpellizz,sfischme*}*@uwaterloo.ca*

***Abstract*—With the growing demand in embedded systems, safety and non-safety critical parts are integrated together although guaranteeing safety is a hard problem to tackle due to the complexity of possible interactions between components involving communication. However, it is sufficiently recognized that separation of computation and communication can reduce the complexity of guaranteeing safety involved in interactions between components.**

In this work, we propose to use component interfaces derived from periodic resource supplies that can meet the demand of components experiencing bounded delays. The advantage of using interfaces is to provide minimal information of components without requiring the entire task specifications to generate a multi-mode communication schedule. We use integer linear programming to find assignments in generating schedules that are guaranteed to have low average mode-change delay. A video monitoring case study demonstrates the advantages on using our approach in generating communication schedules.

I. Introduction

Current real-time systems are inherently complex and built with heavy over-provisioning of resources to compromise between safety and functionality, because transient overloads can lead to failure due to missing deadlines. Some systems such as control systems can tolerate transient overloads (i.e., bounded delays) and thus would not need such high over-provisioning. This will then allow lower overall resource usage or running more functionalities. A complex real-time system is, in fact, a multi-mode system that facilitates high functionality, but also demands guarantees on safety requirements.

Computation and communication in safety-critical applications have to be completely separated and isolated. The conservation of complexity is a justification of separation of concerns. With the growing focus on safety-critical systems, the principle of separation is increasingly important for adaptive embedded systems. Adaptive and reconfigurable embedded systems that integrate safety-critical and non-critical components, or that integrate safety and adaptive behaviours require separation of concerns to control system complexity. Resource dependencies can be minimized by factoring out the reservation and consumption parts into separate programs. The approach of splitting the whole program into a minimal set of resource dependencies makes the programs easier to understand and analyze. They can then be joined in a deterministic manner through specified timed interactions such as timed interfaces.

978-1-4673-7929-8/15/$31

In an interface-based design, an interface describes how a component can be used. A well-designed component interface provides sufficient information to connect other components in a system [12]. Since an interface-based design of components in the system allows separation of concerns, scheduling mechanisms for computation and communication can differ. For example, in a computational component, a number of tasks execute on processors based on widely-used CPU scheduling algorithms such as EDF or RM [11], and message transmission occurs from one component to another using a real-time network scheduling algorithm such as static TDMA or state-based TDMA [8]. An efficient CPU scheduling algorithm is not necessarily suitable for networks because of the differences in characteristics. In a system that supports multiple communication modes, state-based scheduling has lower average mode-change delay than EDF [2], which schedules tasks based on deadlines. Moreover, state-based TDMA schedules are suitable for achieving predictability, reliability, and safety in real-time networks [8]. However, finding a state-based schedule for a time-triggered architecture to satisfy the timing requirements of tasks is challenging, because the scheduler has to consider not only the requirements on each component, but also the global requirements of system-wide behaviour, including messages transmitted on the networks.

In this paper, the goal is to generate state-based schedules using component interfaces to provide separation of computation and communication. Specifications of component interfaces are computed using local information of tasks. Tasks execution inside each component remain isolated from communication because of generating its schedules using the derived component interfaces. This separation of concerns and isolation of components provide increased safety in real-time systems.

II. Problem Statement

A demand bound function (*dbf*) and supply bound function (*sbf*) [11] represent timing requirements of the workload (i.e., tasks) and the resource supply to guarantee that sufficient amount of resource is available to satisfy the demand. The dbf(t) refers to the maximum resource demand during a time interval t by the tasks and the sbf(t) determines the minimum resource supply during t. For hard real-time systems, dbf(t) is no greater than the sbf(t). However, this condition does not need to hold [3] for systems that can tolerate bounded amount of delays.

The sufficient resource supply for a component with workload specifications converts into the demand to find a resource supply with additional workload. To satisfy the demand of a component workload, its interface has a particular resource supply derived using sbf(t) and dbf(t). These interfaces are particularly useful for combining multiple tasks requirements [11].

Recent work [3] on *sbf* and *dbf* shows how to find efficient resource supplies for workloads with a tolerable system delay, which is experienced by any task at a given time. This avoids over-provisioning of resource supply because it allows the sbf(t) to go below the dbf(t) for a bounded amount of delay, which is added to each component rather than its tasks.

A state-of-the-art research in safety-critical real-time systems is the analysis of co-scheduling of computational components and communication medium because of the necessity of separation of concerns and design optimization. Separation of concerns requires isolation of components and the communication medium to reduce the domino effect. The domino effect is a chain reaction that occurs when a small change causes a similar change nearby, which will then cause another similar change, and so on in linear sequence. Design optimization requires suitable scheduling policies to use for computation and communication, and schedules which are necessarily not the same.

Distributed real-time systems require reliable networking solutions for the exchange of time-critical data in addition to the processors. The Network Code framework [8] is one of the real-time Ethernet-based networking solutions that supports a language to describe state-based schedules, which are TDMA-based but have the ability to make on-the-fly decisions. This on-the-fly decision making capability in schedules facilitates low average delay while changing modes. Faster mode-change also depends on how the messages are scheduled. In state-based schedules, mode-change delay will become less if more messages of different states can be allocated to a time slot. This provides a state-based communication schedule with a less number of branches. State-based schedules also allow to transmit best-effort traffic if no real-time traffic is present. The generation of such communication schedules is discussed in [2], but co-scheduling of computation and communication is not discussed. Therefore, the delays introduced by computational units affect the scheduling of communication. This work aims to interconnect computational units through derived interfaces and generated communication schedules, even in the presence of delays introduced in components.

> ***Goal:*** *"Given a set of component interfaces with period, duration, a worst-case delay, generate a multi-mode distributed communication schedule which has low average mode-change delay".*

III. Background and System Model

A. Scheduling Computation

Scheduling computation has been extensively studied for real-time systems. Amongst them, the mostly used scheduling algorithms are EDF and RM. In [4], an extensive comparison between EDF and RM is shown. EDF is preferred over RM for better system utilization. Moreover, EDF performs reasonably well in transient overload situations. Transient overload occurs when the system needs more computing resource than available in order to be able to complete all tasks before their deadlines. In this paper, we assume EDF to schedule tasks of components of the system, leaving the extension to other scheduling schemes for future work.

The computation model consists of a periodic resource model and periodic workload for each of the computational components. For each computational unit, the periodic workload has a set of tasks and a set of messages for communication. Tasks of a component are scheduled using EDF scheduling. We use the following notations to model computational units (i.e., components):

- $\{C_j\}$ = a set of components where $j \in \mathbb{N}^+$
- $\{\tau_i^j\}$ = a set of tasks of component j where $i \in \mathbb{N}^+$
- p_i^j = period of task i of component j
- e_i^j = execution time of task i of component j
- δ_j^* = worst-case tolerable delay of component j
- $R_j(\lambda_j, \theta_j) = \theta_j$ resource supply in every λ_j period for component j
- $I_j(\lambda_j, \theta_j, \delta_j)$ = interface of component j characterized by period λ_j, duration θ_j, and worst-case delay δ_j

Supply and demand bound functions are used to determine schedulability under a particular scheduling policy. Supply and demand bound functions facilitate exact schedulability analysis during all time intervals, rather than sufficient analysis. A demand bound function takes the tasks in the workload (W), the scheduling policy (e.g., EDF), and the time interval (t) as input to determine the worst-case demand during a time interval of length t using Equation 1 as defined in [11].

$$\mathrm{dbf}(W, EDF, t) = \sum_{\tau_i \in W} \left\lfloor \frac{t}{p_i} \right\rfloor e_i. \tag{1}$$

In contrast. the supply bound function calculates the minimum resource supply during any time interval t. Using

Equation 4 from [11], it is possible to find the minimum resource supply (Figure 1) during any time interval of length t as:

$$\text{sbf}(t) = \begin{cases} (t - (\kappa+1)(\lambda-\theta)) & \text{if } t \in [\kappa_1, \kappa_2] \\ (\kappa-1)\theta & \text{otherwise} \end{cases} \quad (2)$$

with $\kappa_1 = (\kappa+1)\lambda - 2\theta$, $\kappa_2 = (\kappa+1)\lambda - \theta$, and κ as

$$\kappa = \max(\lceil (t - (\lambda - \theta))/\lambda \rceil, 1).$$

Note that, the value of κ is greater than or equal to 1.

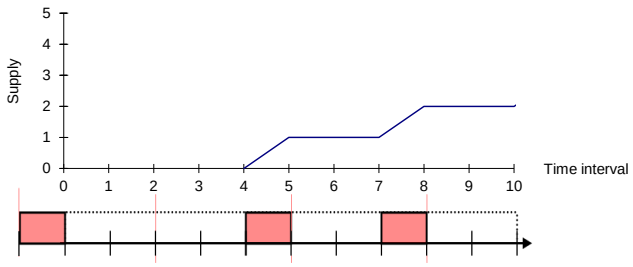

Fig. 1: sbf(t) for $\lambda = 3$ and $\theta = 1$

Using supply and demand bound functions, it is possible to derive bounds on the delays that the system can experience under EDF and a periodic resource supply [3]. For systems that cannot tolerate overloads, the sbf(t) is no less than the dbf(t) during all time intervals of length t until the hyperperiod. On the other hand, sbf(t) can become less than the dbf(t) for some t during all time intervals of length t until the hyperperiod. Fig. 2 shows an example of overload.

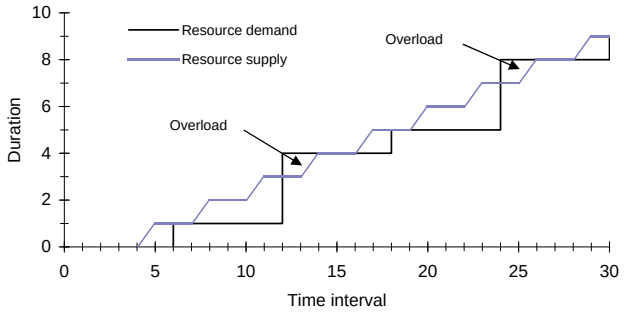

Fig. 2: An example of an overload ($\exists t$: sbf(t) $<$ dbf(W, t, EDF))

Example 1. *Consider a periodic resource supply $R(3,1)$, and a scheduling model $M(W, R, EDF)$ that has two tasks in the workload, $W = \{T_1(6,1), T_2(12,2)\}$. Fig. 2 shows the computed sbf and dbf. This workload is not schedulable with the given resource if the workload cannot tolerate any delay, because the supply is not always greater or equal than the demand. Slightly changing the specifications of the workload with a tolerable delay 2 time units makes the system feasible. The system is feasible because in a time interval of $t = 12$, the system can experience an overload, which becomes resolved in a time interval of $t = 14$.*

An overload can be characterized as the delay that a component can suffer [3]. A periodic transient overload occurs for a certain amount of time and is repeated after the hyperperiod. In this paper, we consider the periodic transient overloads which can be recovered. The time interval when the overload occurs is referred as an overload point. An overload point is associated with a recovery point when the overload becomes resolved. This delay is calculated as the difference between the overload point and the associated recovery point. The worst-case delay is the maximum delay that the system can suffer in the hyperperiod.

Component interfaces facilitate connecting computation and communication in a distributed system. These interfaces abstract the timing requirements of the workload of components. Using the supply and demand bound function analysis, interfaces can be derived, which provide the worst-case timing demand of all tasks in a component. We get the period and duration of an interface from the periodic resource supply that meets the demand of the workload in a component. Interfaces have an additional timing property, which is the worst-case delay that the component can experience.

Continuing Example 1 If the interface is I, then the specification of the interface is $(3, 1, 2)$ where period, execution time, and worst-case delay are 3,1, and 2 respectively.

B. Scheduling Communication

State-based schedules facilitate supporting different communication modes. Different case studies in various domains have demonstrated the advantages of using state-based schedules such as control theory, hybrid systems, hierarchical scheduling, and in general bursty demand models. The runtime framework for state-based schedules provides the necessary abstractions and execution entities. A state-based communication schedule is characterized by the following:

- V = a set of communication modes
- $\{\sigma_m^k\}$ = a set of messages in communication mode k where $m \in \mathbb{N}^+$ and $k \in \mathbb{N}^+$
- c_m^k = duration of message σ_m in mode k
- π_m^k = period of each message σ_m in mode k
- α_m instances for every message σ_m up to the end of communication round (i.e., hyperperiod)
- $v_s \in V$ denotes the current mode
- $V_d \subseteq V$ denotes the set of possible destination modes,
- B is the bandwidth assigned to scheduled messages,
- L is the link capacity, with $B \leq L$.

To illustrate the advantages of state-based schedules, let us consider an example of a Triple Module Redundancy (TMR) application. In a traditional setup for TMR, three different

sensors transmit independent samples of the same variable in consecutive messages σ_1, σ_2, and σ_3. A voting controller receives these messages and performs a majority vote to determine the final value. In a static TDMA configuration, the sequence of transmitted messages is determined only by the progression of time, and then the stations will always transmit the three messages, even if σ_1 and σ_2 are already decisive for the voting. However, a state-based schedule can perform a preliminary voting after receiving the first two samples, and if the voting is already decisive, then the slot associated to the third sample can be empty, leaving the medium available for other purposes such as background traffic.

C. Co-scheduling

This work interconnects EDF with state-based scheduling with an assumption that overloads can occur in components which cause delays in communication. This work also aims to provide isolation of components through their interfaces and allows to interconnect them using state-based communication schedules for better bandwidth utilization and increased safety in real-time systems. Fig. 3 shows an example model for two computational components with interfaces R_1 and R_2 connected through a shared communication medium. Each component has two tasks τ_1 and τ_2.

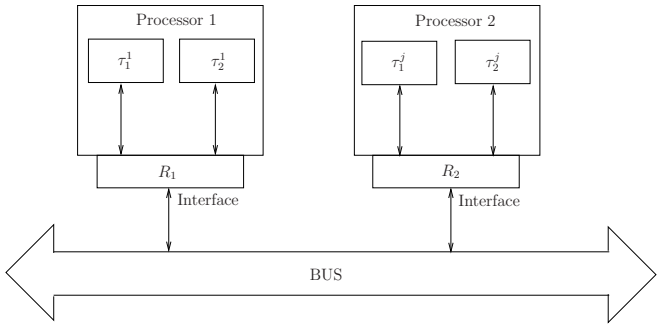

Fig. 3: An example model of isolation between computational components and the shared communication medium

IV. Specification and Design of Component Interfaces

This work introduces a model-driven approach to generate state-based schedules from interface specifications of components. The designer (1) specifies the details of components (i.e., tasks, resources, and their timing demand), transmission states and their interactions with timing requirements, (2) performs schedulability analysis and uses mechanisms to find out interfaces specifications from components information, (3) uses mechanisms not only to perform reachability and schedulability analysis, but also generates state-based schedules using components interfaces based on the application design and constraints.

A smart networked video monitoring system for collision avoidance is considered as a case study, which is effective in different transportation systems. A mining truck is an example of such system. Mining trucks are typically over 7 meters high and need a monitoring system to avoid collisions when running with small vehicles. The monitoring system consists of four cameras attached to wheels to transmit videos of surroundings. The cameras are labelled as front camera right (FCR), front camera left (FCL), rear camera right (RCR), and rear camera left (RCL). All cameras are connected through 1[Gbit/s] Ethernet link.

TABLE I: Timing requirements for computation of video components

	Nominal [ms]		Normalized	
Task	Comp. Time	Period	Comp. Time	Period
τ_d	5	32	1	6
τ_c	10	64	2	12

Computation specifications. In the case study, we assume that four sensors in the wheels are embedded along with the cameras. The cameras transmit two types of videos: standard quality (SQ) and high quality (HQ). When a sensor detects an object, the attached camera transmits HQ video. Sensors activate according to the movement of the truck: front sensors only activate if the vehicle is moving forward, and rear sensors only activate if the vehicle is moving backwards. We assume that the sensors perform a computation task to detect objects and the direction of the vehicle (τ_d), and the cameras perform a computation task to buffer and manage video (τ_c). Therefore, each video component has two computation tasks to schedule under the EDF scheduling policy. Table I shows the timing specifications of computation tasks for each of the video components with an assumption that deadline is equal to the period and the components can tolerate delays. In specifying requirements, we consider that the rate of task τ_c (30 frames per second) is higher than task τ_d to ensure that enough data is available. Assuming an atomic unit for the schedule equal to the minimum computation time for a task, we normalize the timing specifications to this time unit, and take the floor of the normalized period. The normalization may overestimate the actual requirement, but allows us to represent all the timing requirements as multiple integers of the time unit. We assume that the timing specifications allow video components to tolerate a delay of 2 time units in the worst-case, which is within the tolerable delay range of video broadcasting for a quality of service(QoS) as discussed in [6]. This case study assumes to transmit the raw video, leaving the use of encoder (e.g. MPEG-2) and different frames (i.e., I, P, B) as discussed in [5] for the future work. Instead of determining the tolerable delay based on a particular QoS, a control component can have a maximum delay tolerance depending on stability metrics as discussed in [3].

The display component is located near the vehicle driver to receive data periodically from the network. We assume that the display component has a single task that executes

TABLE II: Timing requirements for communication

	Nominal [μs]		Normalized	
FPS	Comm. Time	Period	Comm. Time	Period
15	7600	66700	6	48
30	7600	33350	6	24

TABLE III: Feasible modes with messages specifications

ID	Mode (FCR, FCL, RCR, RCL)	Utilization
1	(SQ,SQ,SQ,SQ)	0.5
2	(SQ,SQ,SQ,HQ)	0.625
3	(SQ,SQ,HQ,SQ)	0.625
4	(SQ,SQ,HQ,HQ)	0.75
5	(SQ,HQ,SQ,SQ)	0.625
6	(HQ,SQ,SQ,SQ)	0.625
7	(HQ,HQ,SQ,SQ)	0.75

faster than other tasks and therefore the computation time and period are set to 2 and 4 time units respectively. We assume that the display component can tolerate a delay of 1 time unit.

Communication specifications. Each of the four cameras operate in two states based on the frames per second (FPS). The SQ frames are transmitted by default and cameras switch to transmitting HQ frames once an object is detected. Moreover, either front cameras or rear cameras can transmit HQ frames after an object is detected.

The possible configurations of the communication is calculated as the cross products of the state machines. The timing requirements of different types of video transmission (i.e., HQ and SQ) as shown in Table II are calculated using pixel specification (640×480) of frames and network bandwidth (1Gbit/s). To find the feasible communication modes without considering computation delays, schedulability analysis can be performed using Equation 3. Table III shows feasible communication modes for the video case study where $\frac{B}{L} = 1$.

$$U(v_k) = \sum_{\sigma_i \in v_k} \frac{c_i}{\pi_i} \leq \frac{B}{L}. \tag{3}$$

A component has a workload consisting of tasks with timing specifications and can use an interface to specify the resource requirement. Under the EDF scheduling policy, a suitable resource supply (i.e., period and duration) is calculated using the characteristics of supply and demand bound functions [3]. To find a suitable resource supply for a component j, the workflow (shown in Fig. 4) is as follows: the utilization of the resource supply is kept the same as the workload utilization. The algorithm based on the supply calculation model [3] searches for resource supplies that have recovery points at δ_j^* distance from possible overload points (periods of tasks) and validates calculated resource supplies using supply and demand bound functions. After getting a set of valid supplies, one of them is chosen as an efficient resource supply based on application characteristics. An efficient resource supply becomes the first two parameters in the specification of an interface I. The delay that the component experiences because of using the suitable resource supply becomes the third parameter of the interface specification.

A suitable resource supply meets the specifications of the workload and avoids over-provisioning of resources. For example, if the display component has the workload $(3, 1)$, then a resource supply $(4, 2)$ does not meet specification because the worst-case delay is greater than 1. However, the resource supply $(2, 1)$ meets the specification and avoids over-provisioning of resources compared to $(1, 1)$. For each of the components located in the wheels, the interface specification is $(3, 1, 2)$, where 3 is the period, 1 is the duration, and 2 is the worst-case delay that the component can experience.

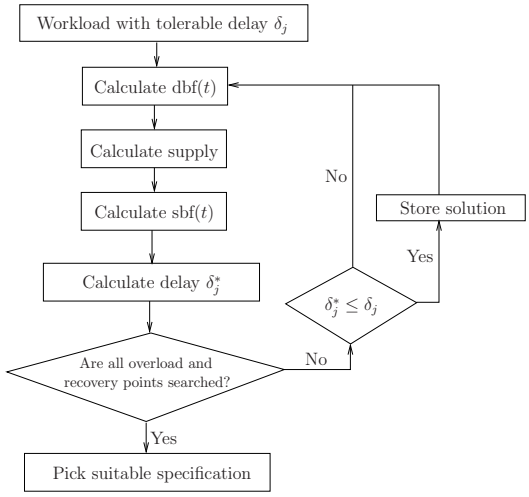

Fig. 4: Finding specification for component interfaces

A hierarchical scheduling in each component is also possible where different specifications in each level are combined together to get a single timing requirement. The resource supply turns into demand in each level and continues until no workload specification is left to combine. A compositional framework uses hierarchical scheduling to reduce complexity by abstracting subcomponents in a component. Therefore, using compositionality can reduce the number of interfaces.

V. Generation of State-Based Schedules

This paper discusses scheduling of computation and communication together but a separate scheduling policy is used for each of them. The EDF scheduling policy determines the schedulability and sequences of execution for computation. State-based schedules determine the sequences of execution for communication.

To generate state-based schedules, the first step is to extract internal state representations of communication for each of the components as discussed in the previous sections. We

refer the cross product of the states that are reachable as communication modes. A schedulability test is required to find communication modes that can meet timing requirements. To characterize a good state-based schedule, we define some properties.

Definition 1 (Overlap)**.** *An overlap is said to exist when a message m can be executed in different k modes on the same time slot l such that $l = 1, \ldots, \text{LCM}(\{\pi_m^k\})$.*

Definition 2 (Group)**.** *A group G_l^z in slot l is a subset of modes that have an overlap, where $z \in \{k\}$.*

With regard to overlaps and groups, a mode-change delay ϕ_{sd} is the time required for changing states at the end of time slot l for a given message m and a given transition. Given a group G_l^z that belong to a slot l, the mode-change delay ϕ_{sd} is either instantaneous (i.e., zero) from a source state $v_s \in G_l^z$ to a destination state $v_d \in G_l^z$ or the time until the next communication round starts if $v_d \notin G_l^z$,

$$\phi_{sd} = \begin{cases} 0 & \text{if } v_d \in G_l^z \\ \\ \text{LCM}\{\pi_m^k\} - j & \text{otherwise.} \end{cases} \tag{4}$$

A *good* state-based schedule has low average ϕ_{sd} to switch between modes at run time. A high number of overlaps in a state-based schedule results a low number of groups. This yields a lower average ϕ_{sd} delay because of faster switching to modes. Thus, generating state-based schedules is an optimization problem with an objective to minimize the number of groups.

Component interfaces provide additional information in generating state-based schedules for communication. In particular, delays in computation have an impact on communication and its schedules. Because, delays in computation cause the data to become available late for transmission and perhaps require the component to operate in a different communication mode to meet the timing requirements.

A. Optimization Model

The problem of finding an optimal state-based schedule with respect to minimizing the number of groups can be solved using integer linear programming (ILP). To do this, the model has a constraint on computation time of each message to obtain at least the required time slots. A boolean variable x_{ml}^k is set to 1 if a message m is allocated to a slot l in mode k, and 0 otherwise. A computation delay that results a message m in mode k to be transmitted late is characterized using δ_m^k. We discuss mechanisms to choose to a delay value in Section V-C. The number of overlaps for a slot l can be increased if the same message m is allocated to the previous slots in all modes. A constraint using a variable s_{ml}^k enforces this. To ensure the occurrence of overlaps earlier than later, a weight mapped to l is multiplied to each slot assignment x_{ml}^k in the objective function. A higher weight indicates a higher l value. Minimizing the sum of values of variables s_{ml}^k and $x_{ml}^k * l$ together ensures minimizing the number of groups in the schedule. We minimize the number of groups to find out the assignments of messages to slots to generate a state-based schedule that has low average mode-change delay.

- x_{ml}^k coefficient determines the usage of a time slot for $m \in N$, $l \in \{1, \ldots, \text{LCM}\{\pi_m^k\}\}$ and $k \in V$, where N represent the number of messages and V represent the number of modes. These coefficients are defined as follows:

$$x_{ml}^k = \begin{cases} 1 & \text{if message } \sigma_m \text{ uses the slot in mode } k \\ \\ 0 & \text{otherwise.} \end{cases}$$

- s_{ml}^k coefficient determines the overlaps. The variable s_{ml}^k is set to 1 if a message m is allocated a slot l in mode k, and other messages are allocated at the same slot l in modes except k.

$$s_{ml}^k = \begin{cases} 1 & \text{if } x_{ml}^k = 1 \wedge \\ & \exists x_{ul}^v, st : u \neq m \wedge v \neq k \wedge x_{ul}^v = 1 \\ \\ 0 & \text{otherwise.} \end{cases}$$

- w_{ml}^k determines whether different messages of other modes are allocated to the same slot l if a message i of mode k is already allocated to slot l.
- δ_m^k is an input delay value for message m in mode k.
- $\Pi_m^k = l \in \{1, \ldots, \text{LCM}\{\pi_m^k\}\}$ is the set of periods of messages in mode k.

$$\begin{aligned}
\min \quad & \sum_{\forall \text{m} \in \text{N}, l \in \Pi_m^k, \text{k} \in \text{V}} s_{ml}^k + \sum_{\forall \text{m} \in \text{N}, l \in \Pi_m^k, \text{k} \in \text{V}} x_{ml}^k * l. \\
st. \quad & \mathbf{C}_1\{\forall \text{m} \in \text{N}, \text{k} \in \text{V}\} : \\
& \sum_{l = g\pi_m^k + 1}^{g\pi_m^k + \pi_m^k} x_{ml}^k \geq c_m^k + \delta_m^k \ , g = 0, \ldots \alpha_m - 1; \\
& \mathbf{C}_2\{\forall l \in \Pi_m^k, k \in \text{V}\} : \\
& \sum x_{ml}^k \leq 1, \\
& \mathbf{C}_3\{\forall \text{m} \in \text{N}, l \in \Pi_m^k, \\
& \text{k} \in \text{V}, \text{u} \in \text{N} - \{m\}, \text{v} \in \text{V} - \{k\}\} : \\
& w_{ml}^k \geq x_{ul}^v, \\
& \mathbf{C}_4\{\forall \text{m} \in \text{N}, l \in \Pi_m^k, \text{k} \in \text{V}\} : \\
& s_{ml}^k \geq x_{ml}^k + w_{ml}^k - 1, \\
& \mathbf{C}_5\{\forall \text{m} \in \text{N}, l \in \Pi_m^k, \text{k} \in \text{V}\} : \\
& s_{ml}^k \geq 0,
\end{aligned}$$

Constraint C_1 specifies that all messages at least get the computation units in their periods even though a delay occurs.

Constraint C_2 specifies that no two messages are assigned to the same slot at the same mode. Constraint C_3 enforces overlaps. Constraint C_4 and C_5 enforces s^k_{ml} to be non-negative and binary. The objective function includes x^k_{ml} to assign overlaps as early as possible in the hyperperiod for faster mode-change ability in the generated state-based schedule.

B. Determining Valid Schedules

Due to computational delays the optimization model may unable to find a state-based communication schedule where messages can meet the timing constraints during the hyperperiod. If a valid schedule is absent for a given set of messages with timing constraints, the optimization model reports on infeasibility. Therefore, it is sufficient to use the optimization model to run for schedulability analysis of a set of messages in a number of communication modes up to the hyperperiod.

Definition 3 (Schedulability with delays)**.** *A given set of communication messages $\{m\}$ in a set of modes $\{k\}$ up to slot $l = 1\ldots, \mathrm{LCM}\{\pi^{\mathrm{k}}_{\mathrm{m}}\}$ will be schedulable for a given delay specification if the optimization model returns an assignment of x^k_{ml} for a given set of communication messages $\{m\}$ in all modes $\{k\}$ upto $l = 1\ldots, \mathrm{LCM}\{\pi^{\mathrm{k}}_{\mathrm{m}}\}$.*

C. Determining Delays

Interfaces provide specifications of maximum tolerable delays of components. Thus, if δ^* is the maximum delay and δ_r is a delay that components may experience, then δ_r will be no greater than δ^*. Since state-based communication schedules have the ability to make decisions at runtime, it is possible to switch between schedules at run time for different delays. Considering a set of delays as *dynamic* possible values of δ_r which are no greater than the maximum tolerable delay may provide a greater range of schedulability with an increase of number of generated schedules. In contrast, a system may use the maximum of tolerable delays of all components as the *static* value of δ_r to generate a state-based schedule. Using the static value of δ_r may reduce the schedulability of operational modes but provides guarantees that the schedule is valid for all dynamic values of δ_r.

1) Dynamic delays: To consider dynamic values of delays, we assume a set Δ_r of finite number of non-negative delays up to the maximum delay (Equation 5). Therefore, if δ_t is the delay that a component experiences at run time, then $\delta_t \leq \delta_r$, where $\delta_t \in \mathbb{R}_{\geq 0}$. Different δ_r values may be feasible to generate a state-based schedule for components running in a communication mode. Fig. 5 shows an example of different δ_r values for which a state-based communication schedule can be generated for component C_1 and C_2 in mode k. We can reduce the cardinality of Δ_r by choosing a local maximum of δ_r as shown in Fig. 5 to generate state-based schedules for a less number of δ_r values.

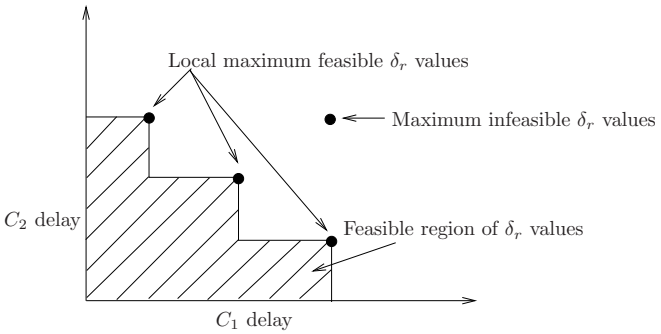

Fig. 5: Analysis on δ_r values in mode k

$$\Delta_r = \{\delta_r \mid (\delta_r < \delta^*) \cup (\delta_r = \delta^*), \delta_r \in \mathbb{N}_{\geq 0}\}. \quad (5)$$

Example 2. *Consider two components with a worst-case tolerable delay of 3 time units for the communication specification as shown in Table II, III. Under the assumption of dynamic delays, δ_r can be set to 0, 1, 2, and 3. State-based schedules are generated for the δ_r values that the optimizer can use to produce a feasible solution. Thus, if components are running in mode 7 and experience a delay $\delta_t \leq 2$, it can still operate in the same mode. However, if $2 < \delta_t \leq 3$, then the components can switch to other feasible modes such as mode 1 to continue operation using valid schedules.*

2) Static delay: Each component j may experience a delay δ^*_j in the worst-case, which not necessarily the maximum delay among all components in the system. Under the static delay assumption, the value of δ_r is the maximum of all delays that components can experience in the worst-case (Equation 6).

$$\delta_r = \max_j \delta^*_j. \quad (6)$$

In the case study, the worst-case delay that the components can suffer in each hyperperiod is 2 time units. Under the assumption of dynamic delays, the δ_r values are non-negative values up to 2 time units. In contrast, the δ_r value is 2 time units under the assumption of static delay.

D. Construction of State-based Schedules

Optimization model provides assignments of messages to slots for each of the communication mode if a state-based schedule can be generated. To construct the schedule that has messages timing requirements met within the hyperperiod, groups are formed using overlaps. However, to guarantee timing requirements upon a mode change, separate groups are formed for messages scheduled in the current slot but not in the previous slot. The optimization model manages to find optimal assignments of x^k_{ml} that the decomposition method (Definition 4) uses to construct a state-based schedule.

Definition 4 (Decomposition method from [2])**.** *Given all schedulable and reachable groups at any time slot l, a decomposition method derives all groups in the slot l+1 from x^k_{ml} followed by a labelled transition (i.e., guard).*

E. Characteristics of Generated Schedules

The following theorem describes that a control system component can have a worst-case tolerable delay. Using Lemma 2, it is shown that the communication delay is also bounded for control system components connected using interfaces and state-based schedules.

Theorem 1 (from [7])**.** *For a given scalar η and gain matrix K, if there exist matrices $P > 0$, $T > 0$, N_i and M_i ($i = 1, 2, 3$) of appropriate dimensions such that* (8) *is true, then the component is asymptotically stable with the state feedback input $u(t) = Kx(t_k)$, $t \in [t_k + \tau_k, t_{k+1} + \tau_{k+1})$, as long as the sampling period p and worst-case delay τ^* satisfy*

$$p + \tau^* \leq \eta. \tag{7}$$

Lemma 1. *The communication delay is bounded for a given a system with a number of components connected using interfaces and state-based schedules.*

Proof. The amount of delays that a component experiences is bounded, transient, and periodic in nature. Let us consider a component j with workload utilization $U^j_W = \sum_i \frac{e^j_i}{p^j_i}$, resource utilization $U^j_R = \frac{\theta^j}{\lambda^j}$, where task i executes e^j_i in every p^j_i and resource R_j provides θ_j units of time in every λ_j. As proved in [3], if $U^j_R = U^j_W$ then after $t = 2(\lambda_j - \theta_j)$, the function $f(t)$ that represents overloads using sbf(t) and dbf(t) is periodic with period LCM$(\lambda, p^j_1, \ldots, p^j_n)$, i.e.,

$$f(2(\lambda_j - \theta_j) + t + y\text{LCM}(\lambda, p^j_1, \ldots, p^j_n)) = f(2(\lambda_j - \theta_j) + t), \ \ \forall t \in \mathbb{R}_{\geq 0}, \forall y \in \mathbb{N}.$$

The delays that components can experience in the worst-case are within the delay tolerance, because this is guaranteed through calculating an efficient resource supply [3]. Therefore, the communication delay depends on the mode-change delay in the worst-case. The communication delay is bounded to the hyperperiod of all messages transmitted in the network because the schedule repeats after the hyperperiod. □

Theorem 2 (from [2])**.** *Given a set of valid state-based schedules for a particular system, the schedule with the fewer number of groups in a communication round will have lower average mode-change delay for a uniform probability of mode-changes. The average mode-change delay $\overline{\phi_{sd}}$ for all slots for all possible groups at runtime is:*

TABLE IV: Average mode-change delay analysis

Types of assignments	Delay [in ms]
Optimized	27.77
Randomized	33.13 (Best=30.62, Worst=35.03)
EDF-assigned	29.45

$$\overline{\phi_{sd}} = \frac{\sum_l \sum_z (\text{LCM}\{\pi_m\} - l)(|V - G^l_z|)|G^l_z|}{\text{LCM}\{\pi_m\}|\bigcup_l(\bigcup_z G^l_z)|}. \tag{9}$$

The messages are assigned to slots in a state-based schedule by maximizing the number of overlaps. Therefore, generated state-based schedules using component interfaces have low average mode-change delay with an ability to transmit best-effort traffic without violating the worst-case transmission guarantees.

F. Optimization Results

We use a well-known solver, AMPL/GUROBI to design our case study but can be extended to any similar model using our open source modeling, optimization, and off-line schedule generation tool [1]. For the case study in the paper, we consider the static delay for the maximum possible utilization. Two of the modes (ID 4 and 7) utilize the communication medium 100% when the worst-case delay is 2 time units.

To demonstrate that the optimizer assigns the slots to messages efficiently with respect to minimizing the number of groups in the generated schedule, we use random assignments of x^k_{ml} and a well-known scheduling policy (i.e., EDF) that also meet the timing requirements of messages in each state. For the video-streaming case study in the paper with the maximum delay of 2 time units, Table IV shows the difference in the average mode-change delay for the generated schedule using optimization constraints and the best-case of 10000 times generated randomized but valid schedules. The average mode-change delay in the generated optimized schedule is found significantly better than the randomized schedules. Moreover, to construct a state-based schedule, we also assign messages to slots based on deadlines as used in the EDF scheduling policy, which also results higher average mode-change delay than the optimized schedule.

VI. Related Work

Separation of concerns has been gaining increasing attention because of the safety requirements in many real-time systems [13]. A computational unit may have different tasks scheduled under multiple scheduling policies such as EDF and RM. Timing requirements of tasks scheduled under different scheduling policies are converted into a single requirement. In [11], the authors propose a compositional real-time scheduling framework for real-time systems which can be

$$\begin{bmatrix} N_1 + N_1^T - M_1A - A^TM_1^T & N_2^T - N_1 - A^TM_2^T - M_1BK & N_3^T - A^TM_3^T + M_1 + P & \eta N_1 \\ * & -N_2 - N_2^T - M_2BK - K^TB^TM_2^T & -N_3^T + M_2 - K^TB^TM_3^T & \eta N_2 \\ * & * & M_3 + M_3^T + \eta T & \eta N_3 \\ * & * & * & -\eta T \end{bmatrix} < 0. \quad (8)$$

used to establish global (system level) timing properties of a component from individual timing properties of tasks running on a resource. The authors present schedulability conditions for a periodic task model and propose a periodic resource model under EDF and RM scheduling. This periodic resource model can compute a single timing demand from multiple timing requirements using supply and demand bound functions. In [12], the authors propose a compositional analysis framework that uses real-time calculus and assume/guarantee (A/G) interfaces. Integrating subsystems into a system having optimal interfaces provides isolation in developing adaptive and reconfigurable systems.

Traditional real-time communication protocols allow limited control to the applications over the communication behaviour at runtime. For example, application developers have to assign message priorities statically on a CAN bus to ensure that the priorities are unique [10]. FlexRay [9] follows a TDMA approach, assigning an specific slot at the end of each round for dynamic and arbitrary traffic. However, stations must always wait for that specific slot to transmit dynamic messages, and the timing of the messages transmitted during that slot is not guaranteed. Using state-based communication schedules [8], stations can make decisions at runtime and timing of the messages transmitted during that slot can be guaranteed. This provides flexibility as well as predictability in message transmissions.

VII. Conclusion

In this paper, we present a workflow with an illustration of a real-time video monitoring case study to implement higher level component abstractions using interfaces and generate state-based schedules that facilitate conditional executions at run time. Component interfaces provide additional information in generating state-based schedules for communication such as delays that cause data to become available late for transmission. We use constraints to find an optimized state-based schedule that low average mode-change delay. Therefore, this work achieves not only safety through isolation or diversity but also performance due to efficient control on communication and low average mode-change delay.

References

[1] State-based schedule modeling, optimization, and generation. http://www.mathworks.com/matlabcentral/fileexchange/44716.

[2] A. Azim, G. Carvajal, R. Pellizzoni, and S. Fischmeister. Generation of Communication Schedules for Multi-Mode Distributed Real-Time Applications. In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2014.

[3] A. Azim, S. Sundaram, and S. Fischmeister. An Efficient Periodic Resource Supply Model for Workloads with Transient Overloads. In *Proc. of the Euromicro Conference on Real-Time Systems (ECRTS)*, 2013.

[4] G. C. Buttazzo. Rate Monotonic vs. EDF: Judgment Day. *Real-Time Systems*, 29(1):5–26, January 2005.

[5] S. Chakraborty, T. Mitra, A. Roychoudhury, L. Thiele, U.D. Bordoloi, and C. Derdiyok. Cache-aware timing analysis of streaming applications. In *Real-Time Systems, 2007. ECRTS '07. 19th Euromicro Conference on*, pages 159–168, July 2007.

[6] Y. Chen, T. Farley, and N. Ye. QoS Requirements of Network Applications on the Internet. *Inf. Knowl. Syst. Manag.*, 4(1):55–76, January 2004.

[7] Q.-L. Han D. Yue and C. Peng. State feedback controller design for networked control systems. *IEEE Transactions on Circuits and Systems – II: Express Briefs*, 51(11):640–644, Nov. 2004.

[8] S. Fischmeister, R. Trausmuth, and I. Lee. Hardware Acceleration for Conditional State-Based Communication Scheduling on Real-Time Ethernet. *IEEE Trans. on Industrial Informatics*, 5:3, 2009.

[9] FlexRay Consortium. *FlexRay Communications System – Protocol Specification*, June 2004. Version 2.0.

[10] Robert Bosch GmbH. *CAN Specification, Version 2*, September 1991.

[11] I. Shin and I. Lee. Compositional Real-Time Scheduling Framework. In *IEEE Real-Time Systems Symposium*, 2004.

[12] E. Wandeler and L. Thiele. Interface-Based Design of Real-Time Systems with Hierarchical Scheduling. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

[13] A. Wassyng, M. Lawford, and T. Maibaum. Separating Safety and Control Systems to Reduce Complexity. In *Conquering Complexity*, pages 85–102. Springer, 2012.