Reputation-Based Networked Control with Data-Corrupting Channels

Shreyas Sundaram
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
ssundara@uwaterloo.ca

Jian Chang
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
jianchan@cis.upenn.edu

Krishna K. Venkatasubramanian
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
vkris@cis.upenn.edu

Chinwendu Enyioha
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA
cenyioha@ee.upenn.edu

Insup Lee
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
lee@cis.upenn.edu

George J. Pappas
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA
pappasg@ee.upenn.edu

ABSTRACT

We examine the problem of reliable networked control when the communication channel between the controller and the actuator periodically drops packets and is faulty *i.e.,* corrupts/alters data. We first examine the use of a standard triple modular redundancy scheme (where the control input is sent via three independent channels) with majority voting to achieve mean square stability. While such a scheme is able to tolerate a single faulty channel when there are no packet drops, we show that the presence of lossy channels prevents a simple majority-voting approach from stabilizing the system. Moreover, the number of redundant channels that are required in order to maintain stability under majority voting increases with the probability of packet drops. We then propose the use of a reputation management scheme to overcome this problem, where each channel is assigned a reputation score that predicts its potential accuracy based on its past behavior. The reputation system builds on the majority voting scheme and improves the overall probability of applying correct (stabilizing) inputs to the system. Finally, we provide analytical conditions on the probabilities of packet drops and corrupted control inputs under which mean square stability can be maintained, generalizing existing results on stabilization under packet drops.

Categories and Subject Descriptors

B.4.5 [**Reliability, Testing, and Fault-Tolerance**]: Redundant Design; G.1.0 [**Numerical Analysis**]: General—*Stability (and instability)*; K.6.m [**Miscellaneous**]: Security

General Terms

Reliability, Security, Design, Algorithms, Theory

Keywords

Networked Control, Reputation, Majority Voting, Stability Analysis

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
HSCC'11, April 12–14, 2011, Chicago, Illinois, USA.
Copyright 2011 ACM 978-1-4503-0629-4/11/04 ...$10.00.

1. INTRODUCTION

Networked control systems are spatially distributed systems where the communication between the sensors, actuators and controllers takes place through a network [12]. The presence of a network within the control loop can adversely affect the performance of the system due to the inherent unreliability of the underlying channel. For instance, the network can drop packets with a certain probability or introduce delays in transmission. Recent years have seen much work on the problem of control in the presence of such imperfections [9, 21, 8, 13, 10, 19, 2].

While the issue of stability over relatively benign packet-dropping channels has been well studied, the topic of maintaining stability under faults or attacks (*i.e.,* data alterations or corruptions) in the *network* has not yet received much attention. The potential for such disruptions is becoming greater as control networks become increasingly integrated with standard corporate and data networks [23], and an unmitigated fault or attack could have disastrous consequences in safety-critical applications. The paper [1] takes a step towards addressing this problem, studying the stability of networked control systems under a malicious denial-of-service attack where an attacker stops packets from reaching the controller or actuator for an extended period of time. Other recent investigations of this topic can be found in [20].

In this work, we study the problem of reliable networked control with channels that are both packet dropping and data-corrupting. We first examine the use of a simple majority voting scheme with multiple redundant channels between the controller and the actuator. Each of these channels may drop packets or modify the data that they carry. We show that a straightforward implementation of triple modular redundancy *may not* be sufficient to ensure stability, due to

the presence of packet drops in the network. Specifically, we show that the number of redundant channels required in order to ensure stability increases with the probability of packet drops by each channel. It is worth noting that [19] also studied the use of multiple packet-dropping channels to obtain stability in a networked control system. However, the channels in that paper were not assumed to corrupt the data, and redundancy was added only to increase the probability of receiving a packet at the actuator. In contrast, we introduce redundant channels in our setup only to deal with data-corruptions in certain channels, and show that the combination of data corruptions and packet drops imposes a lower bound on the number of redundant channels that are required to stabilize the system (even if a single channel is sufficient to stabilize the system under non-faulty conditions).

To address the problem of stabilization under packet drops and data corruptions, we introduce the use of *reputation management* into the networked control setting. At its core, this involves placing a computational element (called the *reputation manager*) in the feedback loop, which examines the values that it receives from the redundant channels and uses the *history* of the channels' (data corruption) behaviors to assign to them a quantitative reputation 'score'. This score is then used to switch between the available channels (or to apply no input at all). Reputation management schemes have been well studied in the computer science community for the past decade for applications such as file sharing [24], peer-to-peer networks [17], spam detection [25], and inter-domain routing [6]. The characterization of "good" versus "bad" behavior with regard to maintaining reputation depends on the particular application and domain. While reputation management systems have traditionally been used to characterize and control virtual processes in the computing domain, in this work, we apply the concept to a *cyber-physical* system revolving around the control of a plant. Consequently, we tie the performance of the reputation manager to a *physical process*, namely the stability of the plant (encapsulated by the square of the largest eigenvalue of the system matrix, as we will show).

The principal contributions of the paper are: (1) an analytical characterization of the probabilities of packet drops and data corruptions under which the networked control system can maintain mean-square-stability, and (2) a demonstration of the capability of reputation management to improve upon triple modular redundancy schemes to provide mean square stability (under certain conditions). In addition to the specific contributions listed above, our analysis and evaluations are intended to lay a foundation for further integration of reliability and security mechanisms developed by the computer science community into feedback control settings.

The paper is organized as follows, Section 2 presents the necessary background on networked control systems and the fault model for this work. This is followed by Section 3, where we describe the system model for our reliable control scheme, and Section 4 delves into the majority voting approach and its pitfalls. Section 5 presents the reputation management scheme. In Section 6, we present a detailed stability analysis for systems with a faulty channel. Finally, Section 7 concludes the paper.

Notation: For a given matrix $\mathbf{A}$, the notation $\mathbf{A}'$ indicates the transpose of the matrix. For any vector $\mathbf{a}$, the notation $\|\mathbf{a}\|$ denotes the Euclidean norm of the vector. More generally, for a given positive definite matrix $\mathbf{P}$ and any vector $\mathbf{a}$, the notation $\|\mathbf{a}\|_{\mathbf{P}}^2$ denotes $\mathbf{a}'\mathbf{P}\mathbf{a}$. The eigenvalues of maximum and minimum modulus of a matrix $\mathbf{A}$ are denoted by $\lambda_{max}(\mathbf{A})$ and $\lambda_{min}(\mathbf{A})$, respectively. The notation $E[\cdot]$ denotes the expectation of a random variable. The notation $\mathbb{N}$ denotes the set of nonnegative integers.

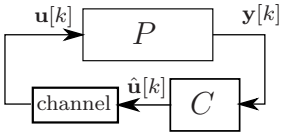


Figure 1: Networked Control

2. BACKGROUND AND FAULT MODEL

Consider the networked control system shown in Fig. 1. The plant P is given by the dynamical system

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \;\; \mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] \;\; , \qquad (1)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state vector, $\mathbf{u} \in \mathbb{R}^m$ is the system input vector, $\mathbf{y} \in \mathbb{R}^p$ is the system output vector, and k denotes the time-step of the system. The matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are real-valued matrices of appropriate dimensions.

To obtain the desired behavior from the plant, the output $\mathbf{y}[k]$ is sent to a controller C, as shown in Fig. 1. Based on $\mathbf{y}[k]$ (or perhaps the entire history of the outputs), the controller computes an input $\hat{\mathbf{u}}[k]$ to apply to the plant. This value is sent through a channel (or more generally, a network), which results in a value $\mathbf{u}[k]$ to be applied at the actuators. There are multiple ways to model the channel; for example, it may drop packets according to some probability, introduce time-varying delays, or have dynamics that cause the input and output to be related in complex ways. For the Bernoulli packet-drop model commonly studied in the literature (*e.g.,* [12, 10, 22]), at each time-step k, we have

$$\mathbf{u}[k] = \begin{cases} \hat{\mathbf{u}}[k] & \text{with probability } 1-p, \\ \emptyset & \text{with probability } p, \end{cases}$$

where $\emptyset$ denotes that no signal is received. In the latter case, we assume that the actuators simply apply the input $\mathbf{u}[k] = 0$. This case has been studied extensively, leading to conditions on the plant and success probability p under which the system will be stable. Due to the probabilistic nature of the applied inputs, the following notion of stability is considered in the literature.

Definition 1. The system is said to be *mean square stable* if $E\left[\|x[k]\|^2\right] < \infty, \;\; \forall k \in \mathbb{N}$.

Various conditions for ensuring mean square stability have been obtained for general plants [13, 21, 8]. When the input matrix $\mathbf{B}$ is square and full rank (*i.e.,* the system is fully actuated), the following result has been established.

THEOREM 1 ([9, 13, 10]). *Suppose that $\mathbf{B}$ is square and full rank, and let p be the packet drop probability. Then, there exists a linear controller C such that the closed loop system is stable if and only if $p|\lambda_{max}(A)|^2 < 1$.*

EXAMPLE 1. *Consider the first-order plant*

$$x[k+1] = 4x[k] + u[k], \;\; y[k] = x[k] \;\; ,$$

where $x, u, y \in \mathbb{R}$. Suppose we wish to control this plant via the standard networked control architecture shown in Fig. 1,

where the channel has drop probability $p = 0.05$. Since $p|\lambda_{max}(A)|^2 = 0.05(4)^2 = 0.8 < 1$, one can find a controller (e.g., $u[k] = -4y[k]$) such that the system is mean square stable.

In this paper, we expand the discussion on networked control systems to the case where the channel is capable of *modifying* the data that it carries, in addition to dropping it with a certain probability. This can happen, for example, if an attacker takes control of one of the intermediate nodes in the network, and executes *man-in-the-middle* attacks [3]. It can also happen due to accidental faults in the network (*e.g.,* when encoding or decoding during transmissions). Regardless of the cause of the error, we will refer to channels that modify the data that they carry as faulty.

Definition 2. Let $\hat{\mathbf{u}}[k]$ be the input to the channel at time-step k, and let $\mathbf{u}[k]$ be the output of the channel. The channel is said to be *faulty* at time-step k if the channel outputs a value that is not equal to the input (*i.e.,* $\mathbf{u}[k] \neq \hat{\mathbf{u}}[k]$ and $\mathbf{u}[k] \neq \emptyset$).

REMARK 1. *Note that the above definition allows the faulty channel to change the control input arbitrarily. Also, note that if the channel drops the packet (*i.e., $\mathbf{u}[k] = \emptyset$*), it is irrelevant whether or not the data was modified en-route; thus, we will use the term faulty to refer only to channels that actually deliver a modified packet at a given time-step k.*

REMARK 2. *In this paper, we will make the assumption that the controller is located at the sensor (without an intermediate channel), and that the controller has full access to the state (*i.e., $\mathbf{y}[k] = \mathbf{x}[k]$*). The latter assumption can be relaxed if there is an acknowledgment mechanism in the network, whereby the actuator can inform the controller of the input value that is applied [13]. We will leave the full treatment of more general scenarios for future work; as we will see, even this scenario offers various challenges for control.*

REMARK 3. *One can readily incorporate random noise into system* (1) *without affecting the property of mean square stability that we discuss in this paper. In order to avoid introducing additional variables and to keep the exposition clear, we will leave noise out of our discussion.*

REMARK 4. *One can also consider the inclusion of cryptographic mechanisms into the feedback loop to protect the data that is being transmitted over the channels. However, such mechanisms are primarily intended to protect the confidentiality of the data, and do not typically address the issue of what happens when the entire data packet is corrupted (i.e., even if the actuator is able to determine that the integrity of the transmitted data is compromised, it has no way to know what input to apply to the system). In other words, the real-time nature of feedback control requires a mechanism by which data is reliably delivered. As we will see in the next section, we will achieve this by transmitting the control inputs along multiple independent channels, and enforce the assumption that the packets on at most one of the channels can be corrupted by the attacker. In return, we do not have to assume that the attacker is computationally bounded (which is a typical assumption in cryptographic systems). This assumption of an attacker restricted by the topology of the network, as opposed to bounded in the computations that he/she can perform, is also considered in the literature on Byzantine fault tolerance [18] and information theoretic security [15].*

3. NETWORKED CONTROLLER MODEL AND ASSUMPTIONS

As a first step to addressing the problem of a faulty channel in Fig. 1, suppose we implement a standard triple modular redundancy scheme by simply sending the controller inputs through three different channels; for example, this could represent disjoint paths through the communication network between the controller and the actuator. Such an implementation is shown in Fig. 2.

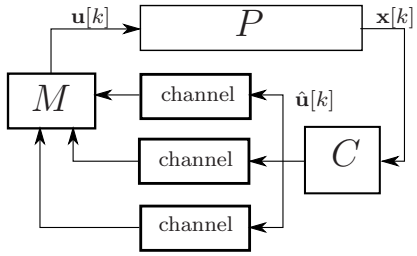


Figure 2: Networked Control with Redundant Channels.

The *manager* block (denoted by M in Fig. 2) compares the different channel values and makes a decision as to what input $\mathbf{u}[k]$ (if any) to apply (i.e., it switches between the available channels, or the input $\mathbf{0}$). We assume that each channel drops its packet with probability p, and that during the course of operation, no more than one[1] of the channels can be faulty (i.e., if a channel is faulty at some time-step, a different channel cannot be faulty at another time-step). We also assume that each channel is independent of the other channels. If the probability of packet drops is $p = 0$, the manager M receives the outputs of all three channels at each time-step, and can simply choose the value that is specified by the majority of the channels as the input $\mathbf{u}[k]$. The more interesting case occurs when we have a nonzero probability of packet drop, and the manager must adopt some rule to choose which input to apply among the ones it does receive. We study this case next.

4. MAJORITY VOTING WITH PACKET DROPS

When the manager does not receive all three signals at each time-step, one can adopt the following natural extension of the standard majority-voting mechanism: if the manager receives at least two signals that match, then that signal is applied. Otherwise, the input $\mathbf{u}[k] = \mathbf{0}$ is applied. We denote the number of received signals at time-step k by R_k. Note that R_k is a binomial random variable with parameters $(3, p)$. Thus, $R_k = t$ with probability $\binom{3}{t}p^{3-t}(1-p)^t$, for $t \in \{0, 1, 2, 3\}$. The probability of receiving two matching signals will depend on whether one of the channels is *faulty.*

4.1 No Faulty Channels

When there are no faulty channels, the correct input is applied whenever at least two signals are received, and $\mathbf{u}[k] = 0$ is applied otherwise. This latter case occurs with probability $\bar{p}_1 = p^3 + 3p^2(1-p)$. One can verify that $\bar{p}_1 \leq p$ for $p \in [0, 0.5]$, and $\bar{p}_1 \geq p$ for $p \in [0.5, 1]$. Thus, when there are

[1]In general, if f faulty/malicious channels are to be tolerated, one requires $2f + 1$ channels in total.

no faulty channels, this mechanism is equivalent to a single non-faulty channel with drop probability $\bar{p}_1$; there will be an overall improvement when p is less than 0.5, and a degradation when p is greater than 0.5.

4.2 One Faulty Channel

When one of the channels is faulty (*i.e.,* its output is not equal to the signal generated by the controller), the probability that the manager receives at least two matching signals is equal to the probability that it receives signals from both of the non-faulty channels. This probability is given by $(1-p)^2$, and thus the drop probability is given by

$$\bar{p}_2 = 1-(1-p)^2 = -p^2+2p. \tag{2}$$

One can verify that $\bar{p}_2 \geq p$ for $p \in [0,1]$. The presence of a faulty channel therefore results in an overall degradation in performance (*i.e.,* the majority voting mechanism functions as a single non-faulty channel with drop probability $\bar{p}_2 \geq p$). This may result in the overall system no longer being stable, as the following example illustrates.

EXAMPLE 2. *Suppose we would like to protect the plant described in Example 1 against channel faults (or attacks) by implementing the modular redundancy scheme shown in Fig. 2. If one of the channels is faulty, the probability that the manager M does not receive two equal values (equal to $-4y[k]$) at any time-step k is given by equation* (2)*, namely* $\bar{p}_2 = -(0.05)^2+2(0.05) = 0.0975$. *However,* $\bar{p}_2|\lambda_{max}(A)|^2 = 0.0975(4)^2 = 1.56 > 1$, *and this violates the bound in Theorem 1. Thus the plant cannot be stabilized via this majority voting scheme when one of the channels is faulty.*

The above example shows that when one considers the possibility of both faulty and packet-dropping channels, a simple triple-modular redundancy scheme with majority voting may not be sufficient to maintain stability, *even* when the probability of packet drop is low enough that a single non-faulty channel with the same drop probability would suffice.

One way to restore the performance of the networked control scheme would be to increase the number of redundant channels. Specifically, suppose that we consider $N+1$ channels, where one channel is allowed to be faulty. Now, the probability that the manager does not get two (non-empty) matching signals is:

$$\bar{p} = p^N + Np^{N-1}(1-p) \ . \tag{3}$$

For any desired drop probability p^*, we would like to find the value of N for which $\bar{p} = p^*$ (*i.e.,* the number of channels that are required in order to obtain the desired performance). This is given by the following theorem.

THEOREM 2. *Consider a majority voting scheme with $N+1$ channels, each of which can drop packets with a probability p. Furthermore, suppose up to one of the channels can be faulty, whereby it changes the value of the data that it carries. The number of redundant channels N required to maintain a drop probability of p^* is given by*

$$N = \frac{1}{\ln p} W\left(\frac{p^* \ln p}{1-p} p^{\frac{1}{1-p}}\right) - \frac{p}{1-p},$$

where $W(\cdot)$ is the Lambert W-function.[2]

[2]This is the inverse of the function xe^x, and cannot be written in terms of more elementary functions [7].

PROOF. In order to have $\bar{p} = p^*$, we use (3) to obtain

$$\begin{aligned}
&p^N + Np^{N-1}(1-p) = p^* \\
\Leftrightarrow &(p+N(1-p))p^{N-1} = p^* \\
\Leftrightarrow &(p+N(1-p))e^{(N-1)\ln p} = p^* \\
\Leftrightarrow &(1-p)\left((N-1)+\frac{1}{1-p}\right)e^{(N-1)\ln p} = p^* \\
\Leftrightarrow &\left((N-1)+\frac{1}{1-p}\right)(\ln p)e^{\left((N-1)+\frac{1}{1-p}\right)\ln p} = \frac{p^* \ln p}{1-p} e^{\frac{\ln p}{1-p}}.
\end{aligned}$$

Letting $f(x) = xe^x$, the above equation becomes

$$f\left((N-1)\ln p + \frac{\ln p}{1-p}\right) = \frac{p^* \ln p}{1-p} p^{\frac{1}{1-p}} \ .$$

The solution to the above equation is given by the expression in the theorem. □

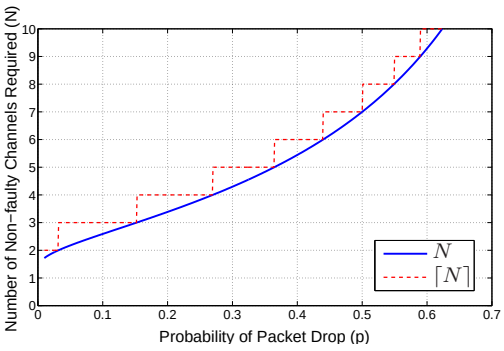


Figure 3: Number of non-faulty channels (N) required in a majority voting scheme in order to tolerate a single faulty channel and maintain a drop probability of $p^* = \frac{1}{4^2}$.

A plot of N versus p is shown in Fig. 3 for the case where $p^* = \frac{1}{4^2}$ (i.e., this is the maximum drop probability able to stabilize the scalar plant in Example 1); since the number of channels must be integer-valued, the value of $\lceil N \rceil$ is also shown in the plot. Note that for $p = \frac{1}{4^2}$, one requires $N = 3$ additional channels (for a total of four channels) in order to tolerate one faulty channel and still maintain the same probability of packet drop as a single non-faulty channel. As the packet drop probability increases, the number of channels required also increases rapidly.[3] Note that, even though we assume that at most one channel is faulty over all times, the analysis presented in this section will hold for the more general case where different channels can be faulty at different time-steps (as long as at most one channel is faulty in any given time-step). This is because the above majority voting scheme is memoryless, and decisions in the present are made without regard to past behavior of the channels.

In the next section, we will present a mechanism to address the shortcomings of modular redundancy in control loops with packet drops. For certain types of faults that afflict the channels, this mechanism will provide mean-square

[3]This result can potentially be used by an intelligent attacker to execute a subtle "jamming" or denial-of-service attack on the network, whereby he or she increases the rate of packet drops to the point where four disjoint channels are no longer sufficient to maintain stability.

stability without requiring the use of more than two redundant channels *i.e.,* it will address the problems described above with regard to the loss of stability in triple modular redundancy.

5. MAJORITY VOTING WITH A REPUTATION MANAGER

In practice, the behavior of a channel at any time-step k will be correlated with past and future behavior. For example, if a channel is compromised by an attacker, it is likely that the channel will misbehave for several time-steps (or perhaps permanently). Similarly, if the channel has an internal state that gets updated at each time-step, and if the state gets disturbed from its normal operating condition for some reason, it will take several time-steps for this error to die away. To formally capture such behavior, we will first need a metric to evaluate "good" versus "bad" behavior by a channel. Let $C = \{c_1, c_2, c_3\}$ be the set of channels between the controller and reputation manager and $I_{c_i,k}$ be the input received from a channel $c_i \in C$ at time-step k (this value can be $\emptyset$ if the input is dropped by the channel during the time-step).

Definition 3. An input $I_{c_i,k}$ is *verifiably correct* if the reputation manager receives at least one additional signal from any channel $c_j \in C$, $i \neq j$ and $I_{c_i,k} = I_{c_j,k} \neq \emptyset$. An input $I_{c_u,k}$ is *verifiably incorrect*, if the reputation manager receives two other signals c_i and c_j, such that $c_i, c_j \in C$, $i \neq j \neq u$, $I_{c_i,k} = I_{c_j,k} \neq \emptyset$ and $I_{c_u,k} \neq I_{c_i,k}$.

Let $G_{c_i,k}$ and $B_{c_i,k}$ represent the sets of verifiably correct and verifiably incorrect inputs, respectively, received from a channel $c_i \in C$ up to time-step k.

Definition 4. The reputation of a channel $c_i \in C$ at a time-step k is a map $Rep : G_{c_i,k} \times B_{c_i,k} \rightarrow [0, 1] \cup \{\phi\}$, where ϕ denotes an *unknown* reputation.

To simplify the notation, we use $Rep(c_i)$ to denote the reputation of a channel c_i at a given time-step. The exact map that is used will depend on the types of faulty behavior that are assumed. The reputation values for each channel will be administered and updated by a *reputation manager* (RM), which we describe next.

5.1 Reputation Manager

The reputation manager is incorporated into the block labeled M in Fig. 2, and operates in four stages: (1) observe the inputs from the channels at a given time-step, (2) verify the correctness (or incorrectness) of the received inputs, (3) update the reputation of the channels whose inputs can be verified, and (4) use the reputation to choose between the available channels (or apply no input at all). As we shall see, the use of reputation management outperforms the majority voting mechanism by increasing the probability of applying a correct control input and reducing the probability of applying no input.

When there is only one faulty channel and the RM receives at least two matching signals, it is *certain* that the input is correct. However, there may be situations when the RM is *uncertain* about the quality of the signal; this can happen, for example, if two non-matching signals are received, and the reputations of the two channels are identical. If the RM randomly chooses one of the two inputs to apply, there is a chance that an *incorrect* value will be applied. It would be prudent in this case to ensure that the input applied by the RM is *bounded* in some sense, so that a faulty channel is not able to immediately increase the system state to a large value. Rather than having the RM scale the received inputs itself, we will instead have the controller inject *two* values into each channel: one value is a state feedback input $-\mathbf{Kx}[k]$, and the other is a *bounded* input $\mathbf{u}^b[k]$ satisfying $\|\mathbf{u}^b[k]\|_{\mathbf{P}} < b$, for some matrix $\mathbf{K}$, some positive definite matrix $\mathbf{P}$ and some positive real number b. We will later discuss conditions under which $\mathbf{K}, \mathbf{P}, b$ and $\mathbf{u}^b[k]$ can be chosen to ensure mean square stability, even when bounded incorrect inputs are applied with a certain probability. Note that a faulty channel can corrupt both the state feedback and bounded input injected by the controllers, but if the corruption causes the norm of the bounded input to increase above b, this can be immediately detected and identified by the reputation manager.

Algorithm 1 Reputation Management Scheme

Require: Controller injects $-\mathbf{Kx}[k]$ and $\mathbf{u}^b[k]$ into each of the three channels. Both of these signals travel together in one packet. Let R_k denote the number of packets received by the RM at time-step k.

```
begin
if R_k = 3 then
    RM applies the value −Kx[k] specified by the majority of the packets, and updates the reputation of all three channels accordingly
else if R_k = 2 and they both match then
    RM applies −Kx[k] and updates the reputation of the two channels.
else if R_k = 2 and they do not match then
    if Rep(c_i) ≥ Θ and 0 ≤ Rep(c_j) < Θ then
        RM applies the value −Kx[k] specified by c_i, and increases the reputation of c_i while decreasing that of c_j.
    else if Rep(c_i) ≥ Θ and Rep(c_j) ≥ Θ then
        RM randomly chooses one of the two channels and applies the input u^b[k] specified by it.
    else if Rep(c_i) ≥ Θ and Rep(c_j) = φ then
        RM applies u^b[k] specified by c_i.
    else if (Rep(c_i) = φ and Rep(c_j) = φ) or (Rep(c_i) < Θ and Rep(c_j) < Θ) then
        RM applies u[k] = 0
    end if
else if R_k = 1 then
    if Rep(c_i) ≥ Θ then
        RM applies u^b[k] specified by c_i.
    else if Rep(c_i) = φ or Rep(c_i) < Θ then
        RM applies u[k] = 0
    end if
else if R_k = 0 then
    RM applies u[k] = 0
end if
end
```

The above ideas are formally presented as Algorithm 1. Here, i and j are indices of channels from which input is received. The reputation value for each channel is initially set to ϕ (*i.e.,* unknown), and gets updated as verifiably correct or incorrect inputs are received from the channels. The parameter $\Theta \in [0, 1]$ is a *threshold* value, and input from a channel with reputation below Θ is not applied. A variety of mappings/functions can be used to obtain the actual value of the reputation $Rep(c_i)$ based on the observed history $H_{c_i,k} = G_{c_i,k} \cup B_{c_i,k} \cup (\cup_{g=1}^{k} I_{c_i,g} - \{G_{c_i,k} \cup B_{c_i,k}\})$ at any time-step k. In this section we describe two such mappings that prove to be simple yet more effective than majority voting. Before delving into the details, we present the fault model that we are considering (the analysis of other fault models will be left as an extension for future work).

5.1.1 Fault Model

We assume that, over all time-steps, at most one channel is faulty, and without loss of generality we take this to be c_3 (of course, this is unknown to the manager).[4] Furthermore, we assume that c_3 becomes faulty at an arbitrary time-step. In this work, we take the faultiness of the channel to be probabilistic and expressed as d_{c_i}, which is the probability that channel c_i is *fault-free* at any given time-step. Thus, we consider the case where $d_{c_1} = 1$, $d_{c_2} = 1$ and $0 \leq d_{c_3} \leq 1$ at any given time-step. Finally, we assume that the reputation manager knows that at most one channel is faulty, but it does not know *a priori* that c_3 is the faulty channel nor does it know the probability of being c_3 being faulty (*i.e.*, $1-d_{c_3}$).

5.2 Stratified Reputation

We will start by considering a simple reputation function that switches between a finite set of reputation values. Initially all the channels have an unknown reputation ϕ.

Definition 5. (Stratified Reputation) Let $G_{c_i,k}$ and $B_{c_i,k}$ represent the number of verifiably correct and verifiably incorrect inputs, respectively, received from channel c_i up to time-step k. Then, the reputation of the channel c_i at time-step k is defined as:

$$Rep(c_i) = \begin{cases} 1 & G_{c_i,k} > 0 \ \& \ B_{c_i,k} = 0, \\ 0 & B_{c_i,k} > 0, \\ \phi & B_{c_i,k} = 0 \text{ and } G_{c_i,k} = 0. \end{cases}$$

Using the stratified reputation function, the moment we verify that a channel is faulty, it is "black-listed" and no unbounded input provided by that channel is ever applied again. Since reputations can only be either 0 or 1 (when not unknown), we can use any $0 < \Theta \leq 1$ as the threshold in Algorithm 1. While simple, this reputation function does not provide us with much information about the extent of faultiness of the channel (which could be useful for identifying the cause of the fault). We will next consider a slightly more complex reputation function that still provides an improvement over majority voting, and also provides an estimate of the fault probability of the channel.

5.3 Bayesian Reputation

To estimate the fault probability of the channel, we use a *Bayesian Reputation* function [14]. To understand this, note that when channel c_i becomes faulty, each verifiable input provided by c_i is correct or incorrect with a Bernoulli distribution, given by the (unknown) probability d_{c_i}. Consequently, given a set of verifiably correct and incorrect inputs from a channel, the quantity d_{c_i} satisfies a *beta distribution*, which is the probability distribution of seeing a particular combination of correct and incorrect inputs from a channel [14]. The expected value of the beta distribution forms the reputation and is given by the ratio of the number of correct values received to the number of total values received. This is also the required estimate of the fault-free probability of c_3 (*i.e.*, d_{c_3}).

Definition 6. (Bayesian Reputation) Let $G_{c_i,k}$, $B_{c_i,k}$ and $S_{c_i,k}$ represent the number of verifiably correct, verifiably incorrect and total number of verifiable inputs, respectively, received from the channel c_i up to time-step k. Then, the reputation of the channel c_i at time-step k is defined as:

$$Rep(c_i) = \frac{G_{c_i,k}}{S_{c_i,k}}, where \ \ S_{c_i,k} = G_{c_i,k} + B_{c_i,k}. \quad (4)$$

Note that, it might be preferable in some situations to give weight to more recent behaviors than older ones, *e.g.,* by introducing time decay functions for the reputation calculation. We will leave a treatment of such reputation functions for future work.

5.3.1 Discussion

Even though both reputation functions can detect the faulty channel, the Bayesian reputation function allows one to estimate the probability of the channel's faultiness through its reputation value. This helps system administrators perform better fault diagnosis. For example, if the reputation of channel c_i is lower than a certain value, it might imply that the root cause of the faulty signal could be serious network outage or a malicious attack, rather than random network operation errors. Consequently, unlike the stratified function, with the Bayesian reputation function, the reputation value is constantly updated even when the RM is aware which channel might be faulty.

Note that, these two are not the only reputation functions that can be used here. Their choice is motivated by the desire to demonstrate simple functions that can provide improvements over majority voting (as we will show below). The choice of the reputation function is largely dictated by the fault model assumed in the system. Identifying different criteria for choosing appropriate reputation functions is the focus of ongoing research.

5.4 Evaluation Metrics

From Algorithm 1 it can be seen that each of the inputs (i.e., a full state-feedback input, a bounded correct input, a bounded faulty input, and $\mathbf{0}$) will be applied with a certain probability. We will characterize the probabilities of each of these events by $\mathcal{C}, \mathcal{C}_b, \mathcal{E}$ and $\mathcal{N} = 1-\mathcal{C}-\mathcal{C}_b-\mathcal{E}$, respectively. In the rest of the paper we refer to these collectively as ECN metrics. Furthermore, we use subscripts NR and R with the ECN metrics, to indicate that the scenario being considered *does not* or *does* use reputation, respectively.

5.4.1 Performance Without Reputation Manager

In the absence of a reputation manager (*i.e.*, the standard triple-modular-redundancy scheme with majority voting from the previous section), it is easy to see that $\mathcal{E}_{NR} = 0$, $\mathcal{C}_{b_{NR}} = 0$, $\mathcal{C}_{NR} = (1-p)^2 + 2p(1-p)^2 d_{c_3}$ and $\mathcal{N}_{NR} = 1 - [(1-p)^2 + 2p(1-p)^2 d_{c_3}]$.

5.4.2 Performance With Reputation Manager

Considering our setup with reputation, Table 1 shows the ECN metrics for different combinations of possible channel reputation values. We omit the value of $\mathcal{C}_R$, since $\mathcal{C}_R = 1 - \mathcal{C}_{b_R} - \mathcal{E}_R - \mathcal{N}_R$. The following example demonstrates how the ECN probabilities are calculated.

EXAMPLE 3. *Assume initially all channels behave as expected without modifying the controller inputs. This will eventually cause the reputations of c_1, c_2 and c_3 to become 1 (*i.e., *perfect). Now suppose c_3 starts behaving in a faulty manner with a probability $1 - d_{c_3}$. As the reputation of c_3 is perfect, there is a non-zero probability of applying an incorrect input to the plant.*

[4]For instance, in a multi-hop network, the three channels could represent three node-disjoint paths in the network between the controller and the manager, and misbehavior on the part of nodes on only one of these paths would leave the other paths (channels) reliable.

Table 1: ECN metrics for different channel reputation values

Reputation of Channel			Metrics		
c_1 (Good)	c_2 (Good)	c_3 (Faulty)	$\mathcal{E}_R$	$\mathcal{N}_R$	$\mathcal{C}_{b_R}$
1	1	$\geq \Theta$	$(1-p)p(1-d_{c_3})$	p^3	$p(1-p)(1+p+d_{c_3}(2p-1))$
ϕ	1	1	$\frac{1}{2}p(1-p)(1-d_{c_3})(3-p)$	$p^3+(1-p)p^2$	$\frac{1}{2}p(1-p)(1+p+2d_{c_3})$
1	ϕ	1	$\frac{1}{2}p(1-p)(1-d_{c_3})(3-p)$	$p^3+(1-p)p^2$	$\frac{1}{2}p(1-p)(1+p+2d_{c_3})$
1	1	ϕ or $< \Theta$	0	$p^3+(1-p)p^2$	$2p(1-p)$
ϕ	ϕ	ϕ	0	$p^3+3(1-p)p^2+2(1-p)^2p(1-d_{c_3})$	0

A bounded incorrect controller input is applied to the plant under two conditions: (1) a value from c_3 alone was received, the probability of which is given by: $p^2(1-p)$, or (2) values were received from c_1 and c_3 or c_2 and c_3 and in either case, c_3 was chosen at random, given by the probability $\frac{1}{2}p(1-p)^2+\frac{1}{2}p(1-p)^2=p(1-p)^2$. Furthermore, the probability that c_3 is faulty, given that its value was received, is given by $(1-d_{c_3})$. Given these values, we have $\mathcal{E}_R=(p(1-p)^2+p^2(1-p))(1-d_{c_3})=p(1-p)(1-d_{c_3})$.

Similarly, given the perfect reputation values, no input will be applied to the plant only when no value is received, i.e., *all three channels drop their packets. The probability of this happening is given by $\mathcal{N}_R=p^3$.*

Finally, a bounded correct controller input is applied to the plant under three conditions: (1) a value from c_1 or c_2 is received, the probability of which is given by: $2p^2(1-p)$; (2) a value was received from c_1 and c_3 or c_2 and c_3, the values do not match, and c_1 and c_2 are chosen at random, respectively. The probability of this scenario is given as: $p(1-p)^2(1-d_{c_3})$; and (3) the value from c_3 alone is received and it is not faulty, which is given by $p^2(1-p)d_{c_3}$. Given these values, we have $\mathcal{C}_{b_R}=p(1-p)(1+p+d_{c_3}(2p-1))$.

Given that c_1 and c_2 are never faulty, and their reputation is updated only when majority of the inputs are identical (2 out of 3), the reputation c_1 and c_2 can therefore only take the values 1 or ϕ. In the case of channel c_3 however, the reputation can be either ϕ or $0 \leq Rep(c_3) \leq 1$ depending upon the faultiness of the channel, and the packet drop characteristics of the three channels.

5.5 Simulation Results

We simulated the triple modular redundancy scheme in a networked control system with one faulty channel. The principal goal of the simulation was to evaluate whether the reputation manager improves the probability of applying correct inputs to the plant. Each simulation run was set to execute for 1000 time-steps with different p and d_{c_3} values. Furthermore, in order to compensate for the variation of individual simulation cycles, the entire simulation process was executed $10,000$ times for each combination of p and d_{c_3} values and the averages of the values are reported. Our simulations use the Bayesian reputation function with the threshold value of $\Theta=1$. However, with this choice, one can verify that Algorithm 1 performs identically under both the Stratified and Bayesian reputation functions.

Fig. 4 shows the variation in the probability of applying a correct state-feedback input ($\mathcal{C}$). The probability of applying a correct input to the plant when reputation is used is always higher than when it is not used (*i.e.*, in the case where majority voting alone is used). As the packet drop rate increases, however, this difference is reduced considerably. Note that all of the curves corresponding to the cases where reputation is used are almost identical, which demonstrates an important property of using reputation: performance does not significant degrade even when the value of

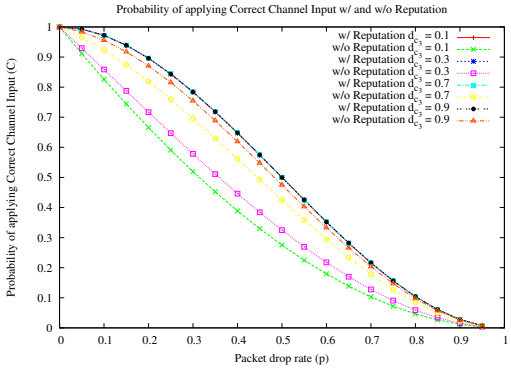


Figure 4: Effect on $\mathcal{C}$ in a system with dynamic channel failure due to reputation

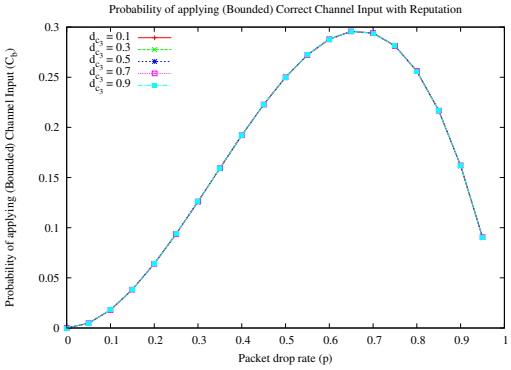


Figure 5: Effect on $\mathcal{C}_b$ in a system with dynamic channel failure due to reputation

d_{c_3} drops. This is a consequence of our fault model, where at most one of the three channels is faulty. Consequently, once the RM has identified the good channels, it automatically does not consider the input from the third one (given the chosen threshold value of $\Theta=1$).

Fig. 5 shows the probability of applying a bounded correct controller input ($\mathcal{C}_b$). At lower packet drop rates, the probability of applying a full (correct) state-feedback input to the plant is much higher, resulting in low $\mathcal{C}_b$ values. As the drop rate increases, not all inputs arrive at the reputation manager. Consequently, the reputation manager has to increasingly apply bounded inputs, resulting in the increase in the value of $\mathcal{C}_b$. As the packet drop rate increases further, not enough inputs arrive which causes $\mathcal{C}_b$ to drop again. Again, all curves are nearly identical.

Fig. 6 shows the probability of applying no controller input ($\mathcal{N}$). This value is always lower when reputation is used, but predictably increases with the packet drop rate. Once again, all curves corresponding to the cases where reputation is used are almost identical.

Fig. 7 shows the probability of applying a bounded incorrect input ($\mathcal{E}$). The values are very close to zero when the

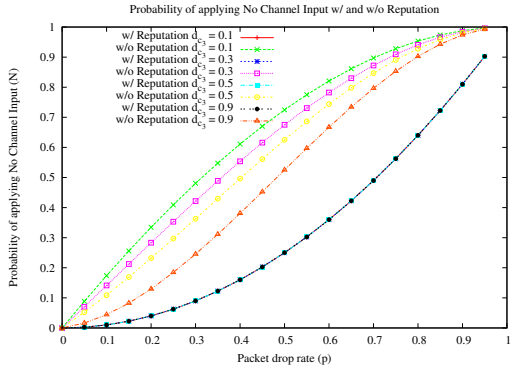


Figure 6: Effect on $\mathcal{N}$ in a system with dynamic channel failure due to reputation

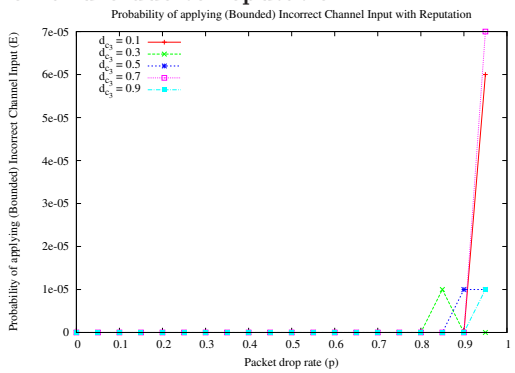


Figure 7: Effect on $\mathcal{E}$ in a system with dynamic channel failure due to reputation

packet drop rate is low, irrespective of the value of d_{c_3}. This is because 2 or more identical inputs are frequently received, causing the reputation of c_1 and c_2 to reach 1 very quickly. For higher packet drop rates, $\mathcal{E}$ is greater for channels with higher d_{c_3}. The reasons are two-fold: (1) c_3 amasses a better reputation than in the case where d_{c_3} is lower, and (2) with the high packet drop rate, a relatively large number of time-steps will have only c_3's input showing up, potentially increasing the number of incorrect inputs applied to the plant. Note that after channel c_3 becomes faulty and the system has run for some time, the reputations reach their steady state values and the probability of applying an incorrect (bounded) input $\mathcal{E}$ is always 0. In Fig. 7, the small non-zero value of $\mathcal{E}$ is due to the few incorrect (bounded) inputs that are applied during the transient period (while the reputations are converging). The convergence rate of the reputations can potentially be analyzed by associating an appropriate Markov chain with the RM and examining its mixing time; this is the subject of ongoing research.

6. STABILITY UNDER INTERMITTENT FAULTY INPUTS

In the previous section, we showed that a reputation manager can improve the probability of applying correct inputs to the plant under a majority voting scheme with packet drops. However, in situations where the RM is uncertain about the quality of the inputs that it receives, it can potentially apply incorrect (bounded) inputs to the plant with a certain probability. While this probability is zero in steady state for the fault model (only one channel can be faulty for all time) and reputation functions that we consider in this paper, the same is not true for more general fault models (e.g., where multiple channels can be faulty). In this section, we will generalize existing results on stability under packet dropping channels to the case where incorrect (but bounded) inputs are periodically applied to the system. We will then verify that for the fault model considered in this paper, reputation management is able to satisfy the resulting conditions for mean square stability better than majority voting.

To this end, suppose that system (1) operates as follows; at each time-step k, the input is:

$$\mathbf{u}[k] = \begin{cases} -\mathbf{K}\mathbf{x}[k] & \text{with probability } \mathcal{C} \\ \mathbf{u}^b[k] & \text{with probability } \mathcal{C}_b \\ \mathbf{d}[k] & \text{with probability } \mathcal{E} \\ \mathbf{0} & \text{with probability } 1-\mathcal{C}-\mathcal{C}_b-\mathcal{E}. \end{cases} \quad (5)$$

In the above, $\mathcal{C}$, $\mathcal{C}_b$, and $\mathcal{E}$ are the probabilities of applying a stabilizing state-feedback input, a bounded stabilizing input or a bounded incorrect input, respectively. The bounds on the inputs $\mathbf{u}^b[k]$ and $\mathbf{d}[k]$ are of the form $\|\mathbf{u}^b[k]\|_{\mathbf{P}} \leq b$ and $\|\mathbf{d}[k]\|_{\mathbf{P}} \leq b$ for some positive definite matrix $\mathbf{P}$ (which we will specify below) and for some positive value b. The input $\mathbf{u}^b[k]$ is taken to be *designed* (*i.e.,* chosen to improve the stability of the system), and the input $\mathbf{d}[k]$ is a *faulty* input (perhaps chosen maliciously), satisfying the above bound. The matrix $\mathbf{K}$ will be chosen to obtain mean square stability. Under these conditions, the following theorem provides a sufficient condition for the probabilities of applying state-feedback control inputs and applying bounded faulty inputs that will maintain mean square stability of the system.

THEOREM 3. *Consider a system* $(\mathbf{A}, \mathbf{B})$*, and let* $\mathcal{C}$ *be the probability of applying the state feedback input* $-\mathbf{K}\mathbf{x}[k]$*, and* $\mathcal{E}$ *be the probability of applying an incorrect, but bounded, input to the plant. If there exists a positive definite matrix* $\mathbf{P}$ *and a matrix* $\mathbf{K}$ *such that*

$$(\mathbf{A}-\mathbf{B}\mathbf{K})'\mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K})\mathcal{C} + \mathbf{A}'\mathbf{P}\mathbf{A}(1-\mathcal{C}+\mathcal{E}) < \mathbf{P}, \quad (6)$$

the system is mean square stable.

PROOF. For system (1) with inputs (5), we will examine the quantity

$$\begin{aligned} \sigma_{k+1} &\triangleq E[\|\mathbf{x}[k+1]\|^2_{\mathbf{P}}] \\ &= E\left[E[\|\mathbf{x}[k+1]\|^2_{\mathbf{P}} \mid \mathbf{x}[k], \mathbf{d}[k], \mathbf{u}^b[k]]\right] \\ &= E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|^2_{\mathbf{P}}]\mathcal{C} + E[\|\mathbf{A}\mathbf{x}[k]+\mathbf{B}\mathbf{u}^b[k]\|^2_{\mathbf{P}}]\mathcal{C}_b \\ &\quad + E[\|\mathbf{A}\mathbf{x}[k]+\mathbf{B}\mathbf{d}[k]\|^2_{\mathbf{P}}]\mathcal{E} \\ &\quad + E[\|\mathbf{A}\mathbf{x}[k]\|^2_{\mathbf{P}}](1-\mathcal{C}-\mathcal{C}_b-\mathcal{E}). \end{aligned}$$

Now, note that $\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u}\|^2_{\mathbf{P}} = \|\mathbf{A}\mathbf{x}\|^2_{\mathbf{P}} + \|\mathbf{B}\mathbf{u}\|^2_{\mathbf{P}} + 2\mathbf{x}'\mathbf{A}'\mathbf{P}\mathbf{B}\mathbf{u}$ for any vectors $\mathbf{x}$ and $\mathbf{u}$. Substituting this into the above expression, we get

$$\begin{aligned} \sigma_{k+1} &= E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|^2_{\mathbf{P}}]\mathcal{C} + E[\|\mathbf{A}\mathbf{x}[k]\|^2_{\mathbf{P}}](1-\mathcal{C}) \\ &\quad + E[2\mathbf{x}'[k]\mathbf{A}'\mathbf{P}\mathbf{B}\mathbf{u}^b[k] + \|\mathbf{B}\mathbf{u}^b[k]\|^2_{\mathbf{P}}]\mathcal{C}_b \\ &\quad + E[2\mathbf{x}'[k]\mathbf{A}'\mathbf{P}\mathbf{B}\mathbf{d}[k] + \|\mathbf{B}\mathbf{d}[k]\|^2_{\mathbf{P}}]\mathcal{E}. \end{aligned} \quad (7)$$

Suppose now that the input $\mathbf{u}^b[k]$ is chosen to minimize[5]

$$2\mathbf{x}'[k]\mathbf{A}'\mathbf{P}\mathbf{B}\mathbf{u}^b[k] + \|\mathbf{B}\mathbf{u}^b[k]\|^2_{\mathbf{P}},$$

[5]This is a Quadratically Constrained Quadratic Program (QCQP), which can be solved numerically [4], but difficult to solve analytically.

subject to the constraint $\|\mathbf{u}^b[k]\|_{\mathbf{P}} \leq b$; note that the minimum is guaranteed to be nonpositive (since choosing $\mathbf{u}^b[k] = 0$ produces a value of 0). Furthermore, note that since $\|\mathbf{a}-\mathbf{b}\|_{\mathbf{P}}^2 = \|\mathbf{a}\|_{\mathbf{P}}^2 - 2\mathbf{a}'\mathbf{P}\mathbf{b} + \|\mathbf{b}\|_{\mathbf{P}}^2 \geq 0$ for any vectors $\mathbf{a}$ and $\mathbf{b}$, we have $E[\mathbf{a}'\mathbf{P}\mathbf{b}] \leq \frac{1}{2}(E[\|\mathbf{a}\|_{\mathbf{P}}^2] + E[\|\mathbf{b}\|_{\mathbf{P}}^2])$. Thus, we can write

$$E[2\mathbf{x}'[k]\mathbf{A}'\mathbf{P}\mathbf{B}\mathbf{d}[k]] \leq E[\|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2] + E[\|\mathbf{B}\mathbf{d}[k]\|_{\mathbf{P}}^2] \ .$$

Substituting these facts into (7), we get

$$\begin{aligned}\sigma_{k+1} &\leq E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|_{\mathbf{P}}^2]\mathcal{C} + E[\|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2](1-\mathcal{C}+\mathcal{E}) \\ &\quad + 2E[\|\mathbf{B}\mathbf{d}[k]\|_{\mathbf{P}}^2]\mathcal{E} \\ &\leq E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|_{\mathbf{P}}^2]\mathcal{C} + E[\|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2](1-\mathcal{C}+\mathcal{E}) \\ &\quad + 2\mathcal{E}\lambda_{max}(\mathbf{B}'\mathbf{P}\mathbf{B})E[\|\mathbf{d}[k]\|^2] \\ &\leq E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|_{\mathbf{P}}^2]\mathcal{C} + E[\|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2](1-\mathcal{C}+\mathcal{E}) \\ &\quad + 2\mathcal{E}\frac{\lambda_{max}(\mathbf{B}'\mathbf{P}\mathbf{B})}{\lambda_{min}(\mathbf{P})}b^2. \end{aligned} \quad (8)$$

Now, examine the quantity

$$\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|_{\mathbf{P}}^2\mathcal{C} + \|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2(1-\mathcal{C}+\mathcal{E}) =$$
$$\mathbf{x}'[k]\left((\mathbf{A}-\mathbf{B}\mathbf{K})'\mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K})\mathcal{C} + \mathbf{A}'\mathbf{P}\mathbf{A}(1-\mathcal{C}+\mathcal{E})\right)\mathbf{x}[k].$$

If we can choose the positive definite matrix $\mathbf{P}$ and matrix $\mathbf{K}$ to satisfy

$$(\mathbf{A}-\mathbf{B}\mathbf{K})'\mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K})\mathcal{C} + \mathbf{A}'\mathbf{P}\mathbf{A}(1-\mathcal{C}+\mathcal{E}) < \mathbf{P},$$

then we would obtain $E[\|(\mathbf{A}-\mathbf{B}\mathbf{K})\mathbf{x}[k]\|_{\mathbf{P}}^2\mathcal{C} + \|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2(1-\mathcal{C}+\mathcal{E})] < E[\|\mathbf{x}[k]\|_{\mathbf{P}}^2]$, and (8) would become

$$\sigma_{k+1} < \sigma_k + 2\mathcal{E}\frac{\lambda_{max}(\mathbf{B}'\mathbf{P}\mathbf{B})}{\lambda_{min}(\mathbf{P})}b^2 < \alpha\sigma_k + 2\mathcal{E}\frac{\lambda_{max}(\mathbf{B}'\mathbf{P}\mathbf{B})}{\lambda_{min}(\mathbf{P})}b^2,$$

for some $0 \leq \alpha < 1$. This signifies a stable system (provided that $b < \infty$), which proves mean square stability. □

Equation (6) in the above theorem can be readily transformed to a *linear matrix inequality* (LMI), which can then be solved to determine whether there exists a feasible pair $(\mathbf{P}, \mathbf{K})$ [4]. When the matrix $\mathbf{B}$ is square and full rank, we can easily choose $\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}$, in which case (6) becomes

$$\mathbf{A}'\mathbf{P}\mathbf{A}(1-\mathcal{C}+\mathcal{E}) - \mathbf{P} < 0. \quad (9)$$

This is a Lyapunov function, and admits a positive definite matrix $\mathbf{P}$ if and only if the matrix $\sqrt{1-\mathcal{C}+\mathcal{E}}\mathbf{A}$ has all eigenvalues inside the unit circle [11]. This immediately leads to the following result.

COROLLARY 1. *Consider a system $(\mathbf{A}, \mathbf{B})$ with matrix $\mathbf{B}$ being square and full rank. Let $\mathcal{C}$ be the probability that the input $-\mathbf{B}^{-1}\mathbf{A}\mathbf{x}[k]$ is applied to the system, and let $\mathcal{E}$ be the probability that an incorrect (but bounded) input is applied to the system. Then, the system is mean square stable if*

$$(1-(\mathcal{C}-\mathcal{E}))|\lambda_{max}(A)|^2 < 1 \ . \quad (10)$$

It is instructive to compare this condition to that in Theorem 1: the term $\mathcal{C}-\mathcal{E}$ indicates that incorrect inputs essentially 'cancel out' a certain number of the correct inputs. As expected, this is worse than a channel that simply drops inputs, which only limits the number of correct inputs that are applied. Note that the value of the bound b does not impact the mean square stability of the system as long as $b < \infty$; the proof of the theorem reveals, however, that the bound on the second moment of the state increases with b.

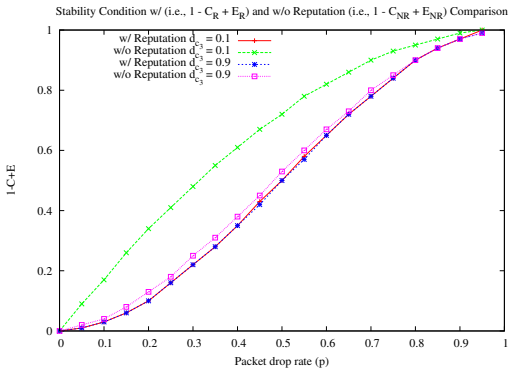


Figure 8: Stability condition $(1-\mathcal{C}+\mathcal{E})$ with and without reputation for different p and d values

Finally, to verify that the reputation manager described in Section 5 improves the metric $1-\mathcal{C}+\mathcal{E}$ for stability from equation (10), Fig. 8 shows the variation in the value of $1-\mathcal{C}+\mathcal{E}$ with respect to p, for a few representative d_{c_3} values. The main point to note in the graph is that the value of $1-\mathcal{C}+\mathcal{E}$ with no reputation is invariably higher compared to reputation-based networked control. The difference in performance decreases with the increase in packet-drop rates. Note that reputation management is not guaranteed to stabilize the system for all drop probabilities; this work is intended to demonstrate that reputation management can provide mean square stability under larger drop probabilities than majority voting. In summary, these results collectively demonstrate the utility of using even a very simple reputation scheme to augment networked controller with modular redundancy to compensate for faults.

7. CONCLUSIONS & FUTURE WORK

In this paper we studied a reliable networked control scheme to ensure mean square stability when the channel between the controller and actuator is faulty, in addition to dropping packets. To achieve this, we first studied the use of triple modular redundancy, but showed that due to the potential for packet drops, a straightforward application of majority voting in such a scheme may not be sufficient to ensure stability. We characterized the amount of additional redundancy that would be required in order to rectify this situation. We then provided a *reputation management* scheme to reduce the amount of redundancy required. The scheme builds on majority voting and improves the probability of applying correct inputs to the system, but potentially injects (bounded) incorrect inputs as well. We then generalized existing results on networked control to show that mean square stability will be maintained as long as the bounded incorrect inputs are applied to the plant infrequently enough.

Having introduced a reputation management scheme for networked control systems, there are a variety of avenues for future research. First, we intend to study the effects of other control policies on the ability to stabilize the system in the presence of data-corrupting channels (e.g., allowing the manager to apply the previously applied input when unverifiable values are received, instead of applying an input of zero). Second, we intend to perform a detailed analysis of the convergence of the channel reputations to their steady states (i.e., how quickly the reputations go from being 'un-

known' to providing an accurate representation of the channels' reliability). One approach would be to model each possible combination of channel reputations as the states of a Markov Chain, and then to analyze the mixing time of the chain. We also intend to study more general networked control architectures, where the plant's sensors are no longer located at a single point, but instead geographically dispersed. This would necessitate the use of different state-estimators or controllers at each of those locations, and would require a scheme to fuse these different values appropriately (perhaps by making connections with recent work on trust-based distributed Kalman filtering [16]). In cases where the values received by the reputation manager are noisy, or are transmitted asynchronously through the network. A new metric would have to be defined in order to determine which values "agree", and which value is sufficiently different from the others that it can be tagged as incorrect. The work on bounded-delay and threshold majority voting from [5] would be of interest in this regard. Finally, the chosen reputation functions were well-suited for the fault model assumed for this work. However, more elaborate fault models may require more elaborate reputation functions to ensure stability. It would be interesting to see what specific mathematical properties reputation functions should possess to ensure the plant is stabilized. Ultimately, our goal is to build a complete foundational framework for reputation-based networked control.

8. ACKNOWLEDGMENT

This research was supported in part by a grant from NSERC, ONR MURI N00014-07-1-0907, NSF CNS-0834524 and NSF CNS-0931239. Chinwendu Enyioha is supported by a Ford Fellowship administered by the National Research Council of the National Academies.

9. REFERENCES

[1] S. Amin, A. Cardenas, and S. Sastry. Safe and secure networked control systems under denial-of-service attacks. In R. Majumdar and P. Tabuada, editors, *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 31–45. 2009.

[2] A. Bemporad, W. Heemels, and M. J. (eds). *Networked Control Systems*, volume 406. Lecture Notes in Control and Information Sciences, Springer-Verlag, 2010.

[3] M. Bishop. *Introduction to Computer Security*. Pearson Education Inc., 2005.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] P. Caspi and R. Salem. Threshold and bounded-delay voting in critical control systems. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1926 of *Lecture Notes in Computer Science*, pages 70–81. 2000.

[6] J. Chang, K. Venkatasubramanian, A. G. West, S. Kannan, I. Lee, B. Loo, and O. Sokolsky. AS-CRED: Reputation service for trustworthy inter-domain routing. In *University of Pennsylvania Technical Report, MS-CIS-10-17*, April 2010.

[7] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, Dec. 1996.

[8] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *IEEE Transactions on Automatic Control*, 54(8):1807–1819, Aug. 2009.

[9] V. Gupta and N. C. Martins. On stability in the presence of analog erasure channels. In *Proc. of the 47th IEEE Conference on Decision and Control*, pages 429–434, 2008.

[10] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proc. of the 41st IEEE Conference on Decision and Control*, pages 1205–1210, 2002.

[11] J. P. Hespanha. *Linear Systems Theory*. Princeton University Press, 2009.

[12] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proc. of the IEEE*, 95(1):138–162, Jan. 2007.

[13] O. C. Imer, S. Yuksel, and T. Basar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 42(9):1429–1439, Sep. 2006.

[14] R. Ismail and A. Josang. The beta reputation system. In *the 15th BLED Electronic Commerce Conference*, page 41, 2002.

[15] K. Jain. Security based on network topology against the wiretapping attack. *IEEE Wireless Communications*, 11(1):68–71, Feb. 2004.

[16] T. Jiang, I. Matei, and J. S. Baras. A trust based distributed Kalman filtering approach for mode estimation in power systems. In S. S. Sastry and M. McQueen, editors, *Proc. of the First Workshop on Secure Control Systems*. 2010.

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proc. of the 12th Int. Conf. on the World Wide Web*, pages 640–651, 2003.

[18] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[19] A. Mesquita, J. Hespanha, and G. Nair. Redundant data transmission in control/estimation over wireless networks. In *Proc. of the 2009 American Control Conference*, pages 3378–3383, 2009.

[20] S. S. Sastry and M. McQueen. Proceedings of the first workshop on secure control systems, 2010.

[21] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proc. of the IEEE*, 95(1):163–187, Jan. 2007.

[22] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proc. of the 2001 American Control Conference*, pages 1491–1496, 2001.

[23] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ICS) security. Technical Report 800-82, National Institute of Standards and Technology, Sep. 2008.

[24] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI'06: Proc. of the 3rd conference on Networked Systems Design & Implementation*, pages 1–1. USENIX Association, 2006.

[25] A. G. West, A. J. Aviv, J. Chang, and I. Lee. Spam mitigation using spatio-temporal reputations from blacklist history. In *Proceedings of ACSAC 2010*, pages 161–170, 2010.