Experiments for Decentralized and Networked Dynamic Resource Allocation, Scheduling, and Control

Nicanor Quijano, Alvaro E. Gil, and Kevin M. Passino

Dept. Electrical Engineering
The Ohio State University
2015 Neil Avenue, Columbus, OH 43210

January 21, 2004

Introduction

The ubiquitous presence of networked computing is significantly impacting the field of control systems. There are already many distributed control systems (DCS) in industry and significant current research on networked multi agent systems. These networked control systems are often viewed as “complicated” since they are typically decentralized, large-scale, and hierarchical. Moreover, they typically have a blend of significant nonlinearities, high-dimensionality, nondeterministic behavior, random delays, and information flow constrained by the topology of the communication network. Since their main purpose is control of a dynamical system they contain many, if not all, of the challenges typically found in control systems (for example, disturbance rejection, tracking, robustness), and additional challenges due to the presence of a computer network that synergistically interacts with the dynamical system. They represent a significant departure from typical, say classical DC motor control or inverted pendulum control problems, and demand many of the same tools, skills, and more, such as expertise in software engineering, object-oriented programming, or

*This work was financially supported by the Intelligent Systems Division of the National Institute of Standards and Technology (NIST). For more information on the experiments described in this paper, including movies of their operation, see: http://eewww.eng.ohio-state.edu/~passino/distdynamicsyslab.html

†Send correspondence to K. Passino at (614)-292-5716, fax: (614)-292-7596, k.passino@osu.edu.
real-time operating systems (for a recent discussion on the “co-design” of software and controls for dynamical systems see [1]). Moreover, they demand that more attention be given to a number of other nontraditional control objectives, including dynamic resource allocation, scheduling of tasks, and control over large networks, than in the past [2].

While industry is bound to continue on this trajectory of exploiting networked computing for control of complicated dynamical systems, universities are left with significant challenges to re-orient their programs for the educational needs of future engineers in this area. This paper focuses on overcoming some of the challenges of providing a university laboratory experience to support this area. Two of the key challenges in implementing such a laboratory are (i) how to construct relatively complex networked control system experiments on a low budget (typically universities do not have access to the DCS in industry as they are typically very expensive) and (ii) how to design the experiments so that they present the new challenges in complex networked control systems such as dynamic resource allocation and scheduling. The experiments in this paper provide solutions to these two problems.

There are two other classes of networked control system problems that are not covered here, yet could provide meaningful solutions to the problem of providing a laboratory experience in this area. First, there is the area of cooperative robotics where networks of ground, underwater, or flying robots are given tasks such as cooperative pushing of an object, cooperative search, or “platooning.” This area is being given significant research attention elsewhere and there are many university courses around the world that provide excellent educational experiences in this area (for example, we have one here at OSU; see also http://www.ri.cmu.edu/lab_lists/index.html). Second, there is the research program on the Univ. of California Berkeley “motes” that holds the potential to study some networked control system problems (for example, tracking an evader moving through a field of sensors). Again, this approach is treated in detail elsewhere (see the web page of wireless sensor networks http://www.xbow.com/Products/Wireless_Sensor_Networks.htm) so we do not discuss it here. It is, however, interesting to note that if you are familiar with these two other areas you will immediately recognize the value of the types of experiments that are introduced here. The experiments we present are dominated by challenges due to dynamic subsystem interactions, air turbulence disturbances, sensor noise, popup targets, and network effects such as random but
bounded communication delays. Moreover, the experiments present new challenges for control
design including achieving a maximum uniform value using minimum resources, such as in the
balls-in-tubes and multizone temperature control problems, or cooperating to maximize point gain,
such as in the electromechanical arcade problem.

We begin by explaining the role of the experiments and laboratory described here in a university
curriculum. Next, we overview some special issues in the choice of laboratory software that must be addressed before establishing such a laboratory. Finally, we introduce the three laboratory experiments. For each case, at the end of the corresponding subsection we provide references to the literature that describes other methods that may be useful for the experiments.

Laboratory Role in University Curriculum

The Ohio State University Department of Electrical Engineering has a modern undergraduate controls laboratory that uses dSPACE hardware and software and Quanser experiments (see http://www.quanser.com). We have two dual-level graduate/undergraduate laboratories, but the majority of students taking these are graduate students, from mainly electrical and mechanical engineering, with typically more than a year of graduate course work in the control area. Both labs can be taken after the undergraduate course and do not require each other as prerequisites. One is a cooperative robotics lab (EE 757) where a range of control problems are studied, such as platooning and cooperative search, for ground-based small robots that are connected via a wireless ethernet. The second one (EE 758) is viewed as a “service course” to the control area to provide the students with laboratory experience in introductory nonlinear and robust control, along with more specialized topics such as games and intelligent control. The first half of the lab, which is 5 weeks as we are on a quarter system, is dedicated to learning dSPACE, data acquisition, modeling and system identification, PID control with anti-windup schemes, and the linear quadratic regulator (LQR) with an observer. We are working now to have a standard nonlinear control experiment all students perform. The experiments we use in the first half of the course are the ones by Quanser.

The second half of the laboratory is dedicated to in-depth study in a topic of interest to the student. If a professor wants their own advisee to study system identification, sliding mode control, adaptive control, intelligent control, game theoretic approaches, advanced robust control, or ad-
vanced nonlinear control in more depth they can assist in designing a set of experimental challenges for the student. We have some additional Quanser experiments that can be used for this second half of the lab, including the 2 degree-of-freedom helicopter, inverted cube, and tanks. Moreover, we can provide them with a sequence of projects in the use of the National Institute of Standards and Technology (NIST) Real-Time Control Systems (RCS) software library (see below) or LabVIEW. The “default” challenge sequence for the second half of the lab is, however, decentralized networked control, the topic of this paper. For this default sequence we typically give the students one of the experiments described here and require them to design, implement, and test resource allocators, schedulers, or controllers like the ones presented here. We often require them to design their strategies to overcome instructor-induced disturbances such as a puff of air on the multizone temperature control experiment. This often provides a nice “test” for their approach. In all cases they are expected to provide a demonstration and written report.

Laboratory Software for Coping with Complexity

One challenge that students, professors, or engineers in industry immediately encounter in this topical area is how to manage the complexity of designing and implementing hierarchical, decentralized, and networked control systems that are implemented across a range of different platforms and are therefore in need of inter-controller communications. Such features also create significant challenges for the professor who wants to establish an educational laboratory; hence, this issue must be addressed first, before experiments are designed and constructed. While there are a number of ways to cope with these problems, for a number of reasons [3], we advocate the use of the NIST RCS design methodology and software library. Since the full details of RCS are covered elsewhere [3], [4], here we only overview the adaptation of it for use in our laboratory with dSPACE software. We refer the interested reader to [3], [4] and to the web site given in the footnote of the title where the software library, a catalog of additional software modules, ideas on combining RCS with LabVIEW, and additional details on RCS are provided.

We use dSPACE hardware and software for all the experiments described in this paper; however, we have also implemented a number of strategies for multizone temperature control in LabVIEW (at the time of writing this, LabVIEW has a number of advantages in platform-to-platform communi-
cation functions relative to dSPACE). In both cases we adhere to the well-tested RCS methodology. Our experiments and software do not depend on using this particular design and implementation methodology but are enhanced by it. The dSPACE software is based on Matlab Simulink. To develop the block diagrams in Simulink for the experiments we use a “pre-process,” a “decision process,” and a “post-process” in a “control module” and several of these connected in a hierarchical and decentralized fashion. In the pre-process we will acquire the data, do some conversions, apply some scaling factors for the signals that are acquired, and store this information in some global variables that will be used in the decision process. In the decision process, we develop several subsystems that will be in charge of making decisions concerning the control, or other tasks such as to update variables to be used in the post-process. The post-process will send the data to other computers, if needed, and will update the digital outputs.

RCS software has “design and diagnostic tools” [4] that are used to develop control algorithms and to monitor and change variables in real time. In the combined dSPACE-Matlab package these two tools can be viewed as Simulink and the graphical user interface (GUI) that is provided in dSPACE. In Simulink we develop the controller and all the necessary functions to run the experiment. Once we have the code, what they call the “model,” we compile it and following some steps that are transparent to the user, we obtain a file that will run the code in real time, and provide the ability to set up a user interface. This GUI in dSPACE can be viewed as the RCS diagnostic tool, since we can change some variables, and see in real time some of the variables defined by the user in the model.

**Balls-in-Tubes**

This experiment was designed to be an inexpensive testbed for dynamic resource allocation strategies. Below, we describe the elements of the experiment, its challenges, and implement and study two different dynamic resource allocation approaches.

**Experimental Apparatus and Challenges**

Figure 1 shows the balls-in-tubes experiment. There are four tubes, each of which holds a ball inside, a fan at the bottom to lift the ball, and a sensor at the top to sense the ball’s height. For each tube there is a box that the fan pressurizes. You can think of this box as a stiff balloon that is
“blown up” by the fan, but which has an outlet hole used to blow air into the tube. The tubes are connected at the fan inlets via an input manifold which has an inlet at the bottom as indicated. Also, there is an output manifold at the top of the tubes with an outlet as shown. The presence of the manifolds is a key part of the experiment. These manifolds force the sharing of air at the input, or constrain its flow at the output, so that there is significant coupling between the four tubes. Characteristics of the coupling can be adjusted by, for instance, making both the inlet and outlet have different opening sizes or by placing solid objects in the manifold to obstruct airflow to some fans. For a range of inlet sizes, if one fan succeeds at lifting the ball to near the top, it can only do this at the expense of other balls dropping. This feature leads to the need for “resource allocation” where here the resource is the air that elevates the balls. Flow characteristics in the manifolds are very complicated due to, for instance, air turbulence in the manifolds and pressurized box. Several of their effects will be explained below. Finally, note that the experiment was designed to be easily extended to more tubes, different arrangement patterns, and to have different manifolds and hence interaction effects.

Figure 1: Balls-in-tubes experiment (tubes 1-4, numbered left to right).

To sense the ball height for each tube we use a Devantech SRF04 ultrasonic sensor. The raw
data obtained from the sensor is very noisy. For example, the spikes that we get from it correspond to errors greater than 10 centimeters. We developed a filter to smooth the sensor output. For our sampling rate, after filtering we achieved a resolution of $\pm 1$ centimeter. The actuators that we selected are Dynatron DF1209BB DC fans commonly found inside computers. We use a pulse width modulation (PWM) signal as an input to the fan. The sampling period is 100 $\mu$sec, and the period for the PWM is 0.01 sec. In total, there is one digital input and one digital output for each sensor, and one digital output for each fan for a total of 12 digital input-output lines that we connect to a DS1104 dSPACE card.

There are a number of control objectives and challenges that can be studied for this experiment, beyond the obvious isolated balancing of a single ball in a tube:

1. Balancing the balls inside the tubes, trying to allocate air pressure to keep all the balls at fixed positions or alternatively, a uniform height but maximally elevated.

2. Balancing and reallocation dynamics in the presence of disturbances such as changes in manifold inlet sizes or flow obstructions in a manifold. Also, effects of using decentralized and networked decision making in the presence of an imperfect communication network could be studied.

**Resource Allocation Strategies and Results**

Here we describe two resource allocation strategies and analyze their performance in the presence of two types of disturbances.

**Resource Allocation Strategy: “Juggler”**

The goal here is to try to balance the balls around a certain common height. That is, we allow the peak-to-peak oscillations of each ball height to be relatively large, but the average of the height values of each ball should approach each other. We complicate achieving this goal by allowing only one fan enough input (that is, percentage of PWM duty cycle), to raise its ball at a time; hence we think of this as juggling the balls to keep them in the air. How is this a resource allocation strategy? Basically, it has to pick a sequence of fan inputs that will keep the balls elevated around a common value. This allocation of pulses is dynamic over time.
There are some characteristics of the experiment that we need to highlight in order to explain the design of the juggler strategy:

1. The rising time (RT) of all the balls is greater than the dropping time (DT) over the whole length of the tube.

2. The RT and DT of the balls in the inner tubes (tubes 2 and 3) is less than the RT and DT, respectively, of the outer ones (tubes 1 and 4). This is a clear consequence of the location of the air inlet at the input manifold.

3. The RT of the fourth tube seems to be greater than the RT of the first one.

4. The percentage of the duty cycle of the PWM signals needed to start raising the balls for all the actuators is about 60%.

Let \( D_{iu}(t) \) and \( D_{id}(t) \) be the duty cycles applied to fan \( i \) at time \( t \), which cause the ball to be raised or dropped respectively. Here, \( D_{iu}(t) > D_{id}(t) \), for \( i = 1, 2, 3, 4 \). At time \( t \), the controller will allocate just one \( D_{iu}(t) \) to fan \( i \) while the other three fans will have \( D_{id}(t) \) for a maximum time, \( T \), equal to 3 seconds (that is, one ball raises and the other three drop while these duty cycles are applied to the fans). We chose these values using the above insights and some trial and error tuning. The “juggler strategy” is:

1. Find ball to focus on: Determine the minimum height at time \( t \) of all the balls, that is, \( h_i^*(t) = \min_i \{h_i(t)\}, \ i = 1, \ldots, 4 \), and the index of the minimum height ball \( i^* = \arg \min_i \{h_i(t)\} \).

2. Push low ball up: Assign \( D_{iu}^*(t) \) to fan \( i^* \) and \( D_{id}(j) \), \( j \neq i^* \) to the other fans for a time equal to \( T \):
   - Truncation rule 1: If \( h_i^*(t) > 0.6 \) meters at time \( t', t' \in [t, t + T] \), then the algorithm will abort the current allocation and will go to step 1 even if a time duration \( T \) has not elapsed. This rule will avoid pushing ball \( i^* \) too high for too long.
   - Truncation rule 2: If \( h_i(t) > 0.6 \) meters, for any \( i \) at time \( t \), then the algorithm will abort the current allocation and it assigns \( D_{id}(t) \) to each fan for 0.6 seconds and when this time elapses go to step 1. This rule helps to avoid having all balls get stuck at the top of the tubes.
So, intuitively the juggling strategy involves dynamically applying different size and length “hits” (air pulses) to the fan corresponding to the minimum ball height.

Figure 2 shows the results for the implementation of this juggler strategy. Notice that there are significant oscillations in the ball heights but this is expected since only one fan is given an input to raise its ball at each time. Moreover, it is acceptable for the objectives stated above. A typical control systems goal where we seek zero steady state tracking error is simply not possible for the given constraints. The control goal is nontraditional, but nonetheless quite intuitive as most people have seen a juggler perform and understand that it is a feedback control problem. Next, note that in Figure 2 the mean heights of the balls are reasonably close. We can tune the values of $D_u^*(t)$ and $D_p^i(t)$ to get closer correspondence between the means; however that can be both tedious and futile. The tube differences that arise from sensors, actuators, and physical construction lead to the differences and we have even seen temperature changes in the room affect the experiment. Hence, for this experiment the goal of achieving close, rather than perfectly the same, mean values is appropriate.

Next, we ran another experiment using the same strategy but with a disturbance. Here, for our disturbance we placed an object inside the input manifold partially obstructing the air flow of tube 4. The presence of the disturbance required a reallocation relative to the no-disturbance case above. To study the reallocation we ran both cases and computed the mean ball heights and

Figure 2: Height of each tube for “juggler” strategy. The standard deviation is 0.14, 0.17, 0.20, and 0.15 for tubes 1 through 4 respectively.
percentage of the total time the experiment was run (200 sec.) that each tube/ball was given
attention by pushing it up (that is, the percentage of time $D_u^i(t)$ was applied). Then, we compile
the results in Table 1. As we can see, the tube that was attended the most is tube 4, even in the
no disturbance case due to the characteristics of the experiment. The mean values for the ball
heights in both cases are comparable to the ones for the disturbance case but generally lower due
to the flow obstruction. Next, notice how the amount of time allocated to tube 4 increased in order
to compensate the deficiencies in air flow caused by the presence of the obstacle close to this fan.
Basically, the effect of the reallocation was to take some of the air allocated from tubes 2 and 3
in order to maintain the relatively equal mean heights. This is the essence of dynamic feedback
resource allocation.

Table 1: Allocation results with and without disturbances.

<table>
<thead>
<tr>
<th></th>
<th>Without disturbance</th>
<th>With disturbance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of time each tube is attended (in %)</td>
<td>Height mean(m)</td>
<td>Amount of time each tube is attended (in %)</td>
</tr>
<tr>
<td>Tube 1</td>
<td>23.22</td>
<td>0.62</td>
</tr>
<tr>
<td>Tube 2</td>
<td>10.95</td>
<td>0.72</td>
</tr>
<tr>
<td>Tube 3</td>
<td>22.69</td>
<td>0.70</td>
</tr>
<tr>
<td>Tube 4</td>
<td>43.13</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Resource Allocation Strategy: Dynamic Proportioning

For this strategy we implement a simple proportional controller for ball height that has a static
nonlinear gain on the error $e(t) = h_d(t) - h_i(t)$, where $h_i(t)$ is the ball height for the $i^{th}$ tube and
$h_d(t)$ is the desired height. A PID controller may perform better but such design improvement for
the low-level controller is not our focus. So, this proportional controller changes the duty cycle
of the PWM used to drive the fans. Its gain depends on the sign of the error. If the error $e(t)$
is positive, then the controller will have one slope, and another one if the error is negative. This
strategy ensures that the balls do not drop too fast, so that they are not too difficult to lift.

Every 10 seconds we change $h_d(t)$ by letting

$$h_d(t) = \max_i \{h_i(t)\} \quad i = 1, 2, 3, 4$$
This resource allocation strategy indicates that the fan for the lowest ball should allocate more air, enough to lift its ball to the same height as the highest ball. The time of 10 sec. is used to allow the ball to move to achieve the reallocation. The interesting feature is that the dynamic iterative application of this simple allocation rule generally results in lifting all the balls to a maximum uniform height. In fact, since in implementation it is essentially impossible for the balls to be at exactly the same height, the policy exploits noise and overshoot to persistently drive the balls to higher uniform positions. What emerges is a proportioning in the allocation with relative amounts of air allocated to overcome flow differences between the tubes and disturbances. Hence, the resource allocation strategy proportionally allocates the PWM input at each time instant and hence dynamically over time. This is quite different from the juggler case where the allocation is more constrained so that it can be thought of as an allocation across time only.

Finally, note that for this experiment we have to limit at the beginning the amount of air that is entering at the manifold or all the balls will hit the top. For that, a piece of paper with a circle in the middle was selected. Below, we will illustrate the effect of changing the area of the inlet to the input manifold on dynamic reallocation.

Figure 3 shows the results for this case. Almost all the balls reach the desired height $h_d(t)$ except for the first one. One of the reasons for that is that the air is entering in the middle of the input manifold, and hence close to tubes 2 and 3. We can see that the balls try to converge to a certain desired height around 0.7 m. and so that their heights are within about 0.2 m. of each other after a couple of minutes. However, even though the ball in tube 1 is in that range, it does not reach the maximum desired height. This problem is due to several factors. For instance, we cannot measure the flow that is inside the manifold, but clearly this flow makes the dynamics inside the manifold change continuously due to turbulence. Some effects of air turbulence are easy to see. Bouncing of the balls is seen for any constant PWM signal, and the balls spin during operation. Moreover, even though we tried to overcome the differences between the tubes by tuning the proportional controllers we did not dedicate much time to this since it is not the main focus. For example, adding an integral term to the controller would likely reduce the error seen in Figure 3. This experiment does, however, point to an important idea of the need to re-tune or re-design low-level controllers based on what performance can be achieved for the higher level controller. This is an
important design principle found in hierarchical and decentralized control [4], [5], [6].

Figure 3: Ball height in each tube and set point $h_d(t)$ (“SP” = set point).

Figure 4 illustrates the results of dynamic reallocation when the inlet area is reduced at $t = 20$ sec. The effect is clearly that since there is not as much air to proportion, the balls fall. However, they fall somewhat uniformly due to the dynamic reallocation. For this case we had to add a constraint, that indicates that if all the balls were below 0.2 m, then the desired height will be a fixed value of 0.4 m. Such characteristics highlight the need to understand the theoretical underpinnings of the dynamic resource allocation strategy such as the choice of $h_d(t)$, effects of dynamics, noise on ultimate achievable height, and achievable height uniformity. It may be possible to address some of these issues using the real-time optimization perspective of extremum seeking control [7]. Resource allocation was, however, originally conceived as a static optimization problem [8]. Extending resource allocation concepts to control systems systems requires new innovations in resource allocation theory to cope with dynamics and disturbances via the exploitation of feedback information.

**Electromechanical Arcade**

This experiment was designed to be an inexpensive testbed for networked decentralized scheduling strategies. Below, we describe the apparatus, highlight the challenges the electromechanical arcade presents, and introduce and implement two scheduling strategies.
Figure 4: Ball height in each tube and set point $h_d(t)$ with a disturbance introduced at 20 sec ("SP" = set point).

**Experimental Apparatus and Challenges**

This experiment is composed of two main components, guns and targets, as shown in Figure 5. Each of them is provided with a Radio Shack # 277-1101 laser and a Radio Shack # 276-1657 photodetector. There are in total eight “pop-up” targets that “appear,” indicated by its laser being on, and “disappear” (laser off) at frequencies that are independent of each other. These lasers are driven by simple timing circuits that can be adjusted by hand to provide different frequencies. The guns can detect a target appearance, if it is pointed directly at it, via a photodetector mounted on the gun. The two guns are each mounted on the shafts of Shinano Kenshi # LA052-040E4N02 motors. Each of these motors has a quadrature encoder that provides the angular position of the motor. We use a PID controller to point any gun to the target that needs to be shot and these PID loops are well tuned so their performance will not impact our strategies below. The photodetectors of all targets (8 digital inputs), the laser and photodetector of each gun (2 digital inputs and 2 digital outputs), the motor encoder, the output of each PID controller (1 analog output each) are connected to one DS1104 dSPACE card. Analog outputs are connected to an Advanced Motion Control # BE12A6 amplifier for each motor.

All lasers located at the targets point to a gun photodetector and if one gun is pointing to a target when it appears, this gun can “shoot” at that target, by turning its laser on which triggers the corresponding photodetector of the target. When the photodetector of a target is triggered,
the gun considers that specific target to be “hit,” gets a point for hitting the target, and then will pursue another target. The sequence of firings is specified by a real-time scheduling strategy. The analogy with arcade games should be clear.

We assume that the guns do not know any information about the rate of appearance of all targets, but strategies could be invented for estimating appearance sequences. The guns do know a priori the position of all targets, and the guns can communicate to each other their decisions about the targets that they are currently processing or pursuing. The challenges for this experiment are as follows:

1. To schedule in real-time a sequence of firings so as to maximize the number of points the team gets. Since target detecting and shooting requires movement of the guns, a good schedule will typically minimize the motion of the guns and at the same time maximize point gain. Feedback is required to overcome, for instance, uncertainty about when targets appear or to develop target appearance time estimation strategies. Open-loop precomputed schedules will not be effective.

2. To cooperatively schedule the shooting of the guns in the presence of an imperfect communication network that allows communication between the two guns. While the network could be the internet and a computer could be dedicated to each gun, networked schedulers can also be simulated within one computer. Communication imperfections such as random but bounded delays, bandwidth constraints, or message misordering could be considered.

We consider then an “environment” for the guns that is highly uncertain due to the uncertain target
appearance times. We have imperfect communications that make it difficult for the two guns to coordinate their actions. Due to the presence of so much uncertainty it is generally not possible to accurately predict far into the future, and hence generally not useful to employ optimization approaches to develop long sequences of planned operations either off- or on-line. Finally, note that this decentralized scheduling problem can be thought of as a type of resource allocation strategy analogous to how we thought of the juggler, but clearly the dynamics, uncertainty sources, and performance objectives are quite different. Moreover, the resource here is the time dedicated to a target so that allocation is temporal rather than spatial.

Scheduling Strategies and Results

The targets are numbered and the set of targets is denoted by $P = \{1, 2, \ldots, N\}$. Here $N = 8$ targets. The set of guns is denoted by $Q = \{1, \ldots, M\}$. Here $M = 2$ guns. Let $p_i, i \in P$, denote the priority of processing target $i$. “Processing” means moving to and shooting at the target when it appears. The priority $p_i > 0$ is proportional to the importance of a target. Let $t$ denote time.

Let $T_i(t), i \in P, t \geq 0$ denote the prioritized time at which target $i$ was last shot and $t_i$ denotes the time since last shooting target $i$. For example, if target $i$ was last shot at time zero then $T_i(t) = p_i t_i$ is its “prioritized time.” Suppose that initially $T_i(0) = 0, i \in P$, so that we act as though initially we had simultaneously shot all the targets, which is clearly physically impossible. Note, however, that this is a good initialization considering the fact that below our scheduling strategies will make decisions about which target to process next based on the sizes of the $T_i(t), i \in P$.

Note that if no gun was used, then clearly $T_i(t) \to \infty, i \in P, t \to \infty$ since it will never shoot a target. The goal of the scheduling strategy is to try to avoid $T_i(t) \to \infty$ for any $i \in P$ and indeed it will try to keep the $T_i(t)$ values as small as possible since this represents that the guns have recently shot each target.

There are three types of delays in the experiment. There is the delay between the appearances of each target. There is the “travel time” to move the gun so that it points at a target and this can be thought of as a type of delay. Finally, there is the communication delay that is random in duration but bounded. We implement an asynchronous scheduler over a simulated communication network in order to pick $i^*_j(t)$, the target that gun $j$ should process at time $t$. To define the scheduler, let $U(t) \subset P$ be the set of unattended targets and $U^0(t) = \{i^*_j(t)\} \cup U(t)$ be the set
of targets that gun \( j \) could consider if it just finished processing (shooting at) target \( i_j^*(t) \). We pass the set \( U(t) \) around the network and let each gun choose a target from \( U_j^a(t) \) and put the one it just finished processing into \( U(t) \). A distributed mutual exclusion algorithm is used and the communication delay is incurred each time \( U(t) \) is passed. While a gun is waiting to receive \( U(t) \) it can keep shooting at \( i_j^*(t) \). Hence, we consider \( i_j^*(t) \) to be attended until gun \( j \) makes the decision and switches to processing a different target.

**Networked Decentralized Scheduling: Focus on Target Ignored the Longest**

Next, we introduce a scheduling strategy from [9], which is an extension of one defined in [10], and for which stability properties have been investigated in [11], [12]. First, let \( t_j \) denote the time at which the scheduling strategy chooses a target to process (that is, it is the decision time), and suppose that \( t_j = 0, \ j \in Q \). A scheduling strategy that processes the targets that were ignored more than the average one makes choices of which targets to process such that at \( t_j \) the scheduling strategy chooses to process target \( i_j^*(t_j) \) where

\[
    T_{i_j^*(t_j)}(t_j) \geq \frac{1}{N-M+1} \sum_{i_j \in U_j^a(t_j)} T_{i_j}(t_j), \ i_j \in U_j^a(t_j)
\]

and makes no other decision until it has finished shooting target \( i_j^* \) and received \( U(t) \). A special case of this strategy is the one that picks the \( i_j^*(t_j) \) such that \( T_{i_j^*(t_j)}(t_j) \) is bigger than any other \( T_{i_j}(t_j) \); this is the “process-the-target-ignored-the-longest” strategy that we will use. If there is more than one maximizer for any gun at any time, then the strategy will simply choose one of these at random.

Figure 6 shows the results when we implement the process-the-target-ignored-the-longest strategy and we use a fixed communication delay of 10 seconds (we used a fixed delay to facilitate comparison to an approach below). Figure 6 shows the \( T_i(t), \ i \in P \), values as well as the targets selected by the two guns during the time the experiment is running. Notice that before the experiment starts both guns are pointing at target 1. Once the experiment starts, gun 1 chooses a new target to process next, while gun 2 has to wait 10 seconds for the arrival of the set \( U(t) \) coming from gun 1. During this 10 seconds, once gun 2 detects target 1, it keeps processing the same target and this is the reason why \( T_1 = 0 \) in this interval. We can draw some important conclusions from
the experimental data. First, notice the effect of the priority values, $p_i$, on the variable $T_i$. We assigned the following priorities to the targets, $p_1 = 0.9$, $p_2 = 0.85$, $p_3 = 0.8$, $p_4 = 0.75$, $p_5 = 0.4$, $p_6 = 0.4$, $p_7 = 0.5$, and $p_8 = 0.2$, where the enumeration is from left to right viewed from the place where the guns are. These values were chosen by observing the illumination frequency of all targets and assigning higher priorities to the targets that appear more frequently. Second, we can see how a communication delay can affect the performance of the strategy. In particular, observe how the $T_i$ values contained in the set $U(t)$ are ignored during this time. Third, we can see how the guns allocate their processing capabilities in order to shoot and gain more points for as many targets as possible in a cooperative manner. To see this, study the sequence of choices.

Figure 6: Performance of the ignored the longest strategy. In the right bottom plot the solid line corresponds to $i^*_1(t)$, and the dashed line corresponds to $i^*_2(t)$.

**Networked Decentralized Scheduling: Focus on Closest Highest Priority Target**

Next, we introduce a strategy that seeks to schedule the next target that a gun should process to avoid ignoring high priority targets for too long and yet to minimize travel time to process targets. We view the electromechanical arcade experiment as a type of cooperative scheduling of targets for autonomous vehicles. For this, it is important to pay attention to minimize the travel time between
targets and this motivated us to define the strategy [12] that the gun processes target $i_j^*$ if

$$T_{i_j^*}(t_j) - \delta_{ji}(t_j) \geq \frac{1}{N - M + 1} \sum_{i_j \in U_j} \left[ T_{i_j}(t_j) - \delta_{ji}(t_j) \right]$$

(1)

where $t_j$ denotes the decision time for gun $j$ and $\delta_{ji}$ is the amount of time it takes for the gun to move from target $j$ to target $i_j$. Choice of the maximum value on the left hand side of Equation (1) results in a “process-the-closest-highest-priority-target” policy. Ties are broken with an arbitrary choice. Notice that the left side of Equation (1) could be viewed as a cost function and the scheduler picks the target to pursue to minimize the time the targets are ignored and simultaneously minimize the travel time. A bound on the ultimate longest time that any gun will ignore target $i$ is found in [12].

Figure 7 shows the performance when the process-the-closest-highest-priority-target strategy is used. As in the previous case, the communication delay is fixed and equal to 10 seconds and the priority values are also the same. Notice that the number of points in this case is at least equal for each $T_i$ to the number of points obtained in Figure 6. In addition to this, we can see in Figure 7 that the value of the peak for each $T_i$ is less than the peaks seen in Figure 6. By trying to minimize travel time we are improving our score. On the other hand, we can see in Figure 7 that the closest targets chosen by the guns occur more frequently than those shown in Figure 6. In fact, compare the sequence of targets chosen by the guns when $t \in [0, 25]$ in both figures and this will be easily noticed.

Next, we further quantify the facts highlighted in the above paragraph. In order to do that, we need to introduce a way to evaluate the performance of the guns when they use the two strategies. There are several ways to measure performance of these strategies. Here, we will compute the average of the length of time since any target has been shot $\frac{1}{N} \sum_{i=1}^{N} T_i(k)$ at each step $k$. We will also compute the time average of this quantity (that is, the time average of the average values) and the maximum average value achieved over the entire run of 100 sec. We will compute the maximum time that any target has been ignored at each time step $k$, $\max_i \{ T_i(k) \}$. We will also compute the time average of this quantity (that is, the time average of the maximum values) and the maximum of the maximum values achieved over the entire run. One final performance measure
Figure 7: Performance of the closest highest priority targets strategy. In the right bottom plot the solid line corresponds to $i_1^*(t)$, and the dashed line corresponds to $i_2^*(t)$.

Table 2: Performance measures for arcade.

<table>
<thead>
<tr>
<th></th>
<th>Ignored the Longest</th>
<th>Closest Highest Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average of average</td>
<td>7.28</td>
<td>5.22</td>
</tr>
<tr>
<td>Maximum of average</td>
<td>11.15</td>
<td>8.69</td>
</tr>
<tr>
<td>Average of maximum</td>
<td>14.46</td>
<td>11.56</td>
</tr>
<tr>
<td>Max of max</td>
<td>23.87</td>
<td>20.21</td>
</tr>
<tr>
<td>Number of points</td>
<td>25</td>
<td>35</td>
</tr>
</tbody>
</table>

is the number of points obtained by the two guns (number of targets shot) during the experiment. Table 2 shows the results obtained from the experiments for the two cases above. The process-the-closest-highest-priority-targets strategy performs better for all performance measures, including the total number of point obtained. This shows that seeking to minimize travel time allows significant point improvement. The strategy did not, however, partition the targets so that each gun always was responsible for the adjacent targets. What emerges in the bottom right plot in Figure 7 is a type of balancing of the objectives quantified in Equation (1).

Finally, we would like to highlight a number of other methods that could be tested on the electromechanical arcade. In addition to extensions of the strategies in [10] there are a multitude of
manufacturing system scheduling methods, both deterministic and stochastic, that may be useful for this experiment [13], [14]. Moreover, it would be interesting to add a manual control and develop schedulers that could challenge the skills of a typical teenager at this simple video game.

**Multizone Temperature Control**

Temperature control experiments provide some of the least expensive yet interesting decentralized control experiments we can build. Here, we briefly describe two multizone temperature control experiments: a 2-d planar temperature grid, and a 3-d building temperature control problem.

**Experimental Apparatus and Challenges**

Our first multizone temperature control experiment has 16 individually controlled zones arranged in a regular planar grid as shown in Figure 8. Each of the zones has a light for a heater and an analog temperature sensor. These elements are connected to two DS1104 dSPACE cards placed in two different computers. Since each of the DS1104 dSPACE cards can support only 8 analog inputs, we had to use two different computers to acquire the data. The information between the two computers is transmitted from one computer to the other via an RS-232 link. One computer, called the “master,” will acquire the data from the other one, and it will be in charge of the control strategies. The other one, called the “slave,” will send and receive the data from the “master.”

![Light (heater) Sensor](image)

**Figure 8: Planar temperature experiment.**

Disturbances play an important role. Ambient temperature and wind currents have a significant impact. But since the physical layout of the experiment has the lamps very close to each other’s sensors, we also have disturbances associated with the heat of other zones. Another problem that
we face in this experiment is the calibration. We tried to pick 16 LM35CAZ sensors from National
Semiconductors that sensed practically the same temperature. Unfortunately, once we soldered
the sensors to the board, it is possible that we changed some of the characteristics that we cannot
measure. Even though calibration after construction of the experiment is a possibility, we decided
not to do it because it is extremely tedious, time consuming, and error prone. Hence, we decided
to use the original 16 sensors that we picked, and to help with the interpretation of our results, we
used an external temperature probe properly calibrated to provide the ambient temperature at the
start of the experiment (the probe is a Fluke meter 179).

The main challenges of this experiment are:

1. Try to regulate temperature to be uniform across the grid but with some fixed (maximum)
   value. Alternatively, we could seek some fixed or dynamic 2-d pattern of temperature values.

2. Try to make one set of zones track the average temperature in another zone, or try to track
   a temperature value.

3. Decentralized control with different controllers for each zone and a communication network
   with random but bounded delays for neighbor-zone sensed information, and delayed control
   inputs.

These challenges are difficult to meet due to many kind of disturbances such as ambient temperature
and wind currents, inter-zone effects, and self-induced wind currents. Moreover, communication
network imperfections present challenges in developing distributed controllers.

**Distributed Control Strategies and Results**

To illustrate some of the control strategies that were developed for this multizone experiment,
we will discuss briefly a temperature tracking approach and then show results for centralized and
decentralized resource allocation strategies. More details on this experiment and its behavior for
many other decentralized and networked strategies can be found in [15]. For instance, there we
implemented a “superzone” temperature tracking system. For this case, we group the 16 zones into
4 “superzones.” The idea for this case is that each superzone will track a desired temperature $T_{i}^{ds}$,
i = 1, 2, 3, 4, fixed by the user. Another challenge is to require that superzones 3 and 4 track the
current temperature of superzones 2 and 1, respectively, when the user sets the desired temperature for those.

To give the reader some insight into the dynamics and challenges of this experiment we will first implement a “centralized resource allocation strategy.” For this approach, we have 16 zones with temperatures $T_i$, and inputs $u_i$, $i = 1, 2, \ldots, 16$. Our strategy simply applies at each time instant a 100 ms duration pulse to the zone that has the lowest temperature. We are trying to allocate a “resource” (the time that the light is on, current available for heating, and hence the current to the heaters). The objective is to try to reach the maximum temperature with uniformity across the temperatures of all the zones, given the constraint that only one light can be on at a time. In other words, at each time $k$, for each $i$

$$u_i(k) = \begin{cases} 
1 & \text{if } T_i(k) \leq T_j(k) \quad j \in \{1, 2, \ldots, 16\} \\
0 & \text{otherwise}
\end{cases} \quad (2)$$

Intuitively, this strategy seeks to iteratively increase the minimum temperature, thereby forcing uniformity and at the same time maximizing all the temperatures. Resource allocation “emerges” from a simple “local in time” optimization implemented by Equation (2). Notice the conceptual similarity to the juggler strategy for the balls-in-tubes experiment.

For this strategy the whole grid reaches a practically uniform final value as shown in Figure 9. For this case, we started with a room temperature of 22.8 degrees Celsius, and the average final value that we obtain is around 24.1 degrees Celsius. Some of the zones are above this value (the maximum one was 24.8 degrees Celsius), but considering the disturbances associated with the experiment the performance is quite good. Since the room temperature is variable, each time that we run an experiment the final steady state uniform value achieved is slightly different.

Another strategy that we implemented is a decentralized version of the previous experiment. For this case, we assume that there is a “local” controller for each zone that can make decisions about whether to turn its heater on or off only based on the temperature in its own zone and possibly those in some “adjacent” zones, but only by obtaining such sensed temperatures from “neighbors” over a communication network that may have delays. We let $N(i)$ denote the “neighbors” of zone $i$ that the controller for zone $i$ can sense temperatures in, including zone $i$. Let for all $i$, at each
Figure 9: Temperature behavior for a centralized resource allocation experiment. This experiment was done with a 22.8 degrees Celsius temperature in the room.

\[ T_i^{\text{min}}(k) = \min \{ T_j(k) \mid j \in N(i) \} \]

and

\[ u_i(k) = \begin{cases} 
1 & T_i(k) = T_i^{\text{min}}(k) \\
0 & \text{otherwise}
\end{cases} \]

Intuitively, this is a local version of the one in Equation (2). More than one zone can have its heater on at a time and hence the strategy can simultaneously increase multiple temperature values. If we run the experiment with an ambient temperature of 21.8 degrees Celsius, we obtain the results that are shown in Figure 10. After 200 seconds, the temperature in some of the zones is between 26 and 27 degrees Celsius. There are other zones that are on the edges that cannot reach this value because of the large disturbances that we have discussed before. On the other hand, there are a couple of zones in the middle of the grid that have a large value of temperature (for example, \( T_6 \)) because the lamps that are in the edges are on practically all the time. Moreover, note that relative to the centralized resource allocation case (see Figure 9), there are more oscillations on the
steady state temperatures (note that different vertical scales were used in the two cases in order to facilitate viewing the data); this is due to the effects of the limited amount of information used by each local controller.

Figure 10: Temperature in all the zones when we use the topology defined for a decentralized experiment, with a temperature in the room of 21.8 degrees Celsius.

**Building Temperature Control**

We also have a 3-d version of the experiment, shown in Figure 11, where we study multizone temperature control for a model building. The building temperature control experiment is a two floor model with four rooms in each floor. Each of the rooms has a temperature sensor and a heater that simulates diverse temperature control challenges. The two floors are separable, so that the student can study each of them separately, or he/she can study the interactions between each floor. Each floor has windows and doors that are adjustable so that there exist some interactions between the temperatures in the rooms. Moreover, there are several electronically controllable fans that can be inserted into windows or doors to simulate wind effects and drafts.

The main challenge of this experiment is multivariable temperature regulation in multiple zones. This is very difficult to achieve since we can have different kinds of interactions between zones (proximity of rooms, doors, windows). Also, the disturbances, such as the ambient temperature,
wind through windows, open doors, or even the fans, make the experiment more complex and difficult to control. For more information, see the web site in the footnote of the title.

Finally, we want to highlight some other strategies that may be useful for the temperature control problems. First, decentralized temperature control is used in the semiconductor processing area and some of the methods there would be useful in the above experiments. For instance, achieving a uniform temperature on a plate is studied in [16]. Decentralized control of wafer temperature for multizone rapid thermal processing systems is studied in [17]. In [18] the authors describe a multizone space heating system that maintains a desired temperature in different zones.

Another approach for distributed control design that could be useful for the above experiments is the one described in [19]. Here, the authors show how systems with a complex interconnection topology can be controlled using conditions that can be expressed as linear matrix inequalities.

Concluding Remarks

We have introduced several inexpensive experiments appropriate for university laboratories that can be used to study the design and implementation of decentralized and networked dynamic resource allocation, scheduling, and control strategies. In each case we overviewed the experimental apparatus, the challenges it presents, and for three of the experiments showed results of implementing control strategies in order to provide insights into their operation.

Due to their modular nature, each of the experiments is designed to be easily expandable and this is a future direction for our work. Adding more tubes with a reconfigurable input mani-
fold would be interesting for the balls-in-tubes experiment. With a planar grid the experiment’s
operation would be more visually appealing since it would present a “sheet” of balls that could
dynamically deform in three dimensions. The electromechanical arcade could be expanded in many
ways (for example, to multiple dimensions with two opposing teams) and in an expanded form
may be a very appealing for current students who typically have significant experience with video
games. The multizone temperature control planar grid was chosen to be of minimum size, but still
interesting since it has edge effects and a “middle.” It is of interest to make it larger or in a different
shape. For all the experiments, if they are expanded significantly the data acquisition problems are
amplified and to overcome these it is likely that an embedded systems approach would be needed
with many inexpensive microprocessors and a CAN bus for communications.

Acknowledgements: A number of other people contributed to the construction of the experi-
ments that we describe here and we gratefully acknowledge their assistance. First, we would like to
thank William Thalgott and Emmet Evans for building the balls-in-tubes experiment, and helping
with some details in the electromechanical arcade and the multizone temperature control exper-
iment construction. The balls-in-tubes and the arcade were originally constructed by groups of
undergraduates in an EE682P design project. The arcade was later improved by Kristina Guspan
and Dawn M. Kin. The building temperature control experiment was originally constructed by
a group of undergraduates in an EE682P design project, it was later significantly improved by
Todd Broceus and Tyson Rathburn, and later still Jorge Finke added electrically controllable fans,
interface cables, and implemented a controller in dSPACE.

References


