Overload Management in Sensor-Actuator Networks used for Spatially-Distributed Control Systems

M.D. Lemmon, Q. Ling, Y. Sun
Dept. of Electrical Engineering
University of Notre Dame

April 9, 2003

abstract: Overload management policies avoid network congestion by actively dropping packets. This paper studies the effect that such data dropouts have on the performance of spatially distributed control systems. We formally relate the spatially-distributed system's performance (as measured by the average output signal power) to the data dropout rate. This relationship is used to pose an optimization problem whose solution is a Markov chain characterizing a dropout process that maximizes control system performance subject to a specified lower bound on the dropout rate. We then use this Markov chain to formulate an overload management policy that enables nodes to enforce the "optimal" dropout process identified in our optimization problem. Simulation experiments are used to verify the paper's claims.

1 Introduction

A *sensor-actuator network* is a collection of sensors and actuators that exchange information over a shared communication network. Such networks can be used in active structural acoustic control [Fra01] or other spatially distributed control systems [D'An99]. An example of such a system is shown in figure 1. This system [D'An99] is a linear array of masses that are linked together by springs. Each mass (node) has an embedded processor attached to it that can measure the node's local state (position) and then transmit that information to its neighboring nodes. This transmitted information is then used by the embedded processor to compute a control input. Distributed controllers for such systems can be designed using a host of methods. Most of these controllers, however, assume there is no loss of data through the network. This paper examines the effect dropouts have on overall system performance and devises policies for optimally managing these dropouts.

Data dropouts occur for many reasons, but this paper confines its attention to networks in which 1) data are dropped due to link failures or 2) data are purposefully dropped in order to stay within a specified bandwidth allocation. The first source of data dropouts cannot be controlled by the nodes and we assume that such dropouts occur at a rate f that is statistically independent of the data source. The second source of dropouts occurs because overload management policies require
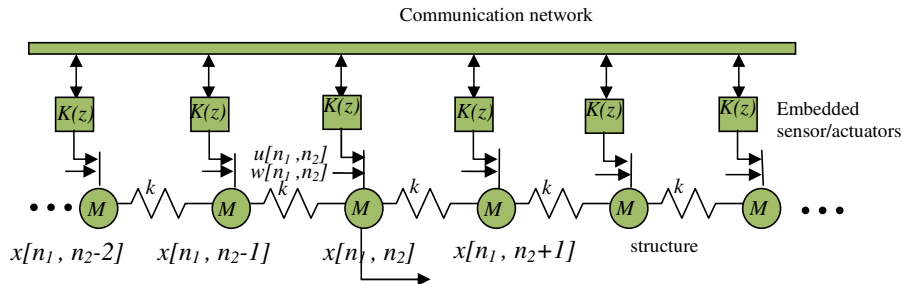
Figure 1: Distributed Control System and Sensor-Actuator Network

nodes to purposefully drop packets to prevent network congestion. In particular, the network allocates a portion of its available bandwidth to each node. Each node then selectively drops packets to stay within this bandwidth allocation. Previous work in overload management [Ram99] has focused on the use of heuristic policies such as the (m, k)-firm guarantee rule. This paper provides an alternative approach to overload management that has provable guarantees on the attainable level of application performance.

The remainder of this paper is organized as follows. Section 2 presents the spatially distributed system studied in this paper. Section 3 discusses prior work relating control system performance and data dropouts. This prior work only pertains to single control loops, so section 4 extends that work to the spatially distributed system in figure 1. Section 5 uses these results to formulate an "optimal" overload management policy and the behavior of this policy is demonstrated on a simulation of a 27-node structure. Final remarks are found in section 6. The theorems' proofs will be found in section 7.

2 Spatially Distributed Control System

This paper confines its attention to the spatially distributed system shown in figure 1. We first assume that the continuous-state dynamics of the system have been discretized in time. So we let $x[n_1, n_2]$ denote the state of the n_2th node at time instant n_1. The state x is a 2-vector characterizing the position and velocity of the node with respect to its equilibrium position. The discrete state satisfies the following recursive equations,

$$
\begin{aligned}
x[n_1 + 1, n_2] &= Ax[n_1, n_2] + B(x[n_1, n_2 - 1] + x[n_1, n_2 + 1]) + F(u[n_1, n_2] + w[n_1, n_2]) \quad (1) \\
z[n_1, n_2] &= Cx[n_1, n_2]
\end{aligned}
$$

for $n_1 \geq 0$ and any n_2. $z[n_1, n_2]$ is an output signal that is used to characterize overall system performance. A, B, C and F are appropriately dimensioned real-valued matrices. There are two inputs to this equation; the disturbance $w[n_1, n_2]$ and the control $u[n_1, n_2]$. The disturbance is a zero-mean white noise process in both time and space. The control input is computed by the embedded processor.

Each node has a processor attached to it. The processor measures the node's local state $x[n_1, n_2]$ and it transmits this information to its neighbors upon request. We assume that the nodes are synchronized in time and that in each sampling interval the node decides whether or not to access its neighbor's state. This means that a node first "requests" that its neighbors send data to it and then the processor computes its control input $u[n_1, n_2]$ upon receiving this data. If neighboring state information has been received, then the control input is computed according to the following equation,

$$u[n_1, n_2] = K_0 x[n_1, n_2] + K_1(x[n_1, n_2 - 1] + x[n_1, n_2 + 1]) \tag{2}$$

where K_0 and K_1 represent control gain matrices that have been chosen by the control engineer. Since our network may occasionally drop packets, the processor needs to use a different control signal if the neighboring state data is not received. In this case, the processor simply sets $u[n_1, n_2] = 0$.

Data will always be dropped by the network. These dropouts occur for two reasons. The first reason is that the medium is unreliable. A transmitted packet has a finite probability f of being lost due to link failure. This probability is assumed to be statistically independent from the state of the packet's source. Dropouts will also occur because a node explicitly decides NOT to request neighboring state measurements. This occurs because an *overload management policy* requires nodes to drop a certain percentage of packets when the network is congested. In particular, the network allocates a specified amount of its bandwidth to each node which we characterized as a lower bound, ε_d, on the node's actual dropout rate. The size of ε_d depends on the amount of network congestion. Overload management through packet dropping clearly has an adverse impact on overall application performance. This is particularly true for hard real-time feedback control systems. This paper determines an overload management policy that is "optimal" in that it maximizes application (control system) performance while ensuring the dropout rate does not drop below ε_d.

Because dropouts cause us to switch between two different control laws, the system's state space model takes the form of a *jump linear system* [Mar90]. In particular, let's define a *dropout process* that is denoted as $d[n_1, n_2]$. It is a binary random process in which $d[n_1, n_2] = 1$ if a dropout occurs and is zero otherwise. Under the dropout process, our system equations take the form,

$$
\begin{aligned}
x[n_1 + 1, n_2] &= A[n_1, n_2]x[n_1, n_2] + B[n_1, n_2](x[n_1, n_2 - 1] + x[n_1, n_2 + 1]) + Fw[n_1, n_2] \quad (3) \\
z[n_1, n_2] &= Cx[n_1, n_2]
\end{aligned}
$$

where $A[n_1, n_2]$ and $B[n_1, n_2]$ are matrix valued random processes such that

$$
A[n_1, n_2] = \begin{cases} A_0 = A + FK_0 & \text{if no dropouts occur (i.e., } d[n_1, n_2] = 0) \\ A_1 = A & \text{if a dropout occurs (i.e., } d[n_1, n_2] = 1) \end{cases}
$$

$$
B[n_1, n_2] = \begin{cases} B_0 = B + FK_1 & \text{if no dropouts occur (i.e., } d[n_1, n_2] = 0) \\ B_1 = B & \text{if a dropout occurs (i.e., } d[n_1, n_2] = 1) \end{cases}
$$

Application performance will be measured by the average power in the control system's output signal. This is a standard measure of performance for regulation problems. In our case, we

want to suppress the effect that the disturbance $w[n_1, n_2]$ has on the system's shape. In particular we assume that w is a white noise process whose covariance matrix is $R = \mathbf{E}\left[Fw[n_1, n_2]w^T[n_1, n_2]F^T\right]$. The control objective is to minimize the noise power in the node's state. So a natural measure of application performance is the average power, $\|z\|_{\mathcal{P}}^2$, in the node's output signal $z[n_1, n_2]$. This power is usually written as

$$\|z\|_{\mathcal{P}}^2 = \mathrm{Trace}\mathbf{E}\left[z[n_1, n_2]z^T[n_1, n_2]\right] = \mathrm{Trace}\left[C\overline{P}_0C^T\right]$$

where $\overline{P}_0$ is the covariance matrix

$$\overline{P}_0 = \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2]\right] \tag{4}$$

Note that throughout this paper we are assuming all nodes in the system are "identical", so that the above covariance matrix is independent of n_1 and n_2. Our problem is to find a way of evaluating $\overline{P}_0$ as a function of the dropout process, $d[n_1, n_2]$.

Throughout this paper, the dropout process $d[n_1, n_2]$ will be generated by an N-state Markov chain with states $\{q_1, q_2, \cdots, q_N\}$, transition probability matrix $Q = [q_{ij}]_{N \times N}$, and stationary distribution $\pi = [\pi_1, \pi_2, \cdots, \pi_N]$. Each node of our system will instantiate a copy of this Markov chain. The state of the Markov chain at node n_2 at time n_1 will be denoted as $q[n_1, n_2]$. The Markov chain generates the dropout process through the function h that maps each Markov state, q_i, onto either 0 or 1. The dropout process is then defined by the equation $d[n_1, n_2] = h(q[n_1, n_2])$. We can therefore see that the matrix-valued processes $A[n_1, n_2]$ and $B[n_1, n_2]$ will take values based on the value of $q[n_1, n_2]$. In particular, if $q[n_1, n_2] = q_i$, then we'll denote $A[n_1, n_2]$ and $B[n_1, n_2]$ as A_i and B_i, respectively.

3 Prior Work

There is a small amount of work studying the effect of dropouts on the performance of networked control systems. Nearly all of this work has confined its attention to single control loops, rather than the multiple coupled control loops found in this paper's application. Early work in [Nil98] treated the dropout process as a Markov chain and developed ad hoc schemes for dealing with dropouts. In [Zha01], the system with dropouts was treated as an asynchronous switched system and switched system analysis methods were used to assess overall system stability. Much of the earlier work focused on system stability. More recently, there have been papers examining system *performance* as a function of the dropout process. In [Had02], this was done for a simple networked control system in which dropped measurements were replaced by zeros. In [Sei01], system performance was measured by its $\mathcal{H}_\infty$ gain and this gain was evaluated as a function of packet loss. Similar results were obtained in [Lin02] and [Lin03] for networked control systems in which the dropout process was modelled as a Markov chain.

The results in [Sei01] and [Lin03] are of particular importance to this paper because they provide formal relationships between system performance and dropout rates. This relationship was used [Lin03] to pose an optimization problem whose solution (if it exists) is a dropout process maximizing control system performance (average output power) subject to a lower bound on the

average dropout rate. The lefthand plot in figure 2 shows results from this prior work. This plot compares the performance of the system under three different stochastic dropout policies; optimal, i.i.d, and a "soft" (m, k)-firm guarantee rule [Ram99] . In this example the system is closed loop stable and open loop unstable (see [Lin03] for details). The figure plots the average output power as a function of the allowed transmission rate (i.e. 1 minus the average dropout rate). As the transmission rate decreases, the average power increases (worse performance) and asymptotically approaches infinity. The results show that an overload management policy enforcing the optimal dropout process performs better (lower average output signal power over a wider range of dropout rates) than policies enforcing the i.i.d or (m, k)-firm guarantee dropouts processes.
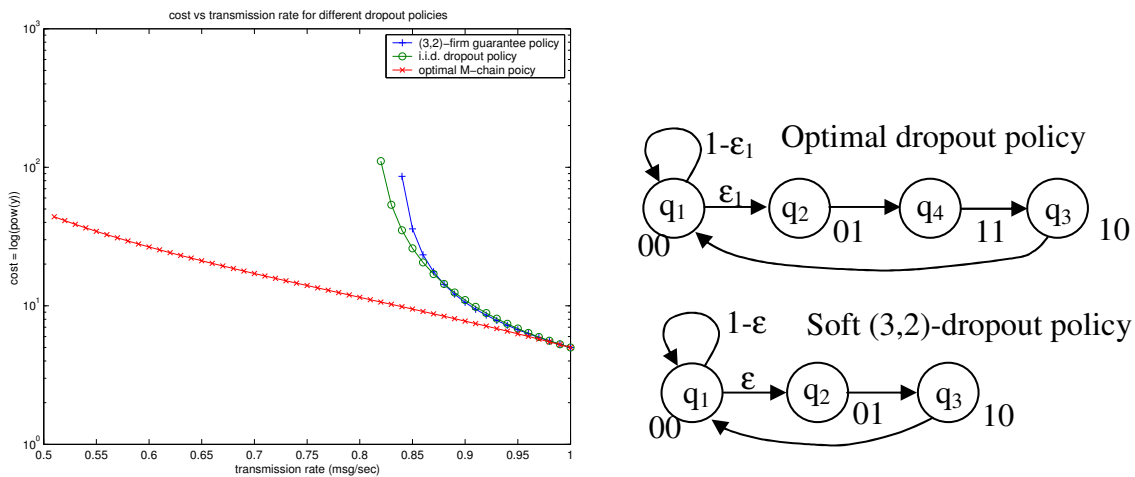

Figure 2: Performance of Various Overload Management Policies

The results in figure 2 are particularly interesting because the behavior of the optimal dropout process runs counter to prevailing intuition. The righthand pictures in figure 2 are the state diagrams for the optimal and "soft" $(3, 2)$ dropout processes. The state of each Markov chain is denoted as $d[n_1 - 1]d[n_1]$, so that the state marked "11" means that the last two packets were dropped. Note that the optimal dropout process requires that if a single dropout occurs, we must immediately drop the second packet as well. The soft $(3, 2)$ rule, however, requires that only one packet be dropped in a row. The results show that for this particular system, it is always better to drop two packets than one. This runs directly counter to the prevailing wisdom embodied in the (m, k)-firm guarantee rule. In the (m, k)-firm guarantee rule, we require that k out of m consecutive packets get through the network. Experimental results in [Ram99] showed that overload management policies enforcing this rule perform well on some specific examples. This prior work, however, provides little formal analysis to suggest that this is necessarily the best policy to follow for all control systems. The results in figure 2 show that "soft" variations on the (m, k) heuristic may sometimes perform poorly and show that it is indeed possible to derive overload management policies that optimally utilize the bandwidth allocated to the controller.

Results of this type are clearly relevant to overload management, for they provide a system-

5

atic method by which optimal management policies might be formulated. The results in [Sei01] and [Lin03] , however, are not directly applicable to the system shown in figure 1 because that work only pertains to single control loop whose feedback is implemented over a network. In systems controlled by sensor-actuator networks, there are a large number of control loops that are coupled through their environment. This is precisely the situation encountered in figure 1. In our system, every node has an embedded processor that implements a local controller. The controller's actions are determined by the neighboring node states and those states are in turn influenced by the local controller through physical interactions in the environment. If we are to develop optimal overload management policies we will need to extend that prior work to this particular class of spatially distributed systems.

4   Performance of Spatially Distributed Systems

This section states two new results concerning the performance of the distributed system in figure 1. Theorem 4.1 characterizes the average output power $\overline{P}_0$ for the distributed system without control (i.e., $u[n_1, n_2] = 0$). Theorem 4.2 extends theorem 4.1 to the jump linear systems found in equation 3. This section simply states the theorems and comments on their significance. Formal proofs of the theorems will be found in section 7.

The first theorem characterizes $\overline{P}_0$ for a non-switching spatially distributed control system. In particular, it states that if the system is stable in the mean square sense (i.e. $\mathbf{E}\left[x^T[n_1, n_2]x[n_1, n_2]\right] < \infty$), then $\overline{P}_0$ is obtained from the solution of an infinite system of linear equations.

Theorem 4.1 *Let $x[n_1, n_2]$ satisfies equation 1 without control input (i.e., $u[n_1, n_2] = 0$) where $w[n_1, n_2]$ is a noise process with covariance matrix R.*

If $\mathbf{E}\left[x^T[n_1, n_2]x[n_1, n_2]\right] < \infty$ (i.e. stability in the mean square sense) , then $\overline{P}_0$ (see Eq. 4) is obtained by solving the following system of equations.

$$\overline{P}_0 = A\overline{P}_0A^T + 2B\overline{P}_0B^T + A(\overline{P}_1 + \overline{P}_1^T)B^T + B(\overline{P}_1 + \overline{P}_1^T)A^T + B(\overline{P}_2 + \overline{P}_2^T)B^T + R \quad (5)$$
$$\overline{P}_1 = A\overline{P}_1A^T + A(\overline{P}_2 + \overline{P}_0)B^T + B(\overline{P}_2 + \overline{P}_0)A^T + B(\overline{P}_1^T + 2\overline{P}_1 + \overline{P}_3)B^T$$
$$\overline{P}_k = A\overline{P}_kA^T + A(\overline{P}_{k+1} + \overline{P}_{k-1})B^T + B(\overline{P}_{k+1} + \overline{P}_{k-1})A^T + B(\overline{P}_{k-2} + 2\overline{P}_k + \overline{P}_{k+2})B^T$$

where the last equation applies for $k \geq 2$.

Remark: The assumption of stability is essential in this theorem. This paper does not determine sufficient conditions for the system to be stable in the mean-square sense, but related results have been in obtained in [Lin02] for single networked control loops. Similar stability conditions may be applicable for this distributed systems, but we have not included such results in this paper due to space limitations.

Remark: Equations 5 is an infinite system of equations in which you must solve for the matrices $\overline{P}_k$ for $k = 0, \ldots, \infty$. $\overline{P}_0$ was defined in equation 4. The other matrices represent the

expectation

$$\overline{P}_k = \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k]\right] \tag{6}$$
$$\overline{P}_k = \overline{P}^T_{-k}$$

Note that because all nodes are identical, this system is shift invariant with respect to both n_1 and n_2. This is why we can drop any explicit mention of n_1 and n_2 in equation 6.

Remark: The infinite set of equations (Eq. 5) may be solved numerically in a recursive manner. In particular, we generate a sequence, $\{\overline{P}_k[L]\}_{L=0}^{\infty}$, of matrices that converge to the true $\overline{P}_k$ as L goes to infinity. The recursion we used is

$$
\begin{aligned}
\overline{P}_0[L+1] &= A\overline{P}_0[L]A^T + 2B\overline{P}_0[L]B^T + A(\overline{P}_1[L]+\overline{P}_1[L]^T)B^T \\
&\quad + B(\overline{P}_1[L]+\overline{P}_1[L]^T)A^T + B(\overline{P}_2[L]+\overline{P}_2[L]^T)B^T + R \\
\overline{P}_1[L+1] &= A\overline{P}_1[L]A^T + A(\overline{P}_2[L]+\overline{P}_0[L])B^T + B(\overline{P}_2[L]+\overline{P}_0[L])A^T \\
&\quad + B(\overline{P}_1[L]^T + 2\overline{P}_1[L] + \overline{P}_3[L])B^T \\
\overline{P}_k[L+1] &= A\overline{P}_k[L]A^T + A(\overline{P}_{k+1}[L]+\overline{P}_{k-1}[L])B^T + B(\overline{P}_{k+1}[L]+\overline{P}_{k-1}[L])A^T \\
&\quad + B(\overline{P}_{k-2}[L]+2\overline{P}_k[L]+\overline{P}_{k+2}[L])B^T
\end{aligned}
\tag{7}
$$

where we let $\overline{P}_0[0] = R$ and $\overline{P}_k[0] = 0$ for $k \neq 0$. The recursion is terminated when $\left\|\overline{P}_0[L+1] - \overline{P}_0[L]\right\|$ is less than a specified error tolerance.

We extend theorem 4.1 to spatially-distributed systems with dropouts. Such systems are modelled as jump linear systems characterized by equation 3. This theorem provides an infinite set of equations that can be used to solve for the covariance matrix,

$$\overline{P}_0 = \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2]\right]$$

Once again because of spatial shift-invariance, $\overline{P}_0$ is independent of n_2. The proof for this theorem will be found in section 7.

Theorem 4.2 *Let $w[n_1,n_2]$ be a zero-mean white noise process with covariance R. Let $x[n_1,n_2]$ satisfy the jump linear system equation given in equations 3 which is driven by a Markov chain with transition matrix $Q = [q_{ij}]_{N\times N}$ with stationary distribution $\pi = [\pi_1, \pi_2, \cdots, \pi_N]$.*

If $\mathbf{E}\left[x^T[n_1,n_2]x[n_1,n_2]\right] < \infty$ (i.e. mean square stability), then

$$\overline{P}_0 = \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2]\right] = \sum_{i=1}^{N}\overline{P}_0^{i,i}$$

where $\overline{P}_0^{i,i}$ satisfy the following infinite set of equations.

$$\overline{P}_0^{i,i} = \sum_{l=1}^{N}\left[q_{li}A_i\overline{P}_0^{l,l}A_i^T + 2\pi_iB_i\overline{P}_0^{l,l}B_i^T\right] + \sum_{l,m=1}^{N}\left[\pi_iB_i\left(\overline{P}_2^{l,m}+\overline{P}_{-2}^{l,m}\right)B_i^T\right] \tag{8}$$

$$+ \sum_{l,m=1}^{N} \left[q_{li} A_i \left(\overline{P}_1^{l,m} + \overline{P}_{-1}^{l,m} \right) B_i^T + q_{mi} B_i \left(\overline{P}_1^{l,m} + \overline{P}_{-1}^{l,m} \right) A_i^T \right] + \pi_i R$$

$$\overline{P}_1^{i,j} = B_i \overline{P}_{-1}^{j,i} B_j^T + \sum_{l=1}^{N} \left[\pi_j q_{li} A_i \overline{P}_0^{l,l} B_j^T + \pi_i q_{lj} B_i \overline{P}_0^{l,l} A_j^T \right]$$

$$+ \sum_{l,m=1}^{N} \left[q_{li} q_{mj} A_i \overline{P}_1^{l,m} A_j^T + \pi_i q_{lj} B_i \overline{P}_1^{l,m} B_j^T + \pi_j q_{mi} B_i \overline{P}_1^{l,m} B_j^T \right]$$

$$+ \sum_{l,m=1}^{N} \left[\pi_j q_{li} A_i \overline{P}_2^{l,m} B_j^T + \pi_i q_{mj} B_i \overline{P}_2^{l,m} A_j^T + \pi_i \pi_j B_i \overline{P}_3^{l,m} B_j^T \right]$$

$$\overline{P}_k^{i,j} = \sum_{l,m=1}^{N} \left[q_{li} q_{mj} A_i \overline{P}_k^{l,m} A_j^T + 2\pi_i \pi_j B_i \overline{P}_k^{l,m} B_j^T \right]$$

$$+ \sum_{l,m=1}^{N} \left[\pi_j q_{li} A_i \left(\overline{P}_{k+1}^{l,m} + \overline{P}_{k-1}^{l,m} \right) B_j^T + \pi_i q_{mj} B_i \left(\overline{P}_{k+1}^{l,m} + \overline{P}_{k-1}^{l,m} \right) A_j^T \right]$$

$$+ \sum_{l,m=1}^{N} \left[\pi_i \pi_j B_i \left(\overline{P}_{k-2}^{l,m} + \overline{P}_{k+2}^{l,m} \right) B_j^T \right]$$

for $k \geq 2$.

Remark: Note that there are strong similarities between the equations 8 of theorem 4.2 and equations 5 of theorem 4.1.

Remark: Equations 8 is an infinite set of linear equations that we solve for the matrices $\overline{P}_k^{i,j}$. In particular, these matrices are the following conditional expectations.

$$\overline{P}_k^{i,j} = \pi_i \pi_j \mathbf{E} \left[x[n_1, n_2] x^T[n_1, n_2 + k] \mid q[n_1 - 1, n_2] = q_i, q[n_1 - 1, n_2 + k] = q_j \right] \quad \text{for } k \neq 0$$

$$\overline{P}_0^{i,i} = \pi_i \mathbf{E} \left[x[n_1, n_2] x^T[n_1, n_2] \mid q[n_1 - 1, n_2] = q_i \right]$$

$$\overline{P}_0^{i,j} = 0 \quad \text{for } i \neq j$$

We can again use a recursion similar to that shown in equation 7 to solve these equations.

5 Optimal Overload Management Policies

Theorem 4.2 presents a method for computing the covariance, $\overline{P}_0$, of a node's state vector as a function of the dropout process' transition matrix, Q. Since we take $\text{Trace}(C\overline{P}_0 C^T)$ to be a measure of the local controller's performance, these results provide the required extension of [Lin03] to spatially distributed control systems. We can now use this result to formulate and solve a problem to find the Markov chain that minimizes $\text{Trace}(C\overline{P}_0 C^T)$ subject to a lower bound ε_d

on the chain's average dropout rate. The resulting Markov chain will then be used to devise an "optimal" overload management policy.

To state our problem, we need to consider a specific Markov chain. In particular, let's assume the Markov chain has four states defined as follows:

$$
\begin{aligned}
q_1[n_1, n_2] &\quad | \quad d[n_1 - 1, n_2] = 0, d[n_1, n_2] = 0 \\
q_2[n_1, n_2] &\quad | \quad d[n_1 - 1, n_2] = 0, d[n_1, n_2] = 1 \\
q_3[n_1, n_2] &\quad | \quad d[n_1 - 1, n_2] = 1, d[n_1, n_2] = 0 \\
q_4[n_1, n_2] &\quad | \quad d[n_1 - 1, n_2] = 1, d[n_1, n_2] = 1
\end{aligned}
$$

When $q[n_1, n_2] = q_i$, then the probability that the next packet will be dropped on purpose is ε_i. In other words ε_i is the probability that the node decided NOT to request data from its neighbors. The total dropout policy (including the effect of link failures) will be $1 - (1 - \varepsilon_i)(1 - f)$ where f is the link failure rate. With these notational conventions, the probability transition matrix for the Markov chain becomes,

$$
Q = \begin{bmatrix}
(1 - \varepsilon_1)(1 - f) & \varepsilon_1 + f(1 - \varepsilon_1) & 0 & 0 \\
0 & 0 & (1 - \varepsilon_2)(1 - f) & \varepsilon_2 + f(1 - \varepsilon_2) \\
(1 - \varepsilon_3)(1 - f) & \varepsilon_3 + f(1 - \varepsilon_3) & 0 & 0 \\
0 & 0 & (1 - \varepsilon_4)(1 - f) & \varepsilon_4 + f(1 - \varepsilon_4)
\end{bmatrix}
$$

Provided f is small enough, then this is a valid stochastic matrix that has the steady state distribution $\pi = [\pi_1, \pi_2, \pi_3, \pi_4]$.

The *average dropout rate*, $\bar{\varepsilon}$ is given by the following equation

$$
\bar{\varepsilon} = \sum_{i=1}^{4} \pi_i \varepsilon_i.
$$

As discussed earlier, the network will limit a node's transmission rate in order to avoid congestion. This limitation takes the form of a lower bound ε_d on the node's actual dropout rate. In other words, the network requires that the node drops packets at least as fast as ε_d. The overload management policy used by the node must assure that $\bar{\varepsilon} \geq \varepsilon_d$ and that the dropped packets degrade overall control system performance as little as possible. So the optimization problem we seek to solve has the following formal statement,

$$
\begin{aligned}
&\text{minimize:} && \text{Trace} \sum_{i=1}^{N} C \overline{P}_0^{i,i} C^T \\
&\text{with respect to:} && \varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4 \\
&\text{subject to:} && \bar{\varepsilon} \geq \varepsilon_d \\
& && \pi = \pi Q \\
& && 1 = \sum_{i=1}^{N} \pi_i
\end{aligned}
\tag{9}
$$

where $\overline{P}_0^{i,i}$ is computed using the equations in theorem 4.2 and ε_d is a specified constant (i.e. a measure of the bandwidth allocated to the local controller).

We considered a specific system to verify the correctness of our theoretical results. This example is a distributed system in which local controllers switch between open-loop and closed-loop configurations. A closed-loop configuration occurs when $d[n] = 0$. This happens when the Markov chain state, $q[n]$, equals q_1 or q_3. An open-loop configuration occurs when a dropout occurs (i.e. $d[n] = 1$). This happens when the Markov chain's state is either q_2 or q_4. The system matrices used in this experiment were,

$$A = \begin{bmatrix} 0.9990 & 0.0100 \\ -0.1999 & 0.9990 \end{bmatrix}, B = \begin{bmatrix} 0.0005 & 0 \\ 0.1000 & 0 \end{bmatrix}, C = \begin{bmatrix} 1.0 & 0 \\ 0 & 0.1 \end{bmatrix}, F = B.$$

When there is no dropout, the control $u[n_1, n_2]$ takes the form

$$u[n_1, n_2] = Kx[n_1, n_2] - x[n_1, n_2 - 1] - x[n_1, n_2 + 1]$$

where $K = \begin{bmatrix} -93.2580 & -10.4700 \\ 0 & 0 \end{bmatrix}$. The dynamics of the closed-loop system therefore become

$$x[n_1 + 1, n_2] = (A + BK)x[n_1, n_2] + Bw[n_1, n_2],$$

From this equation we see that the closed-loop distributed system is really a group of decoupled subsystems. When a dropout occurs there is no control (i.e. $u[n_1, n_2] = 0$) and the physical coupling between subsystems reasserts itself. So the system matrices A_i and B_i that are switched to when the Markov chain's state is q_i are given as follows:

$$\begin{aligned} A_1 &= A_3 = A + BK \\ B_1 &= B_3 = 0 \\ A_2 &= A_4 = A \\ B_2 &= B_4 = B \end{aligned}$$

For our particular problem, `Matlab`'s optimization toolbox was used to numerically solve the preceding optimization problem. In particular, we used the `Matlab` function `fmincon` after a suitable initial condition was identified. These optimizations were done assuming a link failure rate, f, of 0.3 for ε_d between 0.05 and 0.85. The solutions are the transition probabilities ε_i. We found that these probabilities took the following form,

$$[\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4] = \begin{cases} x_1, 0, 0, 0 & \varepsilon_d < 0.2237 \\ 1, 0, x_3, 0 & 0.2237 < \varepsilon_d < 0.4117 \\ 1, x_2, 1, 0 & 0.4117 < \varepsilon_d < 0.5833 \\ 1, 1, 1, x_4 & 0.5833 < \varepsilon_d \end{cases}$$

where x_1, x_2, x_3, x_4 are determined by the average dropout rate condition.

The plot in figure 3 compares the performance under the "optimal" dropout policy (solid line) and an i.i.d. (independently identically distributed) dropout policy (dashed line). This is the

performance computed using our theoretical results. We plot the log power level ($\log(\text{Trace}\mathbf{E}\left[C\overline{P}_0 C\right]$)
as a function of the transmission rate $1 - \varepsilon_d$. We also simulated the distributed system in figure
1 using a `MatLab SimuLink` model. All of the simulations assumed 27 nodes with the endpoints
having free boundary conditions. The results for these simulations are plotted as $*$ and o. We
simulated 3 different runs at 6 different dropout rates. In the simulations, we estimated the power
by time average taken over 100,000 iterations. The theoretical predictions show that the optimal
policy is indeed better than the i.i.d. policy. In reviewing the transition probabilities given above, it
is apparent that the optimal policy is a soft $(3, 2)$-policy for $\varepsilon_d < 0.2237$. For bandwidth allocations
above this rate, however, the structure of the optimal policy changes to allow higher dropout rates.
The simulation results show close agreement with the theoretical predictions for a wide range of
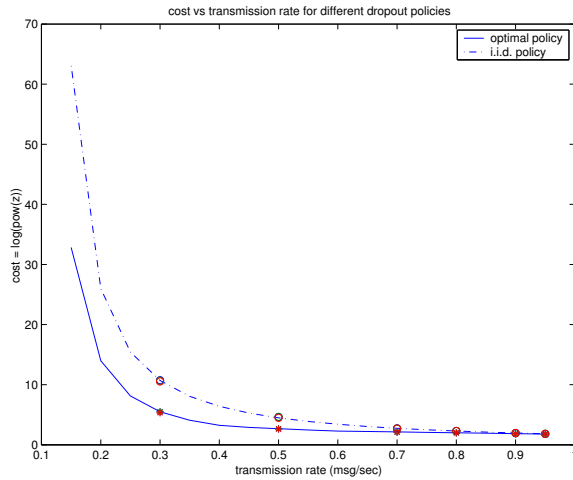dropout rates.

Figure 3: Simulation versus Theoretical Results

The Markov chains derived in this section form the basis for an overload management policy
that is easily implemented on an embedded processor. In particular, a number of these "optimal"
Markov chains would be determined for a range of overload conditions (ε_d) and a range of link
failure rates (f). We store these transition probabilities in a table that is indexed with respect
to ε_d and f. The overload management policy used by each node is a concrete instantiation of
the optimal Markov chain whose transition probabilities are loaded from this table based on 1)
the bandwidth (ε_d) that was allocated to the node and 2) based on the link failure rate (f) that
was estimated by the node. What should be apparent is that the resulting policy is adaptive with
respect to link failure rate and bandwidth allocation. Moreover, since these chains are solutions to
the optimization problem in equation 9, we know that this policy degrades application performance
as little as possible. In other words, we have provable guarantees that this approach makes optimum
use of the bandwidth allocated to the local controller. The simulation and theoretical results shown
in figure 3 suggest that hardware implementations of such policies should also perform well. This
hardware testing, however, remains to be done.

11

6 Final Remarks

One approach to sensor-actuator network middleware advocates the development of software that is *application-independent*. This approach is meaningful for networks that must use a wide variety of traffic types in diverse and unpredictable situations. By their very nature, however, embedded sensor actuator networks are dedicated to a specific type of application whose traffic patterns are often highly regular and predictable. By adopting middleware architectures that ignore our prior knowledge of the plant, we are doomed to under-utilize network resources in support of that application. Ultimately this means we are trading away application performance for software reusability, a trade off that is not always desirable.

This paper demonstrated another approach to network middleware in which application specific knowledge can be used to optimize middleware's support for the application. In particular we were able to use a priori knowledge of the application (in our case a spatially distributed control system) to develop overload management policies that provide provable levels of application performance subject to constraints on the network QoS (as measured by packet dropout rates). The policy developed here is based on a Markov decision process that emerges from solving an optimization problem in which a priori knowledge of the application's dynamics was incorporated. The resulting policy, we feel, can be readily realized as a minor variation on existing overload management schemes, the only real change being that our decision probabilities are based on formal understanding of system dynamics, rather than ad hoc reasoning about the plant.

To the best of our knowledge, this paper provides one of the few positive examples in which network QoS was related to control system performance in a way that directly leads to practical network protocols. Future work will continue in this direction, by attempting to use formal systems science for the systematic development of embedded network middleware. Over the immediate future we intend to relax some of the assumptions inherent in this paper's development, primarily the reliance on spatial shift-invariance and 1-dimensional structures. Over the long term, we intend to validate the ideas developed in this paper on hardware-in-the-loop experiments.

7 Proofs

Proof of Theorem 4.1: Let's define the matrix

$$\overline{P}_k = \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2 + k]\right]$$

Because all nodes are identical, $\overline{P}_k$ has no dependence on either n_1 or n_2. We also know that

$$x[n_1 + 1, n_2] = Ax[n_1, n_2] + B(x[n_1, n_2 - 1] + x[n_1, n_2 + 1]) + Fw[n_1, n_2]$$
$$x[n_1 + 1, n_2 + k] = Ax[n_1, n_2 + k] + B(x[n_1, n_2 + k - 1] + x[n_1, n_2 + k + 1]) + Fw[n_1, n_2 + k]$$

We can now expand out $\overline{P}_k$ as

$$\begin{aligned}
\overline{P}_k &= \mathbf{E}\left[x[n_1 + 1, n_2]x^T[n_1 + 1, n_2 + k]\right] \\
&= A\overline{P}_k A^T + 2B\overline{P}_k B^T + B(\overline{P}_{k-2} + \overline{P}_{k+2})B^T \\
&\quad + A(\overline{P}_{k+1} + \overline{P}_{k-1})B^T + B(\overline{P}_{k+1} + \overline{P}_{k-1})A^T + R_k
\end{aligned}$$

where $R_k = 0$ if $k \neq 0$ and $R_0 = R$ (R is the covariance of the noise process w). The above equation is the third equation in the theorem ($k \geq 2$). We use the fact that $\overline{P}_{-k} = \overline{P}_k^T$ to get the other two equations in the theorem. $\bullet$

Proof of Theorem 4.2: Note that if $q[n_1, n_2] = q_i$ and $q[n_1, n_2 + k] = q_j$, then the states at time $n_1 + 1$ can be written as

$$
\begin{aligned}
x[n_1 + 1, n_2] &= A_i x[n_1, n_2] + B_i(x[n_1, n_2 - 1] + x[n_1, n_2 + 1]) + F w[n_1, n_2] \qquad (10)\\
x[n_1 + 1, n_2 + k] &= A_j x[n_1, n_2 + k] + B_j(x[n_1, n_2 + k - 1] + x[n_1, n_2 + k + 1]) + F w[n_1, n_2 + k]
\end{aligned}
$$

Here, when $q[n_1, n_2] = q_i$, we denote $A(q[n_1, n_2])$ and $B(q[n_1, n_2])$ as A_i and B_i respectively. So we can use this to write out

$$
\begin{aligned}
\overline{P}_k^{i,j} &= \pi_i \pi_j \mathbf{E}\left[x[n_1+1,n_2]x^T[n_1+1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right]\\
&= \pi_i \pi_j A_i \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] A_j^T\\
&\quad + \pi_i \pi_j A_i \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k-1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T\\
&\quad + \pi_i \pi_j A_i \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k+1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2-1]x^T[n_1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] A_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2-1]x^T[n_1,n_2+k-1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2-1]x^T[n_1,n_2+k+1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2+1]x^T[n_1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] A_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2+1]x^T[n_1,n_2+k-1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T\\
&\quad + \pi_i \pi_j B_i \mathbf{E}\left[x[n_1,n_2+1]x^T[n_1,n_2+k+1] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right] B_j^T
\end{aligned}
$$

There are nine conditional expectations in the above equation. The first expectation can be simplified as follows,

$$
\begin{aligned}
&\pi_i \pi_j \; \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j\right]\\
&= \sum_{l,m=1}^{N} \pi_i \pi_j \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k] \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j,\right.\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \left. q[n_1-1,n_2]=q_l, q[n_1-1,n_2+k]=q_m\right]\\
&\qquad P(q[n_1-1,n_2]=q_l, q[n_1-1,n_2+k]=q_m \mid q[n_1,n_2]=q_i, q[n_1,n_2+k]=q_j)\\
&= \sum_{l,m=1}^{N} \pi_i \pi_j \mathbf{E}\left[x[n_1,n_2]x^T[n_1,n_2+k] \mid q[n_1-1,n_2]=q_l, q[n_1-1,n_2+k]=q_m\right]\\
&\qquad P(q[n_1-1,n_2]=q_l \mid q[n_1,n_2]=q_i) P(q[n_1-1,n_2+k]=q_m \mid q[n_1,n_2+k]=q_j)\\
&= \pi_i \pi_j \sum_{l=1}^{N} \sum_{m=1}^{N} \overline{P}_k^{l,m} \frac{q_{mj} q_{li}}{\pi_i \pi_j}
\end{aligned}
$$

$$= \quad \sum_{l=1}^{N} \sum_{m=1}^{N} q_{li} q_{mj} \overline{P}_{l,m}(L)$$

The second expectation can be simplified as shown below. The third, fourth, and seventh expectations have similar derivations and aren't shown.

$$\pi_i \pi_j \quad \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2 + k - 1] \quad | \quad q[n_1, n_2] = q_i, q[n_1, n_2 + k] = q_j\right]$$

$$= \quad \pi_i \pi_j \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2 + k - 1] \quad | \quad q[n_1, n_2] = q_i\right]$$

$$= \quad \pi_i \pi_j \sum_{l,m=1}^{N} \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2 + k - 1] \quad | \quad q[n_1, n_2] = q_i, q[n_1 - 1, n_2] = q_l, q[n_1 - 1, n_2 + k - 1] = q_m\right]$$

$$P(q[n_1 - 1, n_2] = q_l, q[n_1 - 1, n_2 + k - 1] = q_m \quad | \quad q[n_1, n_2] = q_i)$$

$$= \quad \pi_i \pi_j \sum_{l,m=1}^{N} \mathbf{E}\left[x[n_1, n_2]x^T[n_1, n_2 + k - 1] \quad | \quad q[n_1 - 1, n_2] = q_l, q[n_1 - 1, n_2 + k - 1] = q_m\right]$$

$$P(q[n_1 - 1, n_2] = q_l \quad | \quad q[n_1, n_2] = q_i) P(q[n_1 - 1, n_2 + k - 1] = q_m)$$

$$= \quad \pi_j \sum_{l,m=1}^{N} q_{li} \overline{P}_{k+1}^{l,m}$$

The fifth expectation can be simplified as shown below: The sixth, eight, and ninth expectations have similar deriviations and aren't shown,

$$\pi_i \pi_j \quad \mathbf{E}\left[x[n_1, n_2 - 1]x^T[n_1, n_2 + k - 1]|q[n_1, n_2] = q_i, q[n_1, n_2 + k] = q_j\right]$$

$$= \quad \pi_i \pi_j \mathbf{E}\left[x[n_1, n_2 - 1]x^T[n_1, n_2 + k - 1]\right]$$

$$= \quad \pi_i \pi_j \sum_{l,m=1}^{N} \overline{P}_{k}^{l,m}$$

With these simplifications inserted into equation 10, we obtain the third equation in theorem 4.2. A similar derivation can be used to obtain the first two equations. ●

References

[D'An99] R. D'Andrea (1999). Linear Matrix Inequalities, Multidimensional System Optimization, and Control of Spatially Distributed Systems: An Example. In Proceedings, American Control Conference, pages 2713-2717, 1999.

[Fra01] Frampton, K.D (2001), Decentralized Control of Structural Acoustic Radiation, Proceedings of Symposium on Noise Control and Acoustics IMECE 2001, Noise Control and Acoustics Division, NY, NY, November 2001

[Had02] C.N. Hadjicostis and R. Touri (2002), "Feedback control utilizing packet dropping network links", IEEE Conference on Decision and Control, Las Vegas, USA, Dec. 2002.

[Lin02]   Q. Ling and M.D. Lemmon (2002), "Robust Performance of Soft Real-time Networked Control Systems with Data Dropouts", IEEE Conference on Decision and Control, Las Vegas, 2002.

[Lin03]   Q. Ling and M.D. Lemmon (2003), Soft real-time scheduling of networked control systems with dropouts governed by a Markov chain, American Control Conference, Denver, Colorado, June 2003.

[Mar90]   M. Mariton (1990), *Jump Linear Systems in Automatic Control*, Marcel Dekker Inc., 1990.

[Nil98]   J. Nilsson (1998), *Real-time control systems with delays*, Ph.D. thesis, Lund Institute of Technology, 1998.

[Ram99]   P. Ramanathan (1999), "Overload management in real-time control applications using (m, k)-firm guarantee", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, no. 6, pp. 549-559, June 1999.

[Sei01]   P. Seiler (2001), "Coordinated Control of Unmanned Aerial Vehicles", Ph.D. Thesis, University of California Berkeley, Mechanical Engineering, Fall 2001.

[Zha01]   W. Zhang (2001), *Stability analysis of networked control systems*, Ph.D. Thesis, Case Western Reserve University, 2001.