

# Scheduling and Feedback Co-Design for Networked Control Systems

Michael S. Branicky,<sup>1</sup> Stephen M. Phillips,<sup>1</sup> and Wei Zhang<sup>2</sup>  
{msb11, smp2}@po.cwru.edu, wzhang@brocade.com

## Abstract

Feedback control systems wherein the control loops are closed through a real-time network are called networked control systems (NCSs). The insertion of the communication network in the feedback control loop makes the analysis and design of an NCS complex. Driving our research effort into NCSs is the point of view that the design of both the communication protocols and the interacting controlled system should not be treated as separate. In the co-design approach we propose, network issues such as bandwidth, quantization, survivability, reliability and message delay will be considered simultaneously with controlled system issues such as stability, performance, fault tolerance and adaptability. Thus, we study network scheduling when a set of NCSs are connected to the network and arbitrating for network bandwidth. We first define the basic concepts of network scheduling in NCSs. Then, we apply the rate monotonic scheduling algorithm to schedule a set of NCSs. We also formulate the optimal scheduling problem under both rate-monotonic-schedulability constraints and NCS-stability constraints, and give an example of how such optimization is carried out. Next, the assumptions of ideal transmission are relaxed: we study the above network scheduling problem with network-induced delay, packet dropouts, and multiple-packet transmissions taken into account.

## 1 Introduction

A networked control system (NCS) involves communication patterns in which both informational and physical control loops are closed through a real-time network (e.g. [4, 15, 13, 18, 20]; see [19] for a thorough review). In NCSs, the performance of the control loops not only depends on the design of the control algorithms but also on the scheduling of the shared network resource. Roughly speaking, the problem of network scheduling in NCSs is to assign a transmission schedule to each transmission entity (sensor, controller, actuator) on the

network based on a *scheduling algorithm* (a set of rules that, at any time, determines the order in which messages are transmitted).

In the past, control system design and CPU scheduling or network scheduling design have normally been separated. This separation has allowed the control community to focus on its own problem domain without worrying about how scheduling is being done; it has released the scheduling community from the need to understand what impact scheduling has on the stability and performance of the plant under control. However, when the two designs are combined together, many assumptions are not true. These impact system performance.

While co-design of control and CPU scheduling has recently received a considerable amount of attention [2, 7, 17, 6, 1], network scheduling in NCSs is a relatively blank area that needs to be explored.

### 1.1 Problem Description

Consider a set of NCSs which are connected to a network, as illustrated in Figure 1 (which is simplified from the general case: linear plant and control and no actuator-plant communication). Each plant transmits its sensors' data at transmission period  $h_i(t)$ . One can compute the transmission period bounds in order for each individual plant to be stable [19, 20]. However, when the transmission path is shared with other NCSs, transmission scheduling among the plants has to be performed.

A set of NCS transmissions is said to be *schedulable* by a scheduling algorithm (or the transmission schedule is *feasible*) if all transmissions can be completed before their deadlines.

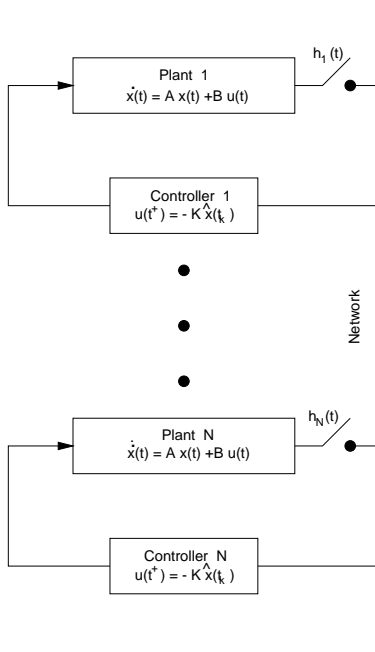
### 1.2 Rate Monotonic Scheduling

In [11], Liu and Layland addressed the problem of prioritized, preemptive scheduling for a set of *independent*, periodic real-time tasks. They proposed the *rate monotonic* (RM) scheduling algorithm, where tasks with shorter periods have higher priorities. RM is a fixed-priority assignment: priorities are assigned to tasks before execution and do not change over time. Moreover, RM can be preemptive: the currently executing task is preempted by a newly-arrived task with shorter period.

---

<sup>1</sup>Electrical Engineering and Computer Science Department, Case Western Reserve Univ., Cleveland, OH 44106-7071 U.S.A.

<sup>2</sup>Brocade Communications Systems, Inc., San Jose, CA 95110



**Figure 1:** Scheduling of a set of NCSs

Liu and Layland showed that RM is optimal among all fixed-priority assignments in the sense that no other fixed-priority algorithm can schedule a task set that cannot be scheduled by RM. They further derived a worst case utilization bound (sufficient condition):

**Theorem 1 (Theorem 6.11, [12])** *A set of  $N$  independent, preemptive, periodic tasks can be feasibly scheduled by the RM algorithm if its network utilization factor,  $U$ , satisfies  $U = \sum_{i=2}^N \frac{c_i}{h_i} \leq N(2^{1/N} - 1)$ , where  $h_i$  is the task period and  $c_i$  is the computation time, each for the  $i$ th process.*

The utilization bound decreases monotonically from 0.83 when  $N = 2$  to  $\log_e 2 = 0.693$  as  $N \rightarrow \infty$ . Thus, any periodic task set of any size will be able to meet all deadlines all of the time if the RM algorithm is used and the total utilization is less than 0.693. A necessary and sufficient condition on the schedulability of a set of tasks scheduled by RM algorithm is given in [10].

RM scheduling of a set of non-preemptive, periodic tasks is discussed in [12, 16]. A summary result is

**Theorem 2 (Theorem 16, [16])** *A set of  $N$  independent, non-preemptive, periodic tasks (indexed by the decreasing order of their priorities with  $i = 1$  being the highest and  $i = N$  being the lowest) are schedulable if for all  $i = 1, \dots, N$*

$$\frac{c_1}{h_1} + \frac{c_2}{h_2} + \dots + \frac{c_i}{h_i} + \frac{\bar{b}_{l,i}}{h_i} \leq i(2^{1/i} - 1), \quad (1)$$

where  $\bar{b}_{l,i}$  is task  $i$ 's worst-case blocking time by the lower priority tasks, i.e.,

$$\bar{b}_{l,i} = \max_{j=i+1, \dots, N} c_j.$$

Note that Theorem 2 is a sufficient condition.

### 1.3 Application to NCSs

The foregoing is directly applicable to scheduling for an NCS by identifying (1) the task period,  $h_i$ , with a (bound on the) worst-case transmission period guaranteeing stability and (2) the computation time,  $c_i$ , with the transmission time, each for plant  $i$ .

When a set of NCS plants are connected to the network and arbitrate for network bandwidth, based on the RM scheduling algorithm, a “faster” plant (i.e., requiring higher transmission rate) is given higher priority over a slower plant. The RM scheduling algorithm can be implemented on priority-based networks, such as CAN and DeviceNet, where the priority of the message can be incorporated into the message identifier.

Theorem 2 can be used to test the schedulability of a set of NCSs connected to the same network when scheduled by the RM algorithm. The following example illustrates how this can be done.

**Example 3 (RM Schedulability Test)** Consider a set of scalar plants represented by  $\dot{x} = ax$  with  $a = 25, 20, 5$  and  $k = 50, 45, 30$ , respectively. Note that all three NCSs have the same closed-loop performance,  $\bar{a} = -25$ . The upper bounds on these plants' transmission periods that preserves their stability,  $h_{\text{true},i}$ , are easily calculated and given in Table 1.

NCS $i$	1	2	3
$a$	25	20	5
$h_{\text{true}}$	0.0439 s	0.0478 s	0.0673 s

**Table 1:** Transmission period bounds for Example 3

Based on the above results and the RM algorithm, the set of NCSs with  $a = 25, 20, 5$  are assigned priorities 1, 2, 3 and indexed by  $i = 1, 2, 3$ , respectively. Let the transmission periods of the NCSs be  $h_1 = 0.026$  s,  $h_2 = 0.030$  s, and  $h_3 = 0.034$  s. Assume every NCS has the same transmission time  $c_1 = c_2 = c_3 = 0.004$  s (this value is chosen based on DeviceNet specification [14]). Therefore, the worst-case blocking time for the NCSs (by lower priority NCSs) are  $\bar{b}_{l,1} = \bar{b}_{l,2} = 0.004$  s. The schedulability analysis using Theorem 2 is as follows:

$$\frac{c_1}{h_1} + \frac{\bar{b}_{l,1}}{h_1} = 0.3077 \leq 1(2^1 - 1) = 1,$$

$$\frac{c_1}{h_1} + \frac{c_2}{h_2} + \frac{\bar{b}_{l,2}}{h_2} = 0.4205 \leq 2(2^{1/2} - 1) = 0.8284,$$

$$\frac{c_1}{h_1} + \frac{c_2}{h_2} + \frac{c_3}{h_3} = 0.4048 \leq 3(2^{1/3} - 1) = 0.7798.$$

We can see that the set of NCSs with the selected transmission periods is schedulable using RM scheduling.

As defined above, a transmission blocking time  $b_i$  consists of two parts: the time to wait for all higher-priority transmissions to finish and the time to wait for an on-going lower-priority transmission to finish. Considering this, the range of the transmission period of each NCS is affected by its worst-case blocking time  $\bar{b}$  (preemption by higher-priority NCSs,  $\bar{b}_{h,i}$ , plus the blocking time by lower-priority NCSs,  $\bar{b}_{l,i}$ ), which are summarized in Table 2. We see that the range of the  $h$ 's

NCS $i$	$b_i = b_{h,i} + b_{l,i}$	$h_i \in$
1	$0 + 0.004 = 0.004$ s	$[0.022, 0.030]$
2	$0.004 + 0.004 = 0.008$ s	$[0.022, 0.038]$
3	$0.008 + 0 = 0.008$ s	$[0.026, 0.042]$

**Table 2:** Worst-case blocking and transmission period ranges

do not exceed their respective upper bounds on transmission periods. Therefore, we conclude that the set of NCSs scheduled by the RM algorithm is schedulable and each NCS is stable.

#### 1.4 Scheduling Optimization

A natural extension to the schedulability problem is, How can we select an optimal feasible schedule which minimizes (maximizes) some performance measure?

Assume each NCS is associated with a performance measure function,  $J_i(h_i)$ , which gives the control cost as a function of transmission period  $h_i$ . Therefore, we can formulate the following optimization problem:

$$\text{minimize (maximize)} \quad J(h_i) = \sum_{i=1}^N J_i(h_i) \quad (2)$$

subject to

- RM schedulability constraints ( $i = 1, \dots, N$ ):

$$h_1 \leq \dots \leq h_N, \quad (3)$$

$$\frac{c_1}{h_1} + \dots + \frac{c_i}{h_i} + \frac{\bar{b}_{l,i}}{h_i} \leq i(2^{1/i} - 1), \quad (4)$$

- NCS stability constraints:

$$h_i \leq h_{\text{suff},i} - \bar{b}_i, \quad i = 1, \dots, N. \quad (5)$$

The worst-case blocking time of each NCS transmission,  $\bar{b}_i$ , needs to be taken into account when setting the upper bound on  $h_i$ .

The selection of the performance measure function  $J_i(h_i)$  is crucial in the optimization problem. In the NCS scheduling optimization problem, we normally choose quadratic cost or exponential cost [3].

**Example 4** Consider the same setup as given in Example 3. The transmission error  $e(t)$  is given by

$$\dot{e}(t) = ae(t) + \bar{a}x(t_k),$$

and the relative error between transmissions is

$$\left| \frac{e(t)}{x(t_k)} \right| = \frac{-\bar{a}}{a} [e^{at} - 1].$$

We can define the performance measure for NCS  $i$  as

$$J_i(h_i) = \frac{-\bar{a}}{a_i} e^{a_i h_i},$$

which is a convex and monotonically increasing function. We can formulate the minimization problem as in (2), subject to the constraints (3), (4), (5) and using  $h_{\text{true},i}$  for  $h_{\text{suff},i}$  in (5). The minimization process may then be carried out using MATLAB function `fmincon`. The optimal transmission periods found were

$$h_1 = 0.0146 \text{ s}, \quad h_2 = 0.0150 \text{ s}, \quad h_3 = 0.0167 \text{ s}.$$

The last RM constraint is tight. Using these transmission periods, the total transmission errors of the three NCSs is minimized when they are scheduled by the RM scheduling algorithm.

## 2 Non-Ideal Transmissions

### 2.1 Effects of Delay

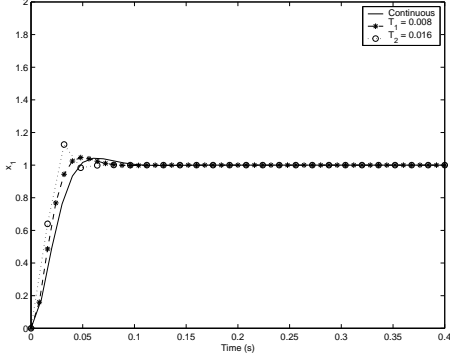
The following example illustrates the influence of the network-induced delay on the performance of NCSs.

**Example 5 (Non-Ideal Transmissions)** Let the continuous open-loop plant be

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \quad (6)$$

A continuous-state feedback controller is  $u = -Kx$ , where  $K = [500, 100]$  (closed-loop poles at  $-50 \pm 50j$ ) and full-state information is transmitted across a network. Now consider two versions of the plant with different constant transmission periods:  $h_1 = 0.008$  s and  $h_2 = 0.016$  s. (The two NCSs are *simply periodic*.) The scaled step responses are given in Figure 2. We can see they are similar to the continuous system response. We further assume the same constant transmission time  $c_1 = c_2 = 0.003$  s.

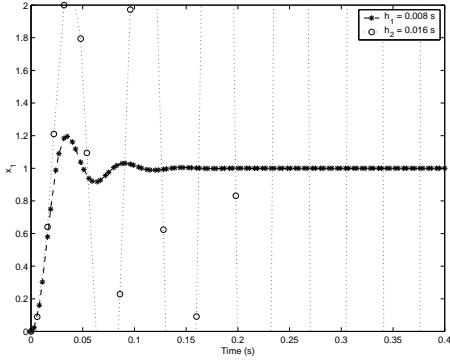
Now, consider the RM scheduling of the two NCSs on a network. Plant 1 receives higher priority, since it has



**Figure 2:** Scaled step responses of original system

a higher transmission rate. Assume the two NCSs are *in-phase*. Then NCS 1's data is delayed by its own transmission time,  $\tau_1 = 0.003$  s, while that of NCS 2 is delayed by its own transmission time plus preemption by the higher priority NCS 1,  $\tau_2 = 0.006$  s.

Figure 3 depicts the scaled step response when RM scheduling is employed and delays are taken into account. With NCS 1's response similar to the original one, NCS 2's response is unstable, caused by the excessive delays during the transmissions.



**Figure 3:** Scaled step responses by RM scheduling

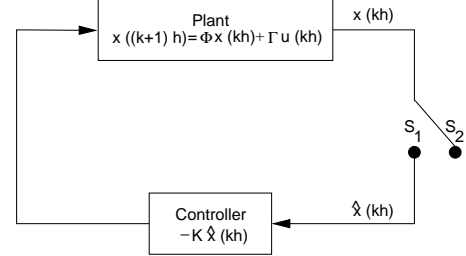
## 2.2 NCSs with Data Packet Dropout

Networks can be viewed as unreliable data transmission paths, where congestion and node failure can occasionally occur. When these happen, depending on the network protocol employed, packets can be lost and are not recoverable. For example, the UDP/IP socket does not guarantee the delivery of data packets. Thus, it is valuable to analyze the data packet dropout rate (percentage) at which the desired performance (e.g. stability) of an NCS can be preserved.

An NCS with data packet dropout can be modeled as an Asynchronous Dynamical System (ADS) with rate

constraints on events. The stability of this type of system is studied in [9]. In this section, we will extend a result therein to NCSs. We still assume the NCS has constant sampling period  $h$ .

Figure 4 illustrates an NCS setup with the possibility of dropping data packets. For constant sampling period  $h$ , we use the sampled version of our linear model.



**Figure 4:** NCS with data packet dropout.

The network can be modeled as a switch that closes at a certain rate  $r$ . When the switch is closed (position  $S_1$ ), the network packet containing  $x(kh)$  is transmitted, whereas when it is open (position  $S_2$ ), the output of the switch is held at the previous value and the packet is lost. Thus the dynamics of the switch (state  $\hat{x}$ ) can be modeled as an ADS as follows:

$$\begin{aligned} S_1 : \quad \hat{x}(kh) &= x(kh), \\ S_2 : \quad \hat{x}(kh) &= \hat{x}((k-1)h). \end{aligned}$$

Define the transmission indicator function  $s(kh)$  as

$$s(kh) = \begin{cases} 1, & \text{sample is transmitted,} \\ 2, & \text{sample is not transmitted.} \end{cases}$$

Let  $w(kh) = [x^T(kh), \hat{x}^T(kh)]^T$  be the augmented state vector; the closed-loop system with the network packet dropout effect is represented by

$$w((k+1)h) = \tilde{\Phi}_{s(kh)} w(kh). \quad (7)$$

When the switch is in position  $S_1$ ,  $s(kh) = 1$  and

$$\tilde{\Phi}_1 = \begin{bmatrix} \Phi & -\Gamma K \\ \Phi & -\Gamma K \end{bmatrix};$$

when the switch is in position  $S_2$ ,  $s(kh) = 2$  and

$$\tilde{\Phi}_2 = \begin{bmatrix} \Phi & -\Gamma K \\ 0 & I \end{bmatrix}.$$

Normally, an NCS can tolerate a certain amount of feedback data loss. The following corollary can be used to test system stability for a certain packet dropout rate.

**Corollary 6 (NCS with Packet Dropout, [19])**

For the above setup, assume the plant state  $x(kh)$  is transmitted at the rate of  $r$ . If there exist a Lyapunov function  $V(w(kh)) = w^T(kh)Pw(kh)$  and scalars  $\alpha_1$  and  $\alpha_2$  for Equation (7) such that

$$\begin{aligned} \alpha_1^r \alpha_2^{1-r} &> 1, \\ \tilde{\Phi}_1^T P \tilde{\Phi}_1 &\leq \alpha_1^{-2} P, \\ \tilde{\Phi}_2^T P \tilde{\Phi}_2 &\leq \alpha_2^{-2} P, \end{aligned}$$

the system is still exponentially stable.

With the plant state transmitted at rate  $r$ , the effective sampling period is  $h_{\text{eff}} = h/r$ . This suggests the plant can be stabilized by a slower sampling rate.

**Example 7 (NCS with Packet Dropout)**

Consider the state-space plant model

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u, \\ y &= [1 \ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \end{aligned} \quad (8)$$

A continuous-state feedback controller is  $u = -Kx$ , where  $K = [3.75, 11.5]$  (closed-loop poles at  $-1/2, -3/4$ ).

With sampling period  $h = 0.3$  s, we obtain

$$\Phi = \begin{bmatrix} 1.0 & 0.2955 \\ 0 & 0.9704 \end{bmatrix}, \quad \Gamma K = \begin{bmatrix} 0.0167 & 0.0512 \\ 0.1108 & 0.3399 \end{bmatrix},$$

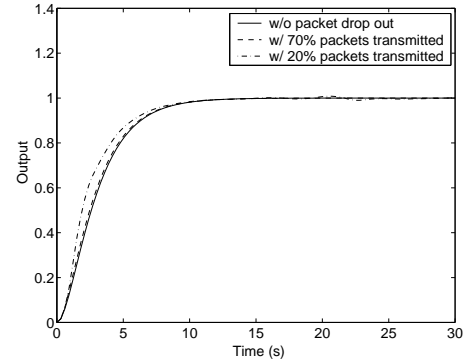
and the closed-loop system  $(\Phi - \Gamma K)$  is still stable with the continuous controller. With the setup shown in Figure 4, and assuming the transmission rate  $r = 0.7$ , we solve the LMI problem [8] in Corollary 6 to find

$$\alpha_1 = 1.1288, \quad \alpha_2 = 0.7552,$$

$$P = \begin{bmatrix} 0.9210 & 0.9196 & -0.6578 & -0.5144 \\ 0.9196 & 2.4788 & -0.5232 & -1.6644 \\ -0.6578 & -0.5232 & 0.7003 & 0.6461 \\ -0.5144 & -1.6644 & 0.6461 & 2.0562 \end{bmatrix},$$

which proves the stability of the system. This means that when the plant state is sampled every 0.3 s, if 70% of the packets are delivered to the controller, we can still guarantee the stability of the feedback control system. The result shows an effective sampling period of  $h_{\text{eff}} = h/r = 0.43$  s. In fact, the sufficient condition on transmission period is  $h_{\text{suff}} = 1.7294$  s [19].

The comparison of step responses with packet dropouts is shown in Figure 5. We can see that the step response with 70% packets transmitted is similar to the original system. A large difference can be seen when only 20% of packets are transmitted (but the system is still stable, as we prove below).



**Figure 5:** Scaled step responses with packet dropouts.

The lower the transmission rate, the less network bandwidth used. The next question would be, “What is the lower bound on dropout rate  $r$  that still guarantees the stability of the system?”

**Theorem 8 (Bound on Dropout Rate, [19])**

Consider the setup of Figure 4, assuming that the closed-loop system with no dropout is stable (i.e.,  $\Phi - \Gamma K$  is Schur).

- If open-loop system  $(\Phi)$  is marginally stable, the system is exponentially stable for all  $0 < r \leq 1$ .
- If the open-loop system is unstable, then the system is exponentially stable for all

$$\frac{1}{1 - \gamma_1/\gamma_2} < r \leq 1,$$

$$\gamma_1 = \log[\lambda_{\max}^2(\Phi - \Gamma K)], \quad \gamma_2 = \log[\lambda_{\max}^2(\Phi)].$$

**Example 9** Example 5 has  $\Phi$  marginally stable, so the networked controller is stable for all  $r > 0$ .

**2.3 Scheduling of NCSs Revisited**

In Section 1, we studied the scheduling of a set of NCSs using the RM scheduling algorithm. There, we implicitly assumed that every data packet is successfully delivered. In Section 2.2, we showed that we can drop a certain percentage of data packets and still provide a stable NCS. This idea can be applied to schedule a set of NCSs. Roughly speaking, if a set of NCSs is unschedulable when every data packet is guaranteed to be delivered, we might consider dropping some of the packets (at a certain percentage) of the faster-sampling NCSs, and still try to guarantee the set of NCSs are all stable.

More formally, consider a set of NCSs given by

$$x_i((k+1)h_i) = \Phi_i x_i(kh_i) - \Gamma_i K_i \hat{x}_i(kh_i), \quad i = 1, \dots, N, \quad (9)$$

each connected to the network, and each with  $r_i \times 100\%$  samples transmitted. For each NCS  $i$ , define the augmented state vector as  $w_i(kh_i) = [x_i^T(kh_i), \hat{x}_i^T(kh_i)]^T$ , the transmission indicator function  $s_i(kh_i)$ , and the system matrix  $\tilde{\Phi}_{i,s_i(kh_i)}$  accordingly as in Section 2.2.

Now for the overall system, define the augmented state as

$$w(k) = [w_1^T(kh_1), w_2^T(kh_2), \dots, w_N^T(kh_N)]^T.$$

The resulting system matrix is in block diagonal form:

$$\tilde{\Phi}(k) = \text{diag} \left( \tilde{\Phi}_{1,s_1(kh_1)}, \tilde{\Phi}_{2,s_2(kh_2)}, \dots, \tilde{\Phi}_{N,s_N(kh_N)} \right).$$

**Theorem 10 (Stability of a Set of NCSs, [19])**

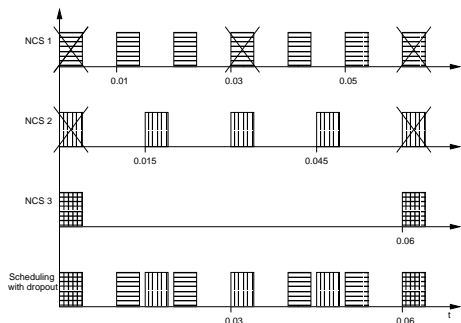
Given the setup above, assume for each NCS  $i$  there exists a Lyapunov function  $V(w_i(kh)) = w_i^T(kh_i)P_i w_i(kh_i)$  and scalars  $\alpha_{i,1}$  and  $\alpha_{i,2}$  such that

$$\begin{aligned} \alpha_{i,1}^{r_i} \alpha_{i,2}^{1-r_i} &> 1, \\ \tilde{\Phi}_{i,1}^T P_i \tilde{\Phi}_{i,1} &\leq \alpha_{i,1}^{-2} P_i, \\ \tilde{\Phi}_{i,2}^T P_i \tilde{\Phi}_{i,2} &\leq \alpha_{i,2}^{-2} P_i, \end{aligned}$$

then the system of Equation (9) is exponentially stable.

**Example 11 (Scheduling of NCSs Revisited)**

Consider the same setup as in in Example 3. Let the transmission periods of the three NCSs be  $h_1 = 0.01$  s,  $h_2 = 0.015$  s, and  $h_3 = 0.06$  s, respectively. The individual transmission sequences are illustrated in the top three rows in Figure 6.



**Figure 6:** Scheduling of NCSs with packet dropout

The set of NCSs is not guaranteed to be schedulable by RM scheduling, as shown by the schedulability analysis using Theorem 2 below:

$$\begin{aligned} \frac{c_1}{h_1} + \frac{\bar{b}_{l,1}}{h_1} &= 0.8 < 1(2^1 - 1) = 1, \\ \frac{c_1}{h_1} + \frac{c_2}{h_2} + \frac{\bar{b}_{l,2}}{h_2} &= 0.9333 > 2(2^{1/2} - 1) = 0.8284. \end{aligned}$$

We see, in Figure 6, that message collisions happen at  $0.06 * j$  (NCS 1, 2, 3 are involved) and  $0.06 * j + 0.03$  (NCS 1, 2 are involved),  $j = 0, 1, \dots$ . Dropping some of the packets, as shown in the fourth row of Figure 6, makes the network schedulable. In particular, we drop messages from NCS 1, 2 at  $0.06 * j$ , and the message from NCS 1 at  $0.06 * j + 0.03$ . The packet drop rate for each NCS is shown in the following table.

NCS $i$	1	2	3
Dropout rate $(1 - r_i)$	0.33	0.25	0.0

**Table 3:** Packet drop rate for NCSs in Example 11

Applying Corollary 6 for each NCS  $i$ , we can find  $\alpha_{i,1}$ ,  $\alpha_{i,2}$ , and  $P_i$  for each  $i$  as follows:

$$\begin{aligned} \alpha_{1,1} &= 1.3342, & P_1 &= \begin{bmatrix} 15.2121 & -11.9942 \\ -11.9942 & 11.4825 \end{bmatrix}, \\ \alpha_{1,2} &= 0.5645, \\ \alpha_{2,1} &= 1.2595, & P_2 &= \begin{bmatrix} 12.1012 & -10.3186 \\ -10.3186 & 11.6616 \end{bmatrix}, \\ \alpha_{2,2} &= 0.5433, \\ \alpha_{3,1} &= 1.2585, & P_3 &= \begin{bmatrix} 5.3734 & -6.7656 \\ -6.7656 & 9.0446 \end{bmatrix}. \end{aligned}$$

Now, applying Theorem 10 we can see that the set of NCSs scheduled by dropping some packets is stable.

Further increasing the transmission period of each NCS will push the network utilization factor,  $U$ , greater than 1. Thus the network is absolutely unschedulable. The merit of dropping a certain percentage of packets is more obvious, since by doing this we may be able to make the network schedulable without loss of stability. This is illustrated by the following example.

**Example 12** Consider Example 11 again, except with even shorter transmission periods. Let the transmission periods of the NCSs be  $h_1 = 0.008$  s,  $h_2 = 0.012$  s, and  $h_3 = 0.016$  s, respectively. The set of NCSs is unschedulable because the network utilization factor

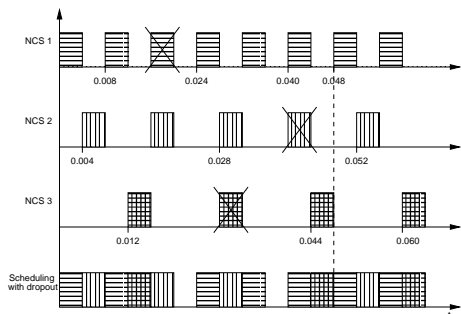
$$U = \sum_{i=1}^3 \frac{c_i}{h_i} = \frac{0.004}{0.008} + \frac{0.004}{0.012} + \frac{0.004}{0.016} = 1.0833 > 1.$$

However, following the transmission and dropout pattern as shown in Figure 7, the network is schedulable with the dropout rates shown in Table 4. Also note that the first transmission of NCS 1 arrives at 0 s, the first transmission of NCS 2 arrives at 0.004 s, and the first transmission of NCS 3 arrives at 0.012 s.

We can further guarantee each NCS's stability by applying Corollary 6 for each NCS  $i$ . We can find  $\alpha_{i,1}$ ,

NCS $i$	1	2	3
Dropout rate $(1 - r_i)$	0.17	0.25	0.33

**Table 4:** Packet dropout rate for NCSs in Example 12



**Figure 7:** Transmission and dropout pattern of a set of unschedulable NCSs

$\alpha_{i,2}$ , and  $P_i$  for each  $i$  as follows:

$$\begin{aligned} \alpha_{1,1} &= 1.1884, & P_1 &= \begin{bmatrix} 4.3627 & -2.7988 \\ -2.7988 & 2.5955 \end{bmatrix}, \\ \alpha_{1,2} &= 0.5860, \\ \alpha_{2,1} &= 1.2595, & P_2 &= \begin{bmatrix} 12.1012 & -10.3186 \\ -10.3186 & 11.6616 \end{bmatrix}, \\ \alpha_{2,2} &= 0.5433, \\ \alpha_{3,1} &= 1.2493, & P_3 &= \begin{bmatrix} 3.0308 & -2.1632 \\ -2.1632 & 2.6542 \end{bmatrix}. \\ \alpha_{3,2} &= 0.6584, \end{aligned}$$

Now, applying Theorem 10 we can see that the set of NCSs scheduled by dropping some packets is stable.

Examples 3, 11, and 12 reiterate NCS tradeoffs. Due to the limitation of network bandwidth, if we want the set of NCSs to be schedulable, then everybody has to “slow down” a little, using a slower transmission rate. Example 3 shows that NCSs with slower transmission rates can be scheduled using RM scheduling. On the other hand, if everybody has a faster transmission rate, then somebody has to “back off” at some point, dropping some packets, as illustrated in Examples 11 and 12. Both methods may provide stable NCSs.

## 2.4 NCSs with Multiple-Packet Transmission

An NCS with multiple-packet transmissions can also be modeled as an ADS. See [19] for examples and results.

## References

[1] P. Albertos *et al.* RT control scheduling to reduce control performance degrading, in *Proc. IEEE Conf. on Decision and Control*, pp. 4889–4894, Sydney, Dec. 2000.  
[2] K. Årzén, A. Cervin, and J. Eker, An introduction to control and scheduling co-design, in *Proc. IEEE Conf. on Decision and Control*, pp. 4865–4870, Sydney, Dec. 2000.

[3] A. Bar-Noy *et al.* Minimizing service and operation costs of periodic scheduling, in *Proc. ACM-SIAM Symp. on Discrete Algorithms*, pp. 11–20, San Francisco, Jan. 1998.  
[4] M.S. Branicky, S.M. Phillips, and W. Zhang, Stability of networked control systems: Explicit analysis of delay, in *Proc. American Control Conf.*, pp. 2352–2357, Chicago, June 2000.  
[5] G.C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Kluwer Academic Publishers, Norwell, MA, 1997.  
[6] G.C. Buttazzo, Adaptive rate control through elastic scheduling, in *Proc. IEEE Conf. on Decision and Control*, pp. 4883–4888, Sydney, Dec. 2000.  
[7] A. Cervin and J. Eker, Feedback scheduling of control tasks, in *Proc. IEEE Conf. on Decision and Control*, pp. 4871–4876, Sydney, Dec. 2000.  
[8] P. Gahinet, A. Nemirovshi, A. Laub, and M. Chilali, *LMI Control Toolbox*, Natick, MA: MathWorks, May 1995.  
[9] A. Hassibi, S.P. Boyd, and J.P. How, Control of asynchronous dynamical systems with rate constraints on events, in *Proc. IEEE Conf. on Decision and Control*, pp. 1345–1351, Phoenix, Dec. 1999.  
[10] J. Lehoczky, L. Sha, and Y. Ding, The rate monotonic scheduling algorithm: Exact characterization and average case behavior, in *Proc. IEEE Real-Time Systems Symp.*, pp. 166–171, Dec. 1989.  
[11] C.L. Liu and J.W. Layland, Scheduling algorithm for multiprogramming in a hard-real-time environment, *J. ACM*, **20**(1):46–61, January 1973.  
[12] J.W.S. Liu, *Real-Time Systems*, Prentice Hall, 2000.  
[13] J. Nilsson, *Real-Time Control Systems with Delays*, Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.  
[14] Open DeviceNet Vendor Association, *DeviceNet Specification*, Boca Raton, FL, 1997. Version 2.0.  
[15] A. Ray and Y. Halevi, Integrated communication and control systems: Part II—Design considerations, in *ASME Journal of Dynamic Systems, Measurement, and Control*, **110**:374–381, Dec. 1988.  
[16] L. Sha, R. Rajkumar, and J. Lehoczky, Priority inheritance protocols: An approach to real-time synchronization, *IEEE Trans. on Computers*, **39**(9):1175–1185, September 1990.  
[17] L. Sha, X. Liu, M. Caccamo, and G. Buttazzo, Online control optimization using load driven scheduling, in *Proc. IEEE Conf. on Decision and Control*, pp. 4877–4882, Sydney, Dec. 2000.  
[18] G.C. Walsh, H. Ye, and L. Bushnell, Stability analysis of networked control systems, in *Proc. American Control Conf.*, pp. 2876–2880, San Diego, June 1999.  
[19] W. Zhang, *Stability Analysis of Networked Control Systems*, Ph.D. Thesis, Electrical Engineering and Computer Science, Case Western Reserve U., May 2001. <http://dora.cwru.edu/msb/pubs/wxzPHD.pdf>  
[20] W. Zhang, M.S. Branicky, and S.M. Phillips, Stability of networked control systems, *IEEE Control Systems Magazine*, **21**(1):84–99, February 2001.