Hybrid Automotive In-Vehicle Networks

Special Session Paper

Debayan Roy
Technical University of Munich
debayan.roy@tum.de

Michael Balszun
Technical University of Munich
michael.balszun@tum.de

Dip Goswami
Eindhoven University of Technology
d.goswami@tue.nl

Samarjit Chakraborty
Technical University of Munich
samarjit@tum.de

ABSTRACT

The design of automotive in-vehicle networks is influenced by several factors like bandwidth, real-time properties, reliability and cost. This has led to a number of protocols and communication standards like CAN, MOST, FlexRay and more recently the use of Ethernet. In the future, wireless in-vehicle communication might also become a possibility. In all of these cases, often hybrid schemes such as the combination of time-triggered (TT) and event-triggered (ET) paradigms have been considered to be useful. Thus, hybrid protocols like FlexRay and TTEthernet, offering advantages of TT and ET communications, are becoming more popular. However, until now the hybrid nature of the protocols has not been exploited in application design. In this paper, we will discuss design strategies for automotive control applications that exploit the hybrid nature of the underlying communication architecture on which they are mapped. Towards this, we will consider a mix of time- and event-triggered schemes as well as a combination of reliable and unreliable communication. Correspondingly, we will show how appropriate abstractions of these hybrid schemes could be lifted to the application design stage.

CCS CONCEPTS

• **Computer systems organization** → **Embedded software**; *Real-time system architecture*; • **Networks** → *Time synchronization protocols*; *Sensor networks*;

ACM Reference format:
Debayan Roy, Michael Balszun, Dip Goswami, and Samarjit Chakraborty. 2017. Hybrid Automotive In-Vehicle Networks. In *Proceedings of NOCS '17, Seoul, Republic of Korea, October 19–20, 2017,* 8 pages.
DOI: 10.1145/3130218.3130235

1 INTRODUCTION

Electrical and electronic (E/E) systems of modern vehicles are becoming increasingly more complex and diverse. Such a system comprises number of processing units (PU), sensors and actuators connected over a communication network. Several applications are mapped onto these systems. These applications range from safety-critical control applications and advanced driver assistance systems (ADAS) to comfort-related applications and infotainment systems. In order to derive advanced and complex functionality, such applications are often implemented in a distributed fashion. Thus, they require data transmissions between electronic components over communication networks. The large spectrum of applications has led to diverse requirements from the underlying E/E architecture, in particular, the communication network architecture.

The design of automotive in-vehicle networks depends on the nature of data to be transmitted. For example, safety-critical control data must satisfy real-time properties and reliability requirements while infotainment and camera-based driver assistance data require high network bandwidth [1]. As a result, the E/E system is composed of several subnetworks connected via gateways. These subnetworks offer different network bandwidth and implements different communication protocol. Each subnetwork serves several applications belonging to a specific functional domain, e.g., FlexRay for chassis, high speed CAN for powertrain, low speed CAN or LIN for body and MOST or Ethernet for infotainment. Furthermore, in-vehicle wireless communication may become a possibility in the future [2] due to its low cost and less cabling overheads. Naturally, reliability is a very big issue with wireless communication.

CAN [3] has been the de facto standard for in-vehicular communications for a long time and offers priority-based ET communication. With increasing number of applications being mapped on a modern E/E system, the amount of data to be transmitted over a CAN bus has also increased manifold. As a result, it is challenging to satisfy the real-time requirements for all data in the worst-case, and correspondingly, safety might be compromised.

TT communication protocols, such as TTP/C, are becoming more popular for safety-critical control applications as they offer timing determinism. Typically, in TT communication, resource is periodically reserved for a message, and, in case no data is sent, the reserved resource is not used. However, this is resource inefficient and is not sustainable in cost-sensitive automotive domain.

The conflicting requirements on timing determinism and resource efficiency have led to the evolution of hybrid protocols. Examples include (i) FlexRay [4] which offers both time-division multiple access (TDMA) and flexible TDMA (FTDMA) communications and (ii) TTEthernet [5] which allows time-triggered, rate-constrained and best-effort traffic simultaneously. These protocols offer the advantages of both TT and ET communications.

This work was supported by (i) Deutsche Forschungsgemeinschaft (DFG) through the TUM International Graduate School of Science and Engineering (IGSSE) and (ii) I-MECH (GA no. 737453) project funded by ECSEL JU.

NOCS '17, Seoul, Republic of Korea

DOI: 10.1145/3130218.3130235

In practice, although FlexRay is employed in modern vehicles, the hybrid nature of the protocol is not exploited while designing the applications. In particular, safety-critical control applications use only TDMA communication in FlexRay. This is resource inefficient in the sense that a control loop is inherently robust and the control law need not be always executed deterministically at high frequency [6]. Therefore, ET communication can be naturally multiplexed with TT communication to improve resource-efficiency.

In this paper, we discuss two design strategies for control applications exploiting hybrid communication architecture. First, we consider that multiple control applications share a common hybrid bus offering both TT and ET communications. In this problem setting, we outline a bi-modal control strategy where each mode consists of a different controller and a bus schedule. Here, the mode switch request is triggered when the corresponding controlled plant is disturbed. Second, we describe another bi-modal control strategy which exploits best-effort communication to improve the control performance while satisfying the worst-case performance using TT communication. The switch between a fast and a slow controller is based on the online evaluation of the worst-case performance.

Towards discussing control strategies over hybrid automotive in-vehicle networks, the paper is structured as follows. In the next section, we provide background on linear feedback control systems and hybrid communication protocols. Subsequently, in Sec. 3, we explain a control/communication co-design strategy exploiting a mix of TT and ET communication. Then, we describe in Sec. 4 how to efficiently utilize elastic communication resource to enhance control performance. Next, we also outline in Sec. 5 some future works in the direction of control over hybrid networks. Finally, we conclude in Sec. 6.

2 PRELIMINARIES

2.1 Feedback Control Systems

In this paper, we consider linear time-invariant (LTI) single-input feedback control systems. The continuous-time dynamics for such a system can be represented mathematically as

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t). \tag{1}$$

For an n-th order system with m outputs, $x(t) \in \mathbb{R}^{n\times 1}$, $u(t) \in \mathbb{R}$ and $y(t) \in \mathbb{R}^{m\times 1}$ represent respectively the system states, the control input and the outputs at time t. For LTI systems, the matrices A, B and C (of appropriate dimensions) are constant.

In an embedded implementation, the sensors read the feedback signals and the actuator applies the control input at discrete instants of time. Therefore, the closed-loop system can be naturally represented by a sampled-data model, as shown in Figure 1. Here, the feedback signals are measured at time instants $\{t_k\}$ and are represented by $\{x[k]\}$, where, $x[k] = x(t_k)$. Traditionally, a controller is implemented according to a constant sampling period h, where, $t_{k+1} - t_k = h$. In addition, software execution and frame transmission result in non-negligible time delays. Particularly, for highly constrained processing units and contention-based networks, these delays can be sufficiently large. They contribute to sensor-to-actuator delay τ (as shown in Figure 1) and must be taken into account in the design.

In this paper, we will consider three different cases corresponding to different values of τ.

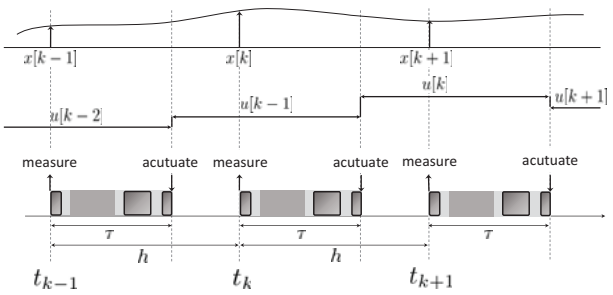

Figure 1: Delayed distrete-time system.

(i) $\tau \approx \mathbf{0}$: In this case, the discrete-time system can be modeled as

$$x[k+1] = \phi x[k] + \Gamma u[k], \quad y[k] = Cx[k]. \tag{2}$$

where ϕ and Γ (for given continuous-time model and sampling period) can be calculated as

$$\phi = e^{Ah}, \quad \Gamma = \int_0^h (e^{At}dt) \cdot B. \tag{3}$$

In this case, the control input $u[k]$ is calculated almost instantaneously based on the value of $x[k]$ and can be written as

$$u[k] = -Kx[k], \tag{4}$$

where K is the feedback gain. To stabilize the system, K can be calculated using Linear Quadratic Regulator (LQR) or pole-placement techniques [7]. We may note that a discrete-time closed-loop system is stable when all the closed-loop poles are within a unit circle.
(ii) $\mathbf{0} < \tau < \mathbf{h}$: The delayed sampled-data model [7], as shown in Figure 1, becomes

$$x[k+1] = \phi x[k] + \Gamma_0 u[k] + \Gamma_1 u[k-1], \quad y[k] = Cx[k], \tag{5}$$

where Γ_0 and Γ_1 (for a given delay τ) are given by

$$\Gamma_0 = \int_0^{h-\tau} (e^{At}dt) \cdot B, \quad \Gamma_1 = \int_{h-\tau}^{h} (e^{At}dt) \cdot B. \tag{6}$$

Considering an augmented state vector $z[k] = \begin{bmatrix} x[k] & u[k-1] \end{bmatrix}^T$, the system can be modeled as

$$z[k+1] = \phi_z z[k] + \Gamma_z u[k], \quad y[k] = C_z z[k], \tag{7}$$

where ϕ_z, Γ_z and C_z are given by

$$\phi_z = \begin{bmatrix} \phi & \Gamma_1 \\ \mathbf{0} & 0 \end{bmatrix}, \quad \Gamma_z = \begin{bmatrix} \Gamma_0 \\ \mathbf{I} \end{bmatrix}, \quad C_z = \begin{bmatrix} C & 0 \end{bmatrix}. \tag{8}$$

Now, for $u[k] = -Kz[k]$, we may use LQR or pole-placement techniques to design the feedback gain K.
(iii) $\tau = \mathbf{h}$: In this case, we consider the controller model from [8] where the control input for the k-th instant is calculated based on the state at the $(k - \lfloor \frac{\tau}{h} \rfloor)$-th instant. For $\tau = h$, the control input $u[k]$ can be written as

$$u[k] = -Kx[k-1]. \tag{9}$$

Based on the above control law in Eq. (9) and the system model in Eq. (2), the control gain K can be calculated using a restricted pole-placement technique described in [8].

Besides stability, the design of a controller often needs to consider performance and physical constraints. Common performance

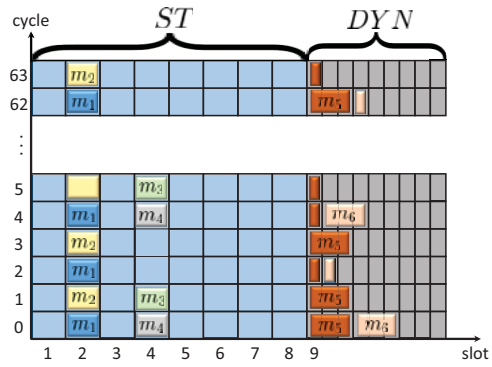

Figure 2: Example of FlexRay communication.

metrics are settling time, steady-state error, overshoot, and so on. Typical physical constraints include input saturation, actuator bandwidth, etc. To design an optimal controller with pole-placement, we need to formulate an optimization problem with closed-loop poles as decision variables satisfying stability condition and physical constraints. Here, the optimization objective is the closed-loop performance. The resulting problem is non-convex in nature, and, we use Particle Swarm Optimization (PSO) technique [9, 10] to solve the problem due to its polynomial-time complexity.

2.2 Hybrid Communication Protocols

As already explained, TT communication offers timing determinism while ET communication is resource-efficient. Hybrid protocols like FlexRay and TTEthernet bring them together as described in the following text.

FlexRay: Each FlexRay time cycle is mainly partitioned into static segment(ST) and dynamic segment (DYN). ST exhibits TDMA communication and comprises number of *slots* of equal length (Ψ). A message assigned to a ST slot is transmitted within the corresponding time window. Thus, the start and end of a message transmission is precisely known. On the other hand, DYN is partitioned into number of *minislots* of equal length (ψ), where typically $\psi << \Psi$. A message assigned to the DYN may consume more than one minislot as shown in Figure 2. A DYN slot is a logical entity with one or more minislots allowing flexible TDMA communication.

This hybrid communication is realized using a slot counter, $\mathcal{SC}$, where the counter starts from 1 at the beginning of each cycle. When $\mathcal{SC} = j$, the message m_i assigned to the j-th slot is sent over the bus (if ready). In ST, the counter is incremented after every Ψ time units, i.e., at the end of each slot. If no data arrives at the beginning of the slot, entire slot of length Ψ goes unused. An example is shown in Figure 2 for m_2 in cycle 5. In DYN, the counter is incremented at the end of the last minislot of transmission for a message m_i. Counter is updated at the end of the current minislot when data is not ready for transmission (e.g., m_4 and m_5 in Figure 2) and only one minislot of transmission time is wasted. This results in time-varying transmission delay of the messages in DYN while the worst-case can still be deterministic.

TTEthernet: It is an Ethernet-based solution for automotive systems. It allows for TT, rate-constrained (RC) and best-effort (BE) traffic. (i) TT communication is realized based on time-synchronized static schedules which are non-overlapping. TT messages have the highest priority, and therefore they are sent exactly when scheduled. (ii) RC messages have a certain upper bound on the inter-arrival rate. They are sent only when there are no TT messages. RC messages can be assigned priorities. Messages of same priority are served according to first-in-first-out (FIFO) scheme. Considering the scheduled TT traffic, the assigned priority and the bounded inter-arrival rate, the worst-case delay of an RC message may be determined [11]. Thus, RC traffic is suited for real-time applications with softer latency and jitter requirements. (iii) BE traffic is realized according to standard Ethernet protocol (CSMA/CD) and has no timing guarantees.

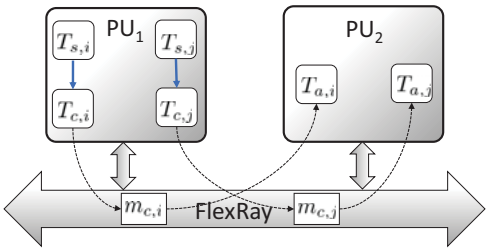

Figure 3: Problem setting.

3 HYBRID PROTOCOL AWARE CONTROL/ COMMUNICATION CO-DESIGN

In this section, we consider a problem setting where multiple control applications communicate over a *shared* FlexRay bus. As discussed in Sec. 2.2, FlexRay ST offers TDMA-based TT communication while DYN can be used for ET communication. Given this communication architecture, we will discuss further in this section how to design a bi-modal control strategy exploiting the hybrid characteristic. In particular, the idea is that a control application will use ET transmission when in steady state and requests for TT slots in the event of disturbance until the system settles down. In this context, we first formally describe the problem and then explain the bi-modal control strategy and the corresponding communication scheduling strategy. Here, we summarize the works [12–14].

3.1 Problem Description

Consider a set of control applications $C = \{C_1, C_2, ..., C_N\}$ sharing the FlexRay bus. The sampling period of the controller C_i is given and is denoted by h_i. Each application C_i is partitioned into three processing tasks, i.e., $T_{s,i}$ (measures $x[k]$), $T_{c,i}$ (computes $u[k]$) and $T_{a,i}$ (applies $u[k]$ to the plant). We assume that each task is triggered periodically according to a processor schedule. The period of task executions in an application C_i is equal to the sampling period h_i. We assume that the tasks $T_{s,i}$ and $T_{c,i}$ are mapped on to the same PU which is attached to the corresponding sensors. The task $T_{a,i}$ is mapped on a different PU connected to the actuator. This setting is common where sensors and actuators are spatially distributed. $T_{a,i}$ receives $u[k]$ computed by $T_{c,i}$ as a message $m_{c,i}$ over the communication bus. Thus, $m_{c,i}$ can be sent as a TT or an ET message. The problem setting is illustrated in Figure 3.

We consider that external signal can perturb the behavior of a plant which we refer to as *disturbance*. It causes the system to go

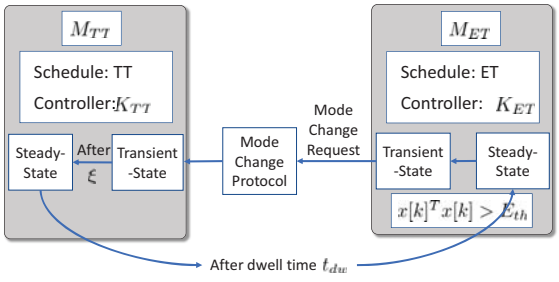

Figure 4: Bi-modal control strategy over a hybrid protocol.

into a transient state. The performance of real-time control systems is often quantified by *settling time* – the time taken by a closed-loop system after a disturbance to reach and remain with a certain error band of the reference signal. In our setting, each control application must satisfy a performance requirement in terms of settling time, i.e., $\xi_{d,i}$. The design objective is to determine the control and communication strategies for all control applications such that they meet their respective settling time requirements even in the worst-case.

Furthermore, we consider that TT slots are limited and a part of them are reserved for system applications. This makes the TT slots expensive and limited in availability. Traditionally, each control application gets one dedicated TT slot for predictable and low-latency communication. However, it is desirable to use as less TT slots as possible. Thus, we aim to reduce the usage of TT slots without compromising much on the control performance.

3.2 Bi-Modal Control Strategy

In the setting described above, for an application C_i two control modes are possible, i.e., $M_{TT,i}$ and $M_{ET,i}$. In $M_{TT,i}$, control message $m_{c,i}$ is sent in FlexRay ST and the timings are precisely known. The application tasks can be scheduled accordingly so as to achieve negligible sensor-to-actuator delay, i.e., $\tau = 0$. In $M_{ET,i}$, $m_{c,i}$ is sent as an ET message in FlexRay DYN. Here, $m_{c,i}$ may have to wait until all the messages mapped on the preceding slot ids (with higher priorities) are transmitted. We assume that the worst-case response time (WCRT) of $m_{c,i}$ leads to the case of one sampling period delay, i.e., $\tau = h$. Corresponding to the modes $M_{TT,i}$ and $M_{ET,i}$, we can design stable and optimal controllers $K_{TT,i}$ and $K_{ET,i}$ respectively according to the method explained in Sec. 2.1.

The control in mode $M_{ET,i}$ is slower due to a higher delay (of one sampling period) compared to the mode $M_{TT,i}$. Thus, performance requirement may not be met if we use only $M_{ET,i}$. Performance is much better (with a shorter settling time) than the requirement when each application has a dedicated TT slot and $M_{TT,i}$ is used always. However, this is expensive in terms of communication. Thus, the idea is to use a bi-modal control strategy for each application, as illustrated in Figure 4, which switches between the two modes ($M_{TT,i}$ and $M_{ET,i}$). In the process, each application only partially uses TT slots. Thus, it is possible to share a TT slot among multiple applications. This potentially reduces the usage of the TT slots saving the communication cost. Here, two important questions are: (i) When to trigger a mode switch request? (ii) When an application get access to a TT slot? In this section, we will only discuss the first question. We will consider the second question in Sec. 3.3.

Based on the reasons explained above, the switching strategy can be devised as shown in Figure 4. In this strategy, a controller stays in $M_{ET,i}$ when the system is in steady state while requests a switch to $M_{TT,i}$ when in transient state. The mode switch request is triggered (or the transient state is detected) when the system energy is greater than a threshold, i.e., $x[k]^T x[k] > E_{th,i}$, where $E_{th,i}$ is the threshold value. In the steady state, the control input and the plant states do not change appreciably and thus the higher value of delay does not deteriorate the control performance. However, in the transient state, the smaller value of delay enables the plant to reach the reference signal quickly. It is to be noted here that after the mode switch request the controller may have to wait a certain time to switch to $M_{TT,i}$. This time depends on the scheduling strategy of the TT slots that we will discuss next in Sec. 3.3.

Now, the question is when should the controller switch back to $M_{ET,i}$? A problem that crops up in such a switching control strategy is to guarantee switching stability. Towards this, we consider a dwell time $t_{dw,i}$ for which a controller must be in $M_{TT,i}$ before switching to $M_{ET,i}$. Let us denote $\xi_{TT,i}$ as the settling time of the system when the mode $M_{TT,i}$ is used to stabilize the system after maximum disturbance. $t_{dw,i}$ can be pessimistically selected such than $t_{dw,i} > \xi_{TT,i}$. Therefore, $t_{dw,i}$ allows the system to stay in mode $M_{TT,i}$ for sufficiently long to reach a steady state, i.e., $x[k]^T x[k] < E_{th,i}$ [15]. This is to ensure that switching does not bring the system to transient-state again.

3.3 Communication Scheduling Strategy

We apply the aforementioned control strategy to every control application in C. Let us denote $\xi_{ET,i}$ as the settling time of the system when the mode $M_{ET,i}$ is used to stabilize the system after maximum disturbance. Now, if $\xi_{TT,i} < \xi_{d,i} < \xi_{ET,i}$, then the scheduler must allocate TT slot to the controller C_i before the deadline expires to ensure the worst-case requirement. The conservative method would be to allocate individual TT slot to each application which requires TT service to satisfy the performance requirement. However, this is pessimistic. Can we do better than this?

This brings us to an important question: How many minimum TT slots are required to satisfy the performance constraints of all the applications? The computation of number of minimum slots consists of two interlocked steps, i.e., (i) the schedulability analysis of one TT slot shared by multiple control applications, and, (ii) the provisioning of TT slots for applications.

3.3.1 Schedulability analysis. In this step, we consider that a number of control applications are contending for a single TT slot. A fixed priority access of the slot can be considered where the priorities are statically determined according to the settling time requirements $\{\xi_{d,i}\}$. Shorter the value of $\xi_{d,i}$ is, higher is the priority that the slot will be accessed by C_i. Furthermore, we consider a certain minimum inter-arrival time r_i of disturbances for an application C_i. We further assume that $\xi_{d,i} \leq r_i$ which implies that an application must reject a disturbance before the next one arrives. The worst-case occurs when the disturbances arrive for all the applications at the same time. Here, we discuss two different slot access policies, i.e., preemptive and non-preemptive.

Fixed priority non-preemptive slot access: We have already stated that (i) an application may have to wait a certain time $t_{w,i}$

before getting access to the TT slot, and then, (ii) the application must have access to the TT slot continuously for a certain dwell time $t_{dw,i}$. Now, during $t_{w,i}$ the controller in mode $M_{ET,i}$ rejects a certain part of the disturbance, and correspondingly, we can determine the $t_{dw,i}$ as

$$t_{dw,i} = \xi_{TT,i} - \beta_i t_{w,i}, \tag{10}$$

where β_i can be approximated by $\beta_i = \frac{\xi_{TT,i}}{\xi_{ET,i}}$. The actual settling time can therefore be written as

$$\begin{aligned} \xi_i &= t_{w,i} + t_{dw,i}, \\ &= \xi_{TT,i} + (1-\beta_i) t_{w,i}. \end{aligned} \tag{11}$$

We may note that ξ_i increases with increase in $t_{w,i}$ and there must be an upper bound of $t_{w,i}$ to meet the requirement $\xi_{d,i}$.

Hence, to test the schedulability of C_i, we need to find the greatest possible $t_{w,i}$ (denoted by $\hat{t}_{w,i}$) which leads to the worst-case settling time of C_i (denoted by $\hat{\xi}_i$). This occurs when C_i suffers the maximum possible interference due to higher priority applications. For this, we will consider that all higher priority applications C_j interfering with C_i require their maximum possible transmission time on the shared slot, i.e., $t_{dw,j} = \xi_{TT,j}$. This assumption is pessimistic since $t_{dw,j}$ actually decreases with the blocking time suffered by C_j. However, this allows us to simplify the analysis and leads to a safe schedulability condition. Now, the worst-case interference on C_i clearly occurs when it needs to have access to the TT slot together with all higher priority C_j (sharing the same slot). This again happens when all higher priority C_j and C_i undergo disturbances at the same time. In addition, since the scheduling is non-preemptive, C_i may also suffer some blocking time due to lower-priority applications.

Computing $\hat{t}_{w,i}$ and $\hat{\xi}_i$ here has some similarities with computing the worst-case response time in a fixed-priority non-preemptive scheduling like the one of CAN. That is, we need to compute the response times of all instances of disturbance rejection within its maximum busy period $t_{max,i}$ [16]. For ease of exposition, we assume that $t_{max,i} \leq r_i$ holds for all C_i, i.e., there is only one transmission of $\frac{t_{dw,i}}{h_i}$ consecutive messages of C_i within its busy period $t_{max,i}$. This way, we only need to compute the response time ξ_i of the sole instance of C_i within $t_{max,i}$ to obtain its worst-case response time $\hat{\xi}_i$, which can be done solving the following recurrence relation:

$$\xi_i(k+1) = \xi_{TT,i} + (1-\beta_i) b_i + (1-\beta_i) \sum_{j=1}^{i-1} \left\lceil \frac{\xi_i(k)}{r_j} \right\rceil \xi_{TT,j}, \tag{12}$$

where $b_i = \max_{z=i+1}^{N_s}(\xi_{TT,z})$ denotes the maximum possible blocking time due to lower-priority applications suffered by C_i and N_s is the number of applications mapped onto the slot s. Without loss of generality, we assume in Eq. (12) and in the remainder of this section that applications are sorted in order of decreasing priority (i.e., C_j has higher priority than C_i and C_i has higher priority than C_k for $1 \leq j < i < k$). Eq. (12) can be solved recursively until ξ_i becomes greater than $\xi_{d,i}$ or converges to a certain value. Clearly, if ξ_i exceeds $\xi_{d,i}$, C_i is not schedulable on the shared slot. On the other hand, if there is a convergence value prior to $\xi_{d,i}$, then C_i can meet its deadline and is schedulable.

Fixed-priority preemptive slot access: In the aforementioned non-preemptive access scheme, a higher priority application with closer settling time deadline may be blocked by a lower priority application with a high dwell time. This may result in deadline miss. However, a higher priority application may meet its deadline if it is allowed to preempt the lower priority application. In this context, we discuss here a scheduling policy with limited preemption. In this scheme, a higher priority application C_j is only allowed to preempt a lower-priority application C_i after a configurable blocking time b'_j. In this scheduling strategy, we consider that the dwell time of an application $t_{dw,i}$ is constant and given. Now, based on the assumption that $t_{max,i} \leq r_i$, maximum allowable blocking time can be calculated as

$$\hat{b}_i = \xi_{d,i} - t_{dw,i} - \sum_{j=1}^{i-1} \left\lceil \frac{\xi_{d,i}}{r_j} \right\rceil t_{dw,j}. \tag{13}$$

Correspondingly, b'_i must be configured such that $b'_i \leq \hat{b}_i$. Now, this reduction in blocking time for higher priority applications comes at the cost of retransmission of lower priority applications.

In case C_j preempts C_i, C_i may have to retransmit for b'_j time. Note that if $b'_j \geq t_{dw,i}$ holds, C_i has enough time to finish sending its message sequence before b'_j expires and it will not incur in retransmission. Further, to obtain the maximum retransmission cost $t_{r,i}$ for a C_i, we need to consider the fact that the transmission of C_i can be canceled by any higher-priority C_j:

$$t_{r,i} = \max_{1 \leq j < i, b'_j < t_{dw,i}} b'_j. \tag{14}$$

Now, for a lower-priority C_i, the retransmission cost of its higher-priority C_j needs to be considered as blocking time for C_i. In worst case, all higher priority tasks of C_i may need retransmission. As a result, we can configure any positive blocking time for C_i that is at most equal to

$$b'_i = \hat{b}_i - t_{r,i} - \sum_{j=1}^{i-1} t_{r,j}. \tag{15}$$

Clearly, if no positive b'_i can be found, C_i cannot be scheduled. Furthermore, the schedulability on one slot can be guaranteed if a positive b'_i can be found for every C_i allocated to the slot. To this end, we need to compute b'_i in order of decreasing priorities from the highest to the lowest priority.

3.3.2 Slot Allocation. The problem of finding the minimum number of slots that guarantees the settling time requirements of all C_i is clearly an allocation problem. Often such problems are NP-hard in the strong sense. Here, we will discuss the well-known First Fit (FF) heuristic approach. FF leads to a number of slots that is acceptably close to the optimum and has polynomial complexity. This approach first sorts the applications $\{C_i\}$ according to increasing priority. Then, it iterates over the sorted set and tries to allocate them in the minimum possible number of slots.

The algorithm starts with only one slot and allocates the control applications to it as long as they are schedulable on that slot. A C_i is schedulable on a slot if it can meet its timing requirement $\xi_{d,i}$ when assigned to that slot. To test this, the algorithm makes use of the schedulability analysis presented in Sec. 3.3.1. In the current iteration, the algorithm first tries to allocate C_i to a slot in the list of existing slots. If C_i could not be scheduled on any of the exiting slots, it then adds a slot to the list. The algorithm concludes when

all C_i have been allocated and returns the number of slots that were necessary for accommodating all of them.

3.4 Experimental Results

	Case 1				Case 2		
C_i	r_i	$\xi_{d,i}$	$\xi_{TT,i}$	$\xi_{ET,i}$	r_i	$\xi_{d,i}$	$t_{dw,i}$
C_1	2000	150	50	200	2000	300	100
C_2	2000	500	200	550	2000	400	120
C_3	1500	150	50	200	1500	450	150
C_4	2000	300	200	400	2000	1000	300
C_5	5000	1000	800	2000	5000	3000	800
C_6	600	600	300	700	500	500	50

Table 1: Case study (All times are in *ms*).

Given the 6 applications with details in Table 1–Case 1, we apply the non-preemptive slot access scheme. We calculated that the minimum number of slots required is 4 with the partition: $\{C_1, C_3\}$, $\{C_2, C_4\}$, $\{C_5\}$ and $\{C_6\}$. Then, we apply the preemptive slot access scheme for data in Table 1–Case 2. Here, we obtained 2 slots with the partition: $\{C_1, C_2, C_3, C_4, C_5\}$ and $\{C_6\}$. Both the schemes result in the allocation of less number of TT slots than there are applications. However, note that the two results are not comparable as they are based on different data sets.

4 EFFICIENT CONTROL OVER ELASTIC COMMUNICATION RESOURCES

In the last section, we have discussed a bi-modal control strategy where the closed-loop dynamics are exactly known for both the modes individually. In this section, we will consider a case where closed-loop dynamics are not always known. In particular, we consider that a control loop can be closed elastically over deterministic and non-deterministic communication. Here, we assume that the non-deterministic communication cannot provide any timing guarantees not even the worst-case response time. For such a setting, we study a bi-modal control strategy proposed in [17] which guarantees the worst-case performance requirement using deterministic communication. At the same time, it tries to improve the control performance by exploiting the available non-deterministic communication resource. In this section, we will first introduce the elastic network setting and the bi-modal controller for such a setting. Next, we will formally state the control goals and explain the mode selection algorithm that satisfies the goals. We will also provide some experimental results.

4.1 Network Setting

It is often the case that faster sampling improves the control performance. Based on this assumption, control over a hybrid network setting is considered as shown in Figure 5. The network consists of deterministic and non-deterministic communication channel. This can be represented by the combination of TT and BE traffic in a TTEthernet network as described in Sec. 2.2.

We consider that the state of the controlled plant is sensed at a faster rate f^F and the corresponding instants are denoted by $\{t_k^F\}$. The sensed data is then transmitted over the hybrid network. In this network, the deterministic channel is expensive. Thus, we use this channel at a very low rate, f^S, corresponding to which the sampling time instances are given by $\{t_k^S\}$ where $\{t_k^S\} \subset \{t_k^F\}$.

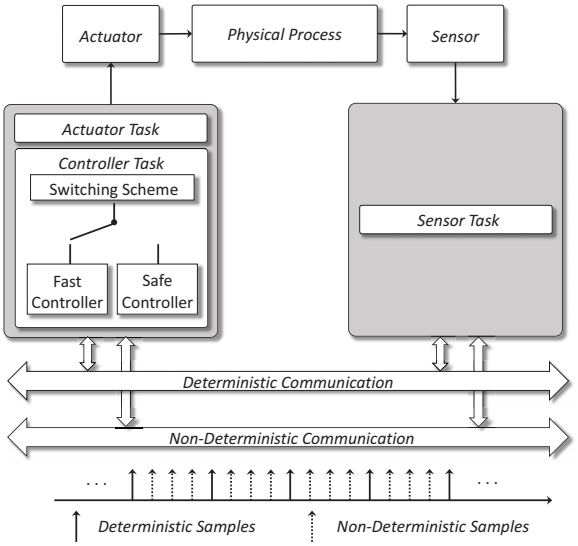

Figure 5: Control strategy over elastic resource.

Intuitively, f^S depends on the worst-case performance requirement of the control loop. Thus, communication over this channel enable the control scheme to guarantee the worst-case performance.

On the other hand, the non-deterministic channel is cheap. Therefore, this channel is used for sampling instances when deterministic channel is not allocated, i.e., $\{t_k^F\} \setminus \{t_k^S\}$. It may be noted that this channel may have unbounded transmission delay or packet drops. However, in most cases the delay is not too high and packet drops do not occur, which is the case with protocols like CSMA/CD, etc. We further assume that if a packet is not transmitted within a certain time then it is dropped. This can be realized, for example, using a CSMA with persistence within a given timeout or time-stamping. Now, considering this network setting, we may say that the control loop is only closed at instants $\{t_k^M\}$ where $\{t_k^S\} \subset \{t_k^M\} \subset \{t_k^F\}$. Here, control over non-deterministic channel can be exploited to improve the control performance by appropriately coming up with a switching control strategy $\mathcal{M}$.

4.2 Bi-modal controller

Due to nondeterminism in the control loop it is difficult to derive the closed-loop mathematical model of the overall system. Consequently, for the described network setting, we intuitively use a control strategy based on two control modes. We name the two controllers appropriately as slow controller $\mathcal{S}$ and fast controller $\mathcal{F}$ for which the control gains are denoted as K^S and K^F respectively.

K^S can be determined using the optimal control design algorithm explained in Sec. 2.1 according to the sampling period $h^S = \frac{1}{f^S}$. Since h^S is large, we expect that this controller makes the system converge to the equilibrium state rather slowly. Moreover, it has been observed from simulations using this controller at higher rate will not improve the rate of convergence appreciably. This is due to the inherent slow action of the controller.

On the other hand, we design the fast controller according to the sampling period $h^F = \frac{1}{f^F}$. This controller is expected to provide a much higher rate of system convergence than the slow controller at higher sampling rate. However, in case of packet drops, performance may degrade from the expected value to a point when requirement is violated.

Therefore, in order to achieve the overall system performance close to the performance of the fast controller in case of less constrained non-deterministic channel while limiting the maximum performance degradation within a threshold of the performance of slow controller, we need to devise an efficient control strategy $\mathcal{M}$.

4.3 Control Goals

We consider the problem where disturbance arrive sparsely. By sparse, we mean that a new disturbance only arrives after the previous one was successfully rejected. A disturbance means that the augmented system state z is moved from an equilibrium region $Z_{\mathcal{E}}$ around the reference point to a disturbed state z_0. This can either be due to an external influence or because the set-point was changed. Now, the goal is to drive the system back to steady state within a time ξ_d. Thus, the settling time ξ must satisfy $\xi \leq \xi_d$.

Here, we need to develop a control strategy $\mathcal{M}$ such that for any valid realization $\{t_k^M\}$ and initial disturbed state z_0, the following inequality holds true:

$$\xi^M(z_0) \leq \xi^S(z_0) \cdot \Delta \tag{16}$$

where (i) ξ^M is the settling time of the mixed control strategy $\mathcal{M}$, (ii) ξ^S is the settling time when only slow controller $\mathcal{S}$ is used at instants $\{t_k^S\}$, and (iii) $\Delta > 1$ represents a design parameter which limits the performance degradation of $\mathcal{M}$ in the worst-case. In addition, the objective for the design of $\mathcal{M}$ is to minimize the settling time $\xi^M(z_0)$ for the case $\{t_k^M\} = \{t_k^F\}$ (i.e. the channel is available at all non-deterministic time instances as well).

4.4 Mode selection algorithm

Given the network setting, the two control modes and the control goals, the mixed control strategy $\mathcal{M}$ and the corresponding mode selection can be realized follows. The mode selection algorithm is invoked at every time instant. The algorithm first checks if the system states are in the equilibrium region $Z_{\mathcal{E}}$. If the states are inside $Z_{\mathcal{E}}$, the slow control $\mathcal{S}$ is picked and the deadline is set to a negative value to indicate that the system is not in a transient state. Note that $\mathcal{S}$ only applies control input at time instances $\{t_k^S\}$ and is also designed accordingly. Thus, the closed-loop system is stable under steady state. Now, in case a new disturbance is detected, then $\xi^S(z_0)$ is calculated. A deadline is set corresponding to the control goal given by Eq. (16).

Now, when a transient state is detected, the algorithm tries to determine whether applying fast control $\mathcal{F}$ is possible without running the risk to arrive in a state from which Eq. (16) can no longer be satisfied. Here, the algorithm first computes the control input u_F corresponding to K^F. Then, it predicts the system states z_n at the next deterministic sample t_n^S provided u_F is applied at this instant. Correspondingly, it calculates the settling time $\xi^S(z_n)$ if $\mathcal{S}$ takes over at z_n. Now, if the calculated time $\xi^S(z_n)$ still satisfies the deadline then u_F is applied else $\mathcal{S}$ is selected.

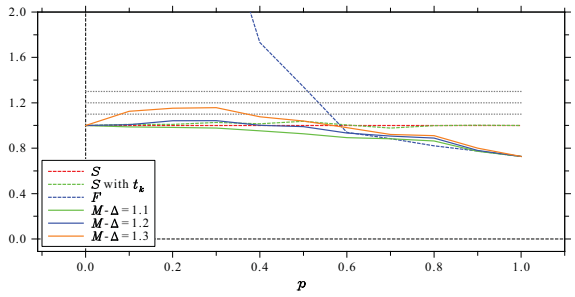

Figure 6: Settling times for different control strategies

The success of this scheme lies in the fact that at every time instance $\{t_k^M\}$ the algorithm evaluates whether the fall back scheme $\mathcal{S}$ can still satisfy the deadline. Thus, the algorithm satisfies the worst-case performance. Moreover, it is clear that when there are no packet drops, $\mathcal{F}$ is used mostly which results in fast control and lower settling time.

Settling time computation: The computationally heavy part of this algorithm is the calculation of $\xi^S(z)$. We are not aware of a closed-form solution to compute the settling time for a discrete-time system. Instead, one has to simulate the system evolution for a number of steps to determine the settling time. Depending on the dimensionality of the problem and the complexity of the control algorithm, this computation might be infeasible during runtime due to limited computational resources on the embedded platforms. An alternative approach would be to rasterize the state-space, pre-compute the worst-case settling times for each slice off-line and store them in a lookup table. This approach is computationally cheap but might have a high memory requirement. For simple state-feedback controllers, where the control law only involves a low-dimensional matrix multiplication, we expect however that it is more efficient to compute the settling time during runtime.

4.5 Experimental Results

To validate our expectations, we have considered a DC Motor speed control system. As shown in Figure 6, we have implemented different speed control strategies considering the elastic network setting. These strategies are (i) slow controller only at deterministic samples ($\mathcal{S}$), (ii) slow controller at both deterministic and non-deterministic samples ($\mathcal{S}$ with t_k^M), (iii) fast controller at both deterministic and non-deterministic samples ($\mathcal{F}$), and (iv) switching control strategy with fixed values of Δ ($\mathcal{M} - \Delta = X$). We have further simulated random packet drops over non-deterministic channel according to a successful transmission rate p ranging from 0 to 1. We have performed several simulations at each value of p for each strategy. The average normalized settling times (y-axis) for different control strategies at different values of p (x-axis) are illustrated in Figure 6.

Important observations (from Figure 6) are stated as follows: (i) $\mathcal{S}$ applied at all successful samples $\{t_k^M\}$ does not improve the performance appreciably as compared to $\mathcal{S}$ at $\{t_k^S\}$ only even at $p = 1$. (ii) $\mathcal{F}$ deteriorates the performance considerably at lower values of p. (iii) At higher values of p, performance of $\mathcal{M}$ matches that of F, and therefore, the objective to improve the performance with non-deterministic channel is achieved. (iv) At no point the performance

requirement is violated (we have observed this even for the worst-case). Thus, we may conclude that the studied bi-modal control strategy satisfies the worst-case performance and also improves the performance by exploiting the cheap communication resource.

5 FUTURE OUTLOOK

Although hybrid protocols have gained importance and are found in communication architectures of modern cars, applications are still not designed in practice exploiting these protocols. Moreover, there have been only a few preliminary research works in this direction. However, we believe that this is an important research direction towards resource-efficient design of automotive systems. In this context, we propose two possible future extensions to the control strategies that we have studied in this paper.

Switching stability: In Sec. 3, we have stated the problem of ensuring stability during mode change. As a solution, we have considered a minimum dwell time to ensure stability when changing the mode from TT to ET. Here, the dwell time may be pessimistic and results in blocking the TT slot unnecessarily. Moreover, we assume that the switching from ET to TT will not jeopardize stability.

In this context, we can apply the well-developed notion of switching stability from control theory. The theorem says that any arbitrary switching between two stable closed-loop systems is also stable if there exists a common quadratic Lyapunov function for the systems [18]. Now, we can consider this condition as a constraint for designing the gains for M_{TT} and M_{ET}. This will allow to switch back from M_{TT} to M_{ET} before the system returns to steady state. This can be exploited to devise a more efficient mode switch protocol and can reduce the number of TT slots required.

Wireless communication: With increasingly more number of applications mapped onto the E/E architecture of a modern car, the wiring complexity and volume of the underlying communication system has also grown. To address this problem, automotive community foresee the introduction of wireless networks for intra-vehicular communication [2].

Reliability is a key issue associated with many standard wireless communication protocols like IEEE 802.11x. However, we may note that wireless channel can naturally replace best-effort communication over TTEthernet in the control strategy discussed in Sec. 4. Thus, we propose to apply the bi-modal control strategy to a combination of wired and wireless communication. This reduces the in-vehicle wiring by minimizing the requirement on wired communication. This can also reduce the cost of using expensive communication infrastructure supporting hybrid protocols.

Furthermore, wireless protocol IEEE 802.15.4 is becoming popular for its low-cost and low-power solution. It is also possible to tune the parameters of this protocol to meet real-time requirements [19]. In the future, one can study this protocol and customize the general control strategies discussed in this paper accordingly.

6 CONCLUDING REMARKS

Hybrid protocols have gained a lot of importance in recent years towards supporting mixed-criticality applications. In this paper, we try to bring into attention the importance of hybrid protocols in the context of safety-critical control applications. In particular, we have pointed out that control applications can be designed in a resource-efficient way exploiting the hybrid characteristic of the underlying communication architecture. We have studied here two control strategies: (i) The first strategy uses ET communication for steady-state control while exploits TT communication only to stabilize the system faster in the event of disturbance. (ii) The second strategy uses deterministic communication only to guarantee the worst-case performance while exploits best-effort communication to improve the performance. Both the strategies reduce the use of expensive TT resources significantly while not compromising the performance. Furthermore, we believe that control over hybrid communication is not fully explored, and therefore, this is a promising research direction for the future.

REFERENCES

[1] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin. Intra-Vehicle Networks: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):534–545, 2015.

[2] M. Laifenfeld and T. Philosof. In-vehicle hybrid electrical architecture. In *IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, pages 1–5, 2014.

[3] ISO 11898. Road vehicles – Controller area network (CAN). URL https://www.iso.org/standard/63648.html.

[4] ISO 17458. Road vehicles – FlexRay communications system. URL https://www.iso.org/standard/59804.html.

[5] AS6802. Time-Triggered Ethernet. URL http://standards.sae.org/as6802/.

[6] P. Martí, J. M. Fuertes, G. Fohler, and K. Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *23rd IEEE Real-Time Systems Symposium (RTSS)*, pages 91–100, 2002.

[7] K. J. Åström and B. Wittenmark. *Computer-controlled systems (3rd ed.)*. Prentice Hall Information and System Sciences. Prentice-Hall, Inc., USA, 1997. ISBN 0-13-314899-8.

[8] D. Goswami, R. Schneider, and S. Chakraborty. Relaxing Signal Delay Constraints in Distributed Embedded Controllers. *IEEE Transactions on Control Systems Technology*, 22(6):2337–2345, 2014.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[10] W. Chang, L. Zhang, D. Roy, and S. Chakraborty. *Control/Architecture Codesign for Cyber-Physical Systems*. Handbook of Hardware/Software Codesign. Springer, Netherlands, 2017. ISBN 978-94-017-7358-4.

[11] D. Tămaş-Selicean, P. Pop, and W. Steiner. Timing Analysis of Rate Constrained Traffic for the TTEthernet Communication Protocol. In *IEEE 18th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 119–126, 2015.

[12] D. Goswami, R. Schneider, and S. Chakraborty. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2011.

[13] A. Masrur, D. Goswami, R. Schneider, H. Voit, A. Annaswamy, and S. Chakraborty. Schedulability analysis of distributed cyber-physical applications on mixed time-/event-triggered bus architectures with retransmissions. In *6th IEEE International Symposium on Industrial and Embedded Systems*, pages 266–273. IEEE, 2011.

[14] A. Masrur, D. Goswami, S. Chakraborty, J. Chen, A. Annaswamy, and A. Banerjee. Timing analysis of cyber-physical applications for hybrid communication protocols. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1233–1238. IEEE, 2012.

[15] G. Chesi, P. Colaneri, J. Geromel, R. Middleton, and R. Shorten. Computing upper-bounds of the minimum dwell time of linear switched systems via homogeneous polynomial Lyapunov functions. In *American Control Conference (ACC)*, pages 2487–2492. IEEE, 2010.

[16] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3): 239–272, 2007.

[17] M. Balszun, D. Roy, L. Zhang, W. Chang, and S. Chakraborty. Effectively Utilizing Elastic Resources in Networked Control Systems. In *IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017.

[18] O. Mason and R. Shorten. On common quadratic Lyapunov functions for stable discrete-time LTI systems. *IMA Journal of Applied Mathematics*, 69(3):271–283, 2004.

[19] M. Ahmed, C. Saraydar, T. El Batt, J. Yin, T. Talty, and M. Ames. Intra-vehicular Wireless Networks. In *IEEE Globecom Workshops*, pages 1–9. IEEE, 2007.