JUL 27 2004

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	26.Jul.04	MAJOR REPORT

4. TITLE AND SUBTITLE
CO-DESIGN OF REAL-TIMECOMMUNICATION AND CONTROL IN A WIRELESS NET WORKED CONTROL SYSTEM

5. FUNDING NUMBERS

6. AUTHOR(S)
MAJ GRAHAM SCOTT R

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF ILLINOIS AT URBANA

8. PERFORMING ORGANIZATION REPORT NUMBER
CI04-507

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

20040730 054

14. SUBJECT TERMS

15. NUMBER OF PAGES
20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

THE VIEWS EXPRESSED IN THIS ARTICLE ARE THOSE OF THE AUTHOR AND DO NOT REFLECT THE OFFICIAL POLICY OR POSITION OF THE UNITED STATES AIR FORCE, DEPARTMENT OF DEFENSE, OR THE U.S. GOVERNMENT.

Co-design of Real-time Communication and Control in a Wireless Networked Control System

Sumant Kowshik Scott Graham Girish Baliga Lui Sha Marco Caccamo
University of Illinois at Urbana-Champaign
{kowshik,srgraham,gibaliga,lrs,mcaccamo}@uiuc.edu

Abstract

Networked control systems such as traffic control consist of multiple autonomously controlled objects (plant) interacting with each other. A controller in a networked control system receives real-time updates from the plant and the environment and sends controls to the plant. In designing a controller of such a system, there has been a separation of concerns between: (a) real-time communication protocol and the state of the plant and (b) the communication protocol and the control algorithm in the controller.

In this work, we develop a cross-layer design of the real-time communication scheduling protocol on a shared wireless channel where the period of the updates change depending on the environment and a control algorithm that ensures safety. Specifically, we develop this design in the context of a traffic control testbed, where a set of cars travel autonomously along assigned paths as fast as possible without collision. We have developed a collision prediction algorithm based on bounding box estimates of neighbor position that can be used to change update periods of cars in *collision range* dynamically. Our real-time communication protocol is a modification of implicit-EDF that allows dynamic period changes while handling message losses. The controller implements a safety criterion that avoids collisions between cars.

1 Introduction

Networked control systems represent the next logical step of the information technology revolution, namely interaction with the physical world. These systems consist of collaborating autonomously controlled devices

(*plant*) sharing resources and communication bandwidth with each other. In a system where the controller is on a mobile device, the communication between the controllers is on a wireless medium. These systems interact with the "real-world", receiving updates from, and sending *controls* to the plant. Thus, they have safety and real-time requirements. In general, there is a *separation of concerns* between the communication of updates from the environment or the plant and the control algorithm which determines the performance or the gain of a controller. In a typical design of a controller, the control algorithms are designed to tolerate degradation in the real-time communication of updates by degradation of its performance. At the very least, the controller maintains certain safety requirements in the presence of delayed or lost updates.

However, there is a strong interplay between the communication of updates to a controller, its controller performance, and the current state of the plant. On one hand, the controller degrades its performance when it does not receive updates or has uncertain or outdated information about the controlled object and its environment. On the other hand, changes in the state of the plant determine the real-time requirement of plant updates by the controller. In this work, we consider cross-layer design of the real-time communication protocol for the updates and the control algorithms in the controller based on the state of the plant.

Our motivating example is a traffic control testbed developed at the IT convergence laboratory at the University of Illinois [5] as a prototype networked control system. The testbed consists of a set of cars operating on an indoor track. Each car is independently controlled by its own software controller that using feedback about its position from sensors. Our specific experiment consists of a set of cars autonomously driving along individual paths as fast as possible without collisions. The cars are the controlled objects which communicate their position to all the other cars periodically over a wireless network. The controller of each car needs to dynamically change its update period depending on its relative position to the other cars and also implement a safety mechanism to avoid collisions if there is a degradation in the communication of updates. This system and the experiment are described in further detail in section 2.

Our approach is a concomitant design that addresses two specific problems in this paper. First, we develop a real-time communication protocol based on implicit-EDF [9] for the timely communication of position updates to neighboring car controllers on a shared channel in a broadcast wireless network in the presence of dynamic changes in the real-time requirement on updates from a car. The changes in period are computed based on the constantly changing environment of a car, i.e., its relative position to the other cars. In particular, a car has a shorter update period when it is within *collision range* of one of its neighbors. We develop a mechanism to detect if a car is within collision range of its neighbor which addresses the asym-

metry of position estimates in different car controllers. The communication protocol and the mechanism to detect collisions are tolerant to message losses in the wireless network. Secondly, we implement a safety mechanism in the controller to avoid collisions if timely updates are not received by the controller, or if an actual collision condition exists.

The key contributions of our work are summarized below:

1. A modified implicit-EDF protocol for wireless networked control that handles dynamic changes in update periods due to environmental changes at short ranges.

2. A car collision prediction mechanism that dynamically determines the real-time period of individual updates, in the presence of asymmetries in the position estimates in neighboring cars.

3. Co-design of the communication protocol, collision prediction, and the control algorithm for safety in wireless networked control systems, illustrated using our experimental setup.

The following section qualitatively formulates the problem and provides an overview of our approach. Section 3 formally states the problem and our assumptions about the underlying network and the individual car controllers. Section 4 describes our techniques for the co-design of communication and control based on the state of the controlled object. Section 5 concludes providing directions for future work.

2 Design Problem and Approach Overview

In this section, we motivate the design problem in networked control systems addressed in this work using the case of our traffic control testbed. We then describe the problem and provide an overview of our approach.

2.1 Motivation: Traffic Control Testbed

Our canonical example of a networked control system is a traffic control testbed built as a research prototype [5]. A description of its software architecture can be seen in [1]. The testbed used in this work consists of a fleet of small cars traveling along individual paths on an indoor track. Each car is controlled by a dedicated controller which periodically computes a series of controls and sends them over a dedicated RF channel to the car with negligible latency. A combination of on-board and off-board sensors (e.g. GPS)

sends the position and the orientation of a car as feedback periodically. The positions and orientations of the other cars in the testbed are periodically broadcast by the car over a shared channel of an ad hoc wireless network. In this configuration, each car has fairly accurate estimates of its own position and orientation, but not that of its neighbors. The controllers receive a path from a higher level path server. The *objective* of our experiment is that each car should reach the destination along its path in the shortest time without collisions. The controllers of the cars determine the relative position of each car by the periodically broadcast updates.

2.2 Problem Description and Approach

Our configuration consists of a set of autonomously controlled cars traveling along assigned paths. Each car can vary its speed from 0 (stopped state) to a certain maximum value. We encounter three challenges in designing this system. First, each car controller periodically needs to communicate its position data as updates to all the other car controllers over a wireless network. These updates have a real-time requirement with a deadline equal to their period of communication. We need to design a communication protocol to ensure that these real-time requirements are fulfilled. Secondly, if two cars are predicted to be within a certain range, they are said to be in *collision range*. When two cars are in collision range, the frequency of their position updates needs to increase. In other words, the deadlines of the updates shrink. Each controller must determine the new update periods and the communication protocol should be able to handle dynamic changes in periods of communication. Finally, each car controller needs to avoid collisions by stopping the car before a potential collision as a *safety mechanism*.

In order to address the above challenges, we have designed a distributed protocol for timely communication of periodic updates on wireless channels in the presence of dynamic changes to the period. Also, we employ a safety mechanism that stops the car before a collision occurs. These algorithms are implemented by all the cars in the system. Specifically, we have developed the following mechanisms:

(a) Determining initial update periods and scheduling these updates on a wireless channel in real-time using Implicit-EDF [9]. (Section 4.2 and 4.3.1)

(b) Determining *bounding boxes* around a car through position estimation and prediction and using these bounding boxes to predict potential collisions – This is used to determine the updates whose period need to be increased/decreased as well as the new update period values. (Section 4.1)

(c) Dynamically changing the update periods in the communication scheduling protocol and computation of the new communication schedule. (Section 4.3.2)

(d) Developing the safety mechanism to avoid collisions. (Section 4.5)

(e) Analyzing the effect of message losses due to wireless links on the above design.(Section 4.4)

In this work, we adopt a simplistic approach to the bounding box computation ((b) above) and the speed control. We focus primarily on the real-time scheduling protocol and its interaction with the dynamically changing neighborhood of each car.

3 Mathematical Model

In this section, we mathematically define the different components in the system and formulate the scheduling problem. We also describe our assumptions about the networked control system and the underlying wireless network. Our configuration consists of n autonomously controlled cars Car_i, where $1 \leq i \leq n$, each traveling along different assigned paths. The goal of each car is to travel as fast as possible along its path without colliding with any other car.

Each car Car_i is controlled by a controller that contains the following *state* information:

- $path_i$ defined as a series of (x,y) co-ordinates along which the car travels.
- $< x_i, y_i, \alpha_i >_t$, the (x,y) co-ordinates and the orientation of Car_i at current time t.
- sp_i, the speed of Car_i at time t, where $0 \leq sp_i < MaxSpeed$.
- $< x_k, y_k, \alpha_k, t_k >$, the latest update from Car_k containing its (x,y) co-ordinates and the orientation received at time t_k, where $1 \leq k \leq n$ and $k \neq i$.

Each car controller contains a model of the controlled car. The controller uses this model to periodically compute a series of control outputs as a function of its current position and its assigned path. For instance, in the testbed, we use a model predictive controller [3]. This controller computes a large set of potential future control outputs for a certain number of time slots into the future by exhaustively generating control output sequences. The different control sequences are evaluated by measuring the mean square deviation from the path. The control output sequence with the least deviation is sent to the car, resulting in changes to

the speed and the direction of the car. The latency of the control outputs sent by the controller is negligible. We assume that Car_i has a control output period of T_i^o. We assume a minimum speed of 0 (stopped), a maximum speed of $MaxSpeed$, and a maximum rate of change of orientation τ for the car

Local updates from sensors about the position of a car to its controller are sent more frequently (with a much smaller period than the period of the updates sent by the other cars). Also, the state estimator in the car controller uses the previous updates and the control outputs sent since that update to estimate the current position of the car. This is discussed in section 4.1 Thus, the estimate of the position of a car in its own controller is assumed to be accurate.

Each car, Car_i communicates its position and orientation, $< x_i, y_i, \alpha_i >$ periodically with a period T_i, which can change dynamically taking values from $MinPeriod$ to $MaxPeriod$. The communication medium is a broadcast wireless ad hoc network organized as hexagonal cells [9, 2]. For this work, we assume that all the cars are within a single cell of such a network. This is reasonable because the broadcast range of these networks is much larger (around 250m) than the distances between cars that can potentially collide. [6] Below we list our assumptions about the communication medium

A1 All nodes communicate using a single channel inside the cell. Thus, a common power level is used such that all nodes are within transmission range of each other.

A2 The clocks of the nodes in the network are synchronized [5].

Throughout the paper, it will be assumed that time is divided in frames and that all network nodes are synchronized on a frame basis. A single update packet can be sent during each frame and these packets have a constant size. The node can immediately broadcast an update upon its turn since reliable position and orientation information is available to each node at any time.

4 Mechanisms for Communication and Control

In this section, we describe our techniques to schedule real-time communication of updates on a wireless medium in the presence of dynamic changes to the update periods due to collision prediction and our control algorithm that implements the safety mechanism. We also describe a technique to predict collisions and a protocol to dynamically increase or decrease the period of updates from a car when a collision is predicted. Finally, we examine the effect of message losses on the communication protocol.

4.1 State Estimators and Predictors: Collision Prediction

We first describe the technique used by individual car controllers to predict collisions. Broadly, each car controller conservatively predicts its own position space using its sensor updates and the controls sent to its car since the update and the position space of its neighbors using the last received update and its maximum speed and rate of change of orientation. For simplicity, we represent these position spaces using conservative bounding rectangular boxes.

Each car controller tolerates delays and noise in the updates received from the sensors on its own car using a Kalman filter [4] as a *state estimator*. The state estimator estimates the state of its own car (position and orientation) as a function of the sensor feedback and controls sent to the actuator since the last sensor update received. In our testbed, we use a calibration of the steering angles and the speed of the car for different control outputs to estimate the effect of the controls sent since the last update from the sensor. In formal terms, an extended Kalman filter maintains a priori and a posteriori state estimates $\tilde{x}$ and $\hat{x}$, which are updated using the following difference equations:

$$\tilde{x}(t) = f(\hat{x}(t-1), u_p(t-1), w(t-1)) \tag{1}$$

$$\hat{x}(t) = h(\tilde{x}(t), I_p(t), v(j)) \tag{2}$$

where the random variables w and v represent the process and measurement noise respectively and u represents the control outputs. Eq. 1 updates the estimate based on the previous control, while eq. 2 corrects the estimate using current observations I_p. The estimate $\hat{x}$ is the output of the filter and is used to compute controls. This state estimator, in combination with an on-board sensor providing frequent updates, provides the controller with accurate estimates of the position of its own car at any time. The same Kalman filter above can be used to predict the state of the car based on the future controls sent to a car. This functionality is termed *state prediction.*

The state of a neighboring car Car_k is predicted at the controller for Car_i as a function of its last received $< x_k, y_k, \alpha_k, t_k >$ update at time t_k, the maximum speed $MaxSpeed$, and the maximum rate of change of orientation, τ. This is because of the uncertainty in the movement of the neighboring car since the time of its last update.

Figure 1 illustrates the prediction of the position (state) of Car_k by Car_i at times t, $t + T_i^o$, and $t + 2T_i^o$, where t is the current time. The cars are represented as rectangles. As seen in the figure, Car_i has an accurate

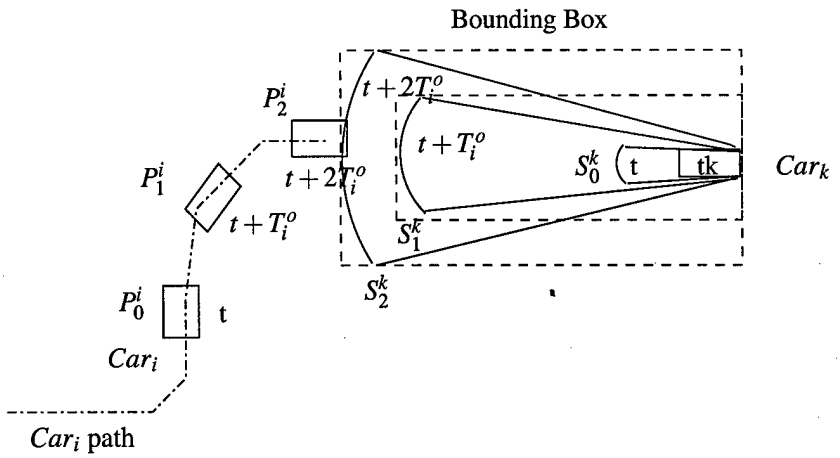


Figure 1: Bounding boxes of the predicted state space of Car_k in Car_i

estimate of its own positions (P_0^i, P_1^i, and P_2^i) using its on-board sensors and state estimator. The possible state space of Car_k, however, grows since t_k, the time Car_i received its last update from Car_k. The predicted positions of Car_k at time t are calculated to be *sectors of circles* with radius, $MaxSpeed \times (t - t_k)$, and a subtending angle, $2 \times \tau \times (t - t_k)$ (the car could turn left or right). The sectors, S_0^k, S_1^k, and S_2^k, represent the set of possible positions of Car_k at times t, $t + T_i^o$, and $t + 2T_i^o$ respectively. For simplicity of computation, we circumscribe a rectangular *bounding box* around the sector representing the set of possible positions. We use this rectangular bounding box as the estimate of position of Car_k in Car_i. Note that our protocol works for bounding boxes of any shape. We merely use the bounding box predictions to predict potential collisions. If there are no message losses in the medium, the bounding box grows to a size proportional to $MaxSpeed \times T_k$, where T_k is the period of updates sent by Car_k. In the presence of message losses the update from Car_k could be missed by Car_i and the bounding box for the position estimate of Car_k in Car_i grows until a Car_k update is received by Car_i.

We use predictive bounding boxes to detect potential collisions in the future. In figure 1, Car_i predicts that it potentially collides with Car_k at time $t + 2T_i^o$. Our collision prediction algorithm runs in the controller every control loop. The algorithm generates a series of bounding boxes for a neighboring car and its own car predicting the position of the cars at different times in the future in steps of the output period T_i^o until a maximum time interval, $MaxTimeInterval$. The bounding boxes for the positions of Car_i itself are predicted for a particular future speed sp_i, while the bounding boxes for the positions of the neighbor, Car_k are predicted assuming that it can travel at any speed between 0 and $MaxSpeed$. The algorithm checks that the

```
//  In Controller for Car i travelling at speed sp
for (each neighbor k of Car i)
   for (time T = CurTime to (CurTime + MaxTimeInterval) step output-period)
      if (IsCollision(BoundingBox(k, T), BoundingBox(i, T, sp)))
          Report ''Collision with k predicted at time T'';
          break;
   Report ''No collision with k OR Collision predicted at time
            CurTime + MaxTimeInterval''

IsCollision(BoundingBox B1, BoundingBox B2)
   B1bins : set of bins corresponding to BoundingBox B1
   B2bins : set of bins corresponding to BoundingBox B2
   if (B1bins INTERSECTION B2bins contains any bins)
      return true; // Potential collision
   else
      return false; // No collision
```

Figure 2: Algorithm to find the predicted collision time (PCT) of Car_i with each neighbor

bounding box for the neighboring car (Car_k) does not intersect with the bounding box of its own car. For this purpose, we have divided our entire track area into square cells called *bins*. Each bounding box is described by the set of bins it occupies. Our algorithm for predicting collisions is shown in figure 2. This algorithm returns the time at which Car_i predicts it will collide with each Car_k. This is defined to be the *predicted collision time* $PCT_i(sp_i,k,t)$ at time t, where Car_i is estimated to travel at a future speed, sp_i.

In figure 1, the predicted collision time $PCT_i(t,sp_i,k)$ is equal to $t+2T_i^o$. Decreasing the future speed of the car, sp_i increases $PCT_i(t,sp_i,k)$. Thus, $PCT_i(t,sp_i,k)$ can be computed for different potential speeds of Car_i. We use $PCT_i(t,sp_i,k)$ and $PCT_k(t,sp_k,i)$ to predict if two cars, Car_i and Car_k are going to be in collision range.

Collision Theorem: If two cars, Car_i and Car_k predict collision times of $PCT_i(t,sp_i,k)$ and $PCT_k(t',sp_k,i)$ with each other at time t and t' respectively, then the actual collision time is $\geq max(PCT_i(t,sp_i,k),PCT_k(t',sp_k,i))$. The above theorem can be proved by observing that the estimated bounding box of the potential position space of a car (neighbor or itself) is the superset of positions that the car can occupy at a particular time. We prove the above theorem assuming that the position of a car and the bounding boxes can be described by the bins it occupies

Proof of the Collision Theorem: Let $Bins_i(T)$ and $Bins_k(T)$ be the set of bins actually occupied by Car_i and Car_k at at time T.

Let $Bins_{ik}(T, sp_i)$ and $Bins_{ki}(T, sp_k)$ be the set of bins occupied by $BoundingBox_i(k, T, sp_i)$ and $BoundingBox_k(i, T, sp_k)$ as predicted by Car_i and Car_k respectively.

$Bins_k(T) \subseteq Bins_{ik}(T, sp_i)$ and $Bins_i(T) \subseteq Bins_{ki}(T, sp_k)$.

Case I: $PCT_i(t, sp_i, k) \leq PCT_k(t', sp_k, i)$

Consider the bounding boxes of the positions occupied by Car_i and Car_k at time $T < PCT_k(t', sp_k, i)$,

Since the earliest predicted collision time by Car_k, $PCT_k(t', sp_k, i)$, is $> T$, Car_k's estimate of the bounding box of Car_i does not intersect with its own bounding box estimate i.e. $BoundingBox_k(i, T, sp_k)$ does not intersect with $BoundingBox_k(T)$.

$\Rightarrow Bins_{ki}(T, sp_k) \cap Bins_k(T) = \phi$.

$\Rightarrow Bins_i(T) \cap Bins_k(T) = \phi$, since $Bins_i(T) \subseteq Bins_{ki}(T, sp_k)$.

$\Rightarrow$ there is no collision between Car_i and Car_k at time $T < PCT_k(t', sp_k, i)$

$\Rightarrow$ actual collision time $\geq PCT_k(t', sp_k, i)$.

Case II: $PCT_i(t, sp_i, k) > PCT_k(t', sp_k, i)$

Analogous to Case I, we can show that actual collision time $\geq PCT_i(t, sp_i, k)$

The collision theorem above proposes that the larger collision time prediction be used in the presence of asymmetry in the prediction of collision time between two cars. This is true due to the conservative nature of position estimation by our bounding box approach. The asymmetry in prediction arises from the uncertainty in the path and speed of the neighbor and even missed updates due to message losses. As an example, in figure 1, if Car_k simply stopped at time t, its $PCT_k(t, sp_k, i)$ would be $> t + 2T_k^o$, while $PCT_i(t, sp_i, k)$ would be $t + 2T_k^o$. In this case, we use the $PCT_k(t, sp_k, i)$ value as a prediction of the actual collision time. We use this theorem in our protocol to determine if two cars are within collision range.

Corollary: Two cars Car_i and Car_k are estimated to be in collision range if $max(PCT_i(t, sp_i, k), PCT_k(t', sp_k, i)) < CurTime + CRT$, where $CurTime$ is the current time, t and t' are the times at which Car_i and Car_k predict collision, and CRT is defined to be the *collision range threshold.*

In the following subsections, we describe the periods of the updates sent by a car and use the above corollary to determine when to dynamically change the period of updates from a car. We increase the frequency of updates when two cars are within collision range.

4.2 Update Periods

Initially, all the cars are assumed to have the same update period *MaxPeriod*. If two cars are within collision range, their update periods need to shrink in order to obtain more reliable position information about each other. In this work, we assume that the update periods of the cars form a *harmonic series*. Thus, the periods are designed to take the following discrete values $\{MaxPeriod, MaxPeriod/2, MaxPeriod/4, \ldots, MinPeriod\}$. In the event that the distance between Car_i and Car_k decrease such that they are detected to be within collision range, both their periods are halved. A schedule containing a set of tasks with harmonic periods gives us the following useful property:

HP1. The *hyperperiod*[1] of the schedule is equal to the longest period.

The property, HP1, follows from the length of the hyperperiod simply being the least common multiple of the different periods of the tasks. In this case, the least common multiple is simply the longest period.

4.3 Scheduling Periodic Updates

Our fundamental scheduling problem is to schedule periodic update packets on an ad hoc wireless broadcast network, where all the nodes are within broadcast range of each other, i.e. whenever a node transmits, all nearby nodes receive the signal if collisions are avoided. We also require the protocol to allow dynamic changes in periods based on collision distances. In this subsection, we describe the protocol assuming that there are no message losses. Later, in section 4.4, we describe the effect of message losses.

4.3.1 Implicit-EDF

Caccamo *et al.* described an implicit prioritized access protocol for wireless sensor networks in [9]. This protocol uses a distributed scheme called *Implicit-EDF* based on earliest-deadline-first (EDF) [7] and does not rely on the existence of a base station which constructs a centralized scheduling algorithm. Other MAC protocols for real-time communication on a wireless channel are either centralized, assume infrastructure or have a control overhead [10, 8, 13]. The key idea of implicit-EDF is to replicate the EDF schedule at each node for packet transmission. If the schedules are kept identical, each node will know which one has the message with the shortest deadline and has the right to transmit next. While alternate scheduling protocols based on local carrier sensing and on the exchange of control packets reduce the probability of conflicts, they

[1] A hyperperiod is defined as the time interval corresponding to the single repeating sequence of a static schedule

do not eliminate it. A real-time property requires that a deterministic contention free schedule is developed. For instance, suppose each node is given a message table as shown in Figure 3, the same schedule is derived by every node in the cell according to EDF (deadlines ties are broken in favor of the node with the highest rank). If a node is not listening to the channel, it is able to select the right frame to transmit just by counting the frames and assuming that all previous messages used all their reserved frames.

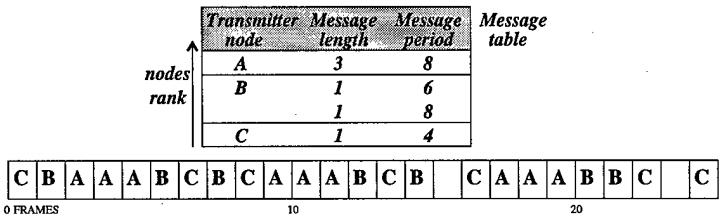


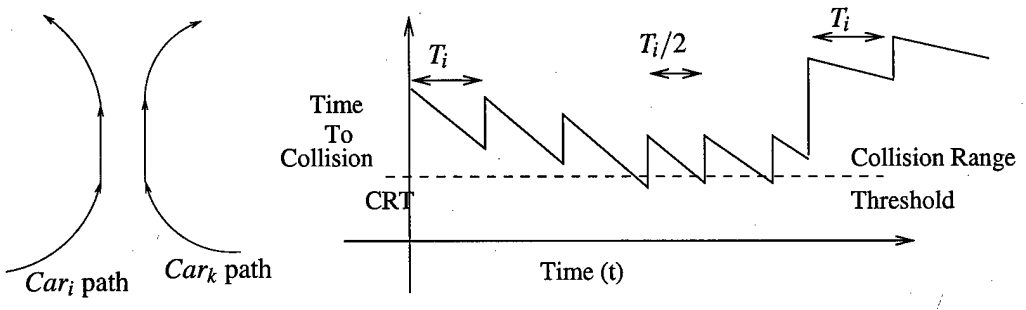
Figure 3: Example of implicit contention using EDF.

We use the above protocol to schedule the periodic updates from the cars. As described in [9], the nodes in the network need to be initialized by transmitting the message table to all the nodes in the network. In practice, all the cars are assumed to start communicating their updates using the same period. However, in the event of a car requesting a higher period of update from a neighbor (when within collision range), the message table and the previously calculated schedule needs to be modified. Below, we describe an enhancement to implicit-EDF that allows dynamic period changes.

4.3.2 Initiating a Period Change

As described in the corollary to the collision theorem in section 4.1, two cars Car_i and Car_k are in collision range at time $CurTime$, if $max(PCT_i(t, sp_i, k), PCT_k(t', sp_k, i)) < CurTime + CRT$, where CRT is the collision range threshold. We decrease the period of broadcast updates of a car when it is within collision range of any of its neighbors. In this work, we use a single threshold CRT to determine if two cars are within collision range. Thus the period of updates of a car takes only two values. For Car_k, the initial period of broadcasted updates is T_k, which shrinks to $T_k/2$ when it is within collision range of another car. A straightforward modification of the described techniques can be used if multiple collision thresholds and multiple periods of updates are used.

We decrease the period of both the cars Car_i and Car_k if they are detected to be within collision range at time, $CurTime$, provided that Car_i or Car_k are not already in collision range of a third car. Since our



(a) Car Paths

(b) Time To Collision for Car_i with Car_k

Figure 4: Variation of time to collision and period of Car_i

periods are harmonic, we simply halve both T_i and T_k. Thus, a new schedule is computed by each car using the modified periods by each car and this schedule is deployed. The threshold CRT determines the time at which Car_i and Car_k are in collision range, at which point the new schedule is computed and the period change for Car_i and Car_k is effected. CRT should be large enough that Car_i and Car_k broadcast updates with the new period before $max(PCT_i(t, sp_i, k), PCT_k(t', sp_k, i))$. Later in this subsection, we derive a formal bound on CRT in the context of our protocol. After the two cars are out of collision range, their periods need to be increased (in this case doubled). Figure 4(a) shows Car_i and Car_k traveling along paths that gradually bring them closer into a narrow passage. The period of updates of Car_i is initially T_i. In the time-line snippet shown in figure 4(b), Car_i is within collision range of Car_k once its time-to-collision (PCT - current time) is less than CRT. This reduces its period to $T_i/2$. When Car_i is out of collision range with all cars, its period is increased to T_i.

We now describe the modified implicit-EDF protocol to allow dynamic changes in the update periods assuming no message losses. Later in section 4.4, we describe a modification to the above protocol to handle lost updates. Initially the cars use implicit-EDF and broadcast their updates as per the schedule. In each update packet broadcast by a car, Car_i, during its assigned slot in the schedule at time t, we include a header consisting of its predicted collision times, $PCT_i(t, sp_i, k)$, with each of the neighboring cars, Car_k. Thus, the predicted collision time estimated by each car is broadcast to all the other cars as a part of its update. Each car maintains a table, $PCTTable(i, j), i \neq j$, where the entry corresponding to the ith row and the jth column is the latest $PCT_i(t, sp_i, j)$ value broadcast by Car_i. From the corollary to the collision theorem in section 4.1, at time $CurTime$, Car_i and Car_j are within collision range if $max(PCTTable(i, j), PCTTable(j, i))$ is less than $CurTime + CRT$. This condition is checked for each pair of cars just before an idle slot after a scheduled

slot in the existing schedule or at the end of a hyperperiod. If two cars are detected to be within collision range, their period is halved unless they are already in collision range in the current schedule. The feasibility of scheduling the update with the decreased period is checked before it is incorporated in the schedule. Since each update is of unit length, using the classical Liu and Layland condition, the schedulability condition is given by $\sum_{i=1}^{n} \frac{1}{T_i} \leq 1$. Each controller updates the periods of the cars in a fixed order which guarantees that the schedule is computed uniformly in each car controller. If a car is out of collision range with all its neighbors, its period is increased (doubled). If any of the periods have changed from the previous schedule, a modified EDF schedule is computed. The new schedule comes into effect immediately after the idle slot when the above algorithm is run before an idle slot, or in the new hyperperiod when it is run at the end of a hyperperiod. Figure 6 illustrates the initiation of period change. The new schedule shown in time-line (b) represents the schedule if the period of A and B are halved (from 8 to 4) while the period of C remains the same. The algorithm runs just before the idle slot after C is scheduled and the new schedule is deployed at the end of the idle slot.

The *period change algorithm* shown in figure 5. The function `isSchedulable(car, oldPeriod, newPeriod)` checks the schedulability condition given above and returns true if the new period is schedulable. The algorithm in figure 5 traverses the cars in a fixed order. While this ensures that each car controller computes the same schedule, it does not ensure fairness. The traversal order can be changed deterministically using a circular list of cars in order to remedy this problem. Also, the algorithm can be modified to first increase the periods of the cars that are out of collision range thus freeing up bandwidth, following which, the requirement to decrease periods can be fulfilled.

The algorithm initiates an increase in the frequency of updates from cars that are within collision range of other cars, provided the changed period is schedulable. We assume that only a small fraction of the cars in the system are in collision mode at any time. Thus, any request for an increase in frequency for updates is schedulable. While this assumption is optimistic, we avoid collision in *all* cases due to our safety mechanism described in section 4.5. The *period change latency (PCL)* is defined as the interval between the time τ at which $max(PCT_i(t, sp_i, k), PCT_k(t', sp_k, i))$ became less than $\tau + CRT$ and the time at which T_k and T_i were incorporated into a new schedule. In the worst case, this is the interval between two consecutive runs of the period change algorithm, assuming that the schedulability check did not fail. The longest interval between consecutive runs of the period change algorithm is equal to a hyperperiod which occurs for a schedule whose utilization is 1. Thus, in the worst case, the period change latency is equal to the longest period. In practice,

```
// Run in all car controllers before any idle slot in the schedule and at
// the end of a hyperperiod
CurTime       : Current time
CRT           : Collision range threshold
PCTTable(i,k) : Table containing latest predicted collision times broadcast
inCollision(i): True if car i is currently in collision range with some car
                False otherwise

ScheduleChange = false;
for (each car, Car i), 1 <= i <= n
   for (each neighbor Car k, 1<= k <= n, k != i)
      if (max(PCTTable(i,k), PCTTAble(k,i)) < CurTime + CRT)
         // Decrease their periods if they are not in collision range already
         if (!inCollision(i) && isSchedulable(i, T(i), T(i)/2))
            T(i) = T(i)/2
            inCollision(i) = true
            ScheduleChange = true
   if (max(PCTTable(i,k), PCTTable(k,i)) >= CurTime + CRT)) forall cars k
      // Increase its period if its out of collision range with all
      // its neighbors
      T(i) = T(i) * 2
      inCollision(k) = false
if (ScheduleChange)
   Adopt new EDF schedule after idle slot or in next hyperperiod
```

Figure 5: Period Change Algorithm run at all car controllers before an idle slot in the schedule

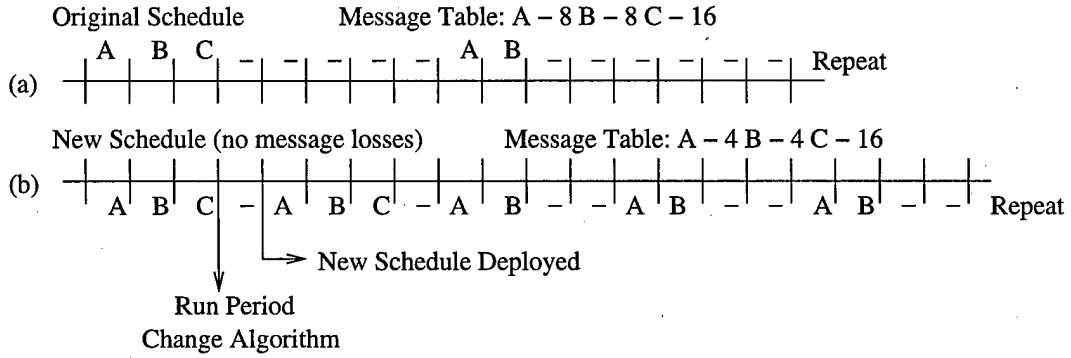


Figure 6: Illustration of period change algorithms in a schedule without message losses

however, the utilization is less than 1. In these cases, the period change latency is typically much less than the length of the hyperperiod since the period change algorithm is run before the idle slots following scheduled slots.

We use this bound on the period change latency to derive a constraint on the collision range threshold

time (*CRT*). We discussed earlier that two cars that enter collision range should exchange updates using the new period before collision time. Hence, $CRT > PCL + new\ hyperperiod$ or

$$CRT > Old\ hyperperiod + new\ hyperperiod$$

Since we use harmonic periods, *CRT* should be fixed to be twice the length of the maximum period. In our system, this is equal to twice the common initial update period, T_i.

In the following subsection, we examine the effect of lost messages on our protocol.

4.4 Effect of Lost Messages

Due to the unreliability of the wireless channel, update broadcasts are unreliable and can thus be missed by individual cars. In this subsection, we describe the effect of missed updates on our collision detection algorithm, implicit-EDF, and our period change algorithm. We propose a modification to the period change algorithm in order to make it work in the presence of message losses.

Collision Detection

Our collision detection algorithm, which estimates the position space of a neighboring car at a particular time with a bounding box around the position space, is robust to missed updates. The size of the bounding box depends on the time the last update was received. $T_i(j)$ represents the time the last Car_j update received by Car_i. In the presence of missed updates from a neighboring car, the bounding box of the neighbor grows. The length of the rectangle representing the bounding box around Car_j as predicted by Car_i is proportional to $MaxSpeed \times T_i(j)$. The width of the rectangle is proportional to $2 \times cos(\tau \times T_i(j)) \times MaxSpeed \times T_i(j)$, where τ is the maximum rate of change of orientation. A bigger bounding box estimate results in a smaller *PCT*. We demonstrate the effect of missed updates using the following examples.

Example 1: If all the updates between two cars, Car_i and Car_k were lost, each car's bounding box estimates of its neighbor grows until it intersects with its own bounding box estimate. Eventually each car stops due to our safety mechanism.

Example 2: If there are two cars, Car_i and Car_k. If Car_i misses an update from Car_k, but Car_k does not miss the update from Car_i, then $BoundingBox_i(t, sp_i, k)$ grows while $BoundingBox_k(t, sp_k, i)$ does not. Thus, potentially $PCT_i(t, sp_i, k) < PCT_k(t, sp_k, i)$. In this case, the collision theorem predicts that the collision time is, in fact, at least $PCT_k(t, sp_k, i)$. Upon receiving the next update Car_i's bounding box estimate of Car_k's position becomes small again.

Communication Protocol

Our communication protocol uses implicit-EDF with a modification in packet format in order to notify the *PCT* values of individual cars used to dynamically change the update period. If the periods are fixed, implicit-EDF addresses the issue of message losses [9]. Since the schedule is statically determined, each car simply waits for its slot in the schedule to broadcast its update. During the remaining slots, the car receives broadcast updates from its neighbors. Even if an update is missed, the schedule is followed strictly and implicitly by each car. However, in our modified algorithm with dynamic changes in period, a loss of an update by a single car means that the *PCTTable* entries are not the same in each car controller. Thus, the changes to the update period in different nodes can vary leading to conflicting schedules in each car. E.g. If Car_i missed updates from Car_j and Car_k within a single hyperperiod where $PCTTable(j,k)$ and $PCTTable(k,j)$ values fall below the collision range threshold, CRT, then Car_j and Car_k halve their periods and compute a new schedule, while Car_i uses the old schedule. This could result in collisions on the wireless channel.

We now describe a *black-burst based period change algorithm* that works correctly in the presence of message losses. In this algorithm, the period change algorithm shown in figure 5 is run just before idle slots after a scheduled slot in the schedule and at the end of hyperperiods, as before. However, a car jams the channel with a pulse of energy called a *black-burst (BB)* [11, 12] during an idle slot and at the end of a hyperperiod if it missed any updates since the last idle slot or end of hyperperiod. Upon sensing a BB on the network during an idle slot or at the end of a hyperperiod, all nodes discard the *PCT* updates since the last idle slot or end of hyperperiod and the new schedule (if any), and continue as per the old schedule. Thus, a single car that missed an update can stop the remaining cars from adopting a new schedule, thus keeping all the nodes in synchrony and avoiding schedule conflicts. If all the nodes receive all the broadcast updates, then no node broadcasts a BB during the following idle slot or end of hyperperiod, thus leaving it idle. In this case, the new schedule (if there is one) is deployed in the slot immediately after the idle slot or at the beginning of the next hyperperiod. The BB can be sensed reliably by each node in the network. Carrier sense multiple access (CSMA) is a popular MAC protocol among ad hoc wireless networks. The black-burst based period change algorithm is illustrated in time-line (b) in figure 7. Here, one of the nodes A,B or C (say A) misses an update and broadcasts an energy pulse due to which the the new schedule and *PCT* values of A,B, and C received in the previous three updates are discarded. The old schedule is continued. The period change algorithm is run again just before an idle slot after A and B are scheduled again.

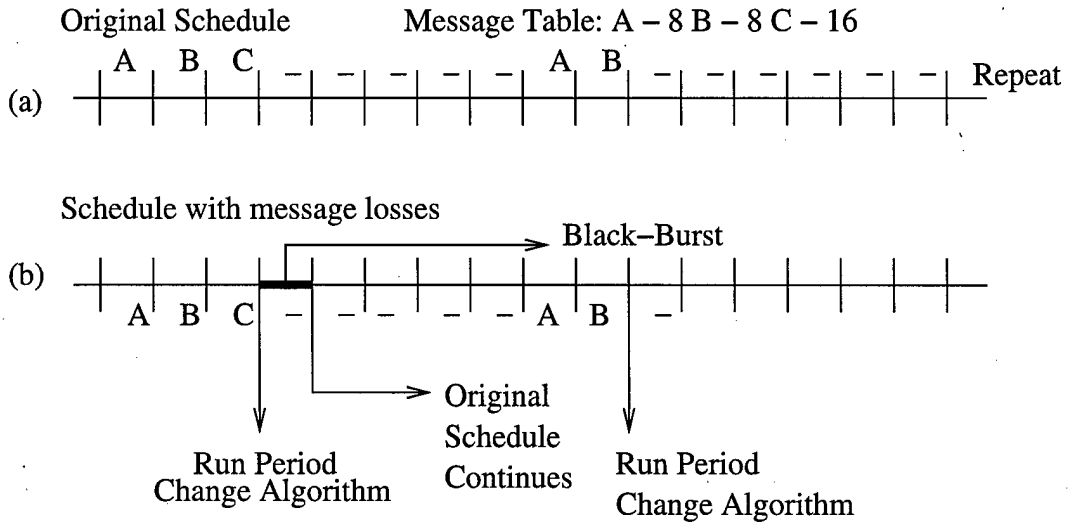


Figure 7: Illustration of period change algorithms in a schedule with message losses

In order to allow a potential BB slot at the end of each hyperperiod, each car assumes a dummy BB car (with lowest priority) whose period is equal to that of the maximum update period among all the cars. The slot corresponding to the dummy BB car in the EDF schedule is exchanged with the last slot in the hyperperiod. If no update is lost since the previous idle slot or hyperperiod, this slot is left idle and a new schedule can be deployed at the end of this slot. Otherwise, all nodes that missed updates transmit a BB during that slot.

The period change latency, *PCL*, can be potentially unbounded in the presence of message losses. Using our technique, in the worst case, a single loss of an update can cause the *PCL* to be increased by a hyperperiod (the worst case occurs when the utilization is 1 and the period change algorithm is run only just before the dummy BB slot). The expected value of *PCL* is given by:

$E(PCL) = p + 2p(1-p) + 3p(1-p)^2 + \ldots = 1/p$

where p is the probability that there are *no* missed updates during a hyperperiod.

4.5 Control Algorithm and Safety

In our experiment, an individual car predicts collisions due to the following reasons: (a) actual collisions in the paths of the individual cars, (b) missed broadcast updates from neighboring cars, and (c) inability to change the update period due to schedulability constraints or message losses. In order to protect itself in each of these cases, every car controller implements a safety criterion to avoid collisions. In this work, due

to the objective of our experiment being that the car travel along its path as fast as possible, each car simply travels at its fastest speed except when the safety criterion is enabled, at which point it stops.

In the event that a collision is predicted within the following control output period T_i^o, the car (Car_i) can be stopped immediately by sending the corresponding control output. This is necessary for safety and to guarantee no collisions. However, due to inertial effects the car would slow down to a halt over m output periods leading to the following safety criterion.

Safety criterion: If a car Car_i is predicted to potentially collide with another car within a time interval mT_i^o ($< CRT$), where mT_i^o is the braking time of the car, the control outputs to STOP within the time interval needs to be immediately sent to the car.

Since collision prediction is assumed to run on all the cars in the system, in the event of a potential collision between cars, both cars stop in their path. At this point, a higher level planner would need to re-route the cars or establish a protocol by which the cars go one after another. Such a protocol however is outside the scope of this work.

In our work, we have assumed that the car travels as fast as possible and comes to a halt when it senses a collision within its braking time. However, in practice, a secondary objective of driving the car smoothly may be important. The speed variation in these cases can be independently varied by the controller. Another opportunity for co-design of control and communication is seen when a period change cannot be effected in the communication protocol due to inschedulability or due to missed updates. During this period, the car can be slowed down until the period change is effected in order to avoid coming to a complete stop. Both of these fulfill the secondary objective of smooth motion of the car and constitute future work.

5 Conclusions and Future Work

In this work we described a cross-layer design in a networked control system such as a traffic control testbed where we designed a real-time protocol for scheduling communication of updates to a car controller with dynamic real-time requirement based on the environment (position of the other cars), an algorithm for predicting collisions in the presence of asymmetric position estimates among different cars, which is used to change the period of updates, and the safety mechanism in the controller to avoid collisions. In our integrated system with all the above mechanisms, the period of update communication and hence its real-time requirement is varied depending on the state of the cars in the system. Also, the controller itself degrades

its performance and stops as a safety mechanism in the absence of timely updates or in the presence of actual collision. Our communication protocol is based on implicit-EDF with the modification for dynamically updating periods. We enable the application of implicit-EDF by the collision theorem which determines uniformly in each node if a cars is within collision range of its neighbor, which determines the period of the updates from the car. Our real-time communication scheduling protocol for the updates, a modified version of the period change algorithm based on black-bursts, and the control algorithm are tolerant to message losses.

In the current work, we simplified our position estimates as bounding boxes and our speed control to simply implement a safety mechanism As future work, we intend to develop a sophisticated control algorithm (in a more complicated experiment perhaps) which dynamically optimizes the speed based on the timeliness of the updates received by the controller. We also intend to extend our cross-layer design to cars travelling across cells of an ad hoc network leading to dynamic entry and exit of cars within a single cell.

References

[1] Girish Baliga and P. R. Kumar. A middleware architecture for federated control systems. In *Proceedings of Middleware*, Rio Di Janiero, Brazil, 2003.

[2] Marco Caccamo and L.Y. Zhang. The capacity of implicit edf in wireless sensor networks. In *IEEE Proceedings of the 15th Euromicro Conference on Real-Time Systems*, Porto, Portugal, 2003.

[3] E. F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer Verlag, June 1999.

[4] C. K. Chui and G. Chen. *Kalman filtering with real-time applications*. Springer-Verlag, 1999.

[5] S. Graham and P. R. Kumar. Time in general-purpose control systems: The control time protocol and an experimental evaluation. In *Submitted to IEEE Conference on Decision and Control*, 2004.

[6] IEEE Computer Society. IEEE standard 802.11: wireless lan medium access control (MAC) and physical layer (PHY) specifications. The institute of Electrical and Electronics Engineers, New York, NY, 1997.

[7] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[8] I.Lee M. Adamou and I. Shin. An energy efficient real-time medium access control protocol for wireless ad-hoc networks. In *WIP session of the IEEE RTSS*, London, UK, 2001.

[9] Lui Sha Marco Caccamo, Lynn Y. Zhang and Giorgio Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *RTSS*, Austin, Texas, 2002.

[10] P. Pradhan and T. Chiueh. Real-time performance guarantees over wired/wireless lan. In *IEEE RTSS*, June 1998.

[11] J.L. Sobrinho and A.S. Krishnakumar. Real-time traffic over the ieee 802.11 medium access control layer. *Bell Labs Tech. J.*, 1:172–187, 1996.

[12] J.L. Sobrinho and A.S. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *IEEE Journal on Selected Areas in Communication*, 17(8):1353–1368, Aug 1999.

[13] R. Barua T.W. Carley, M.A. B and D.B.Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE RTSS*, Cancun, Mexico, 2003.