A Play-Back Algorithm for Networked Control*

Vincenzo Liberatore

Abstract

Sensing, actuation, and decision units can be used to control a remote physical environment and enable physical actions independently of distance. Networked control is a real-time distributed application whose effectiveness depends on its ability to tolerate losses and to adapt to delays and jitter. The primary contribution of this paper is a play-back algorithm for networked control. The algorithm is validated on extensive simulations and on a preliminary set of traces collected in the field. The play-back scheme brings system variability close to its inherent values, thereby eliminating almost completely the effect of network vagaries.

1 Introduction

Sensing, actuation, and decision units can be used to control a remote physical environment (Figure 1). Networked control enables physical actions to take place independently of distance, with fundamental epistemological and social consequences [9]. Applications are far-reaching and include, for example, our own contributions to industrial automation (e.g., [28]), distributed instrumentation (e.g., [1]), unmanned vehicles (e.g., [13]), and home robotics (e.g., [21]). A potential development is to augment distributed audio and video with haptics (tactile) interactions (e.g., [14]). A fundamental problem in networked control is that sensing and actuation operate in real-time and, consequently, late or missing signals can jeopardize the stability, safety, and performance of the controlled physical environment. Networked control is a real-time distributed application whose effectiveness depends on its ability to tolerate losses and to adapt to delays and jitter.

The primary contribution of this paper is a play-back algorithm for networked control. A play-back scheme would smooth out network vagaries and would apply control signals to the physical environment at predictable times. Control play-back can in some cases leverage on the methods and intuitions of multimedia play-back. However, networked control differs substantially from multimedia applications in behavior and objectives, and play-back strategies will differ as well. Furthermore, play-back buffers are a departure from known control-theoretical sampling and control algorithms (e.g., [22]).

Networked control addresses sensing and actuation under disparate network conditions and for environments whose physics differs radically in nature and scale. As a result, distributed control applications can differ tremendously in methodology and requirements. At the same time, remote sensing and actuation present also clear similarities across a variety of environments, and so a set of general methods should be established for entire classes of applications. This paper will address networked control over wide-area IP networks and for the broad class of stable linear systems, which include any environment that can be described by linear differential or difference equations.

*Division of Computer Science, Case Western Reserve University. 10900 Euclid Avenue, Cleveland, Ohio 44106-7071, USA. Fax: (216) 368 6039. E-mail: vincenzo.liberatore@case.edu. URL: http://vincenzo.liberatore.org/NetBots/. Work supported in part under NSF CCR-0329910 and Department of Commerce TOP 39-60-04003.

A physical environment is, roughly speaking, characterized by the speed of its dynamics. For example, a Tokamak reactor [25] has much faster dynamics than a manufacturing robot [20]. A faster environment is typically harder to control than a slower one. Moreover, most environments become more difficult to control as network service levels deteriorate. For example, controlled systems cannot always be made stable in the presence of packet losses [33]. As a result, network service levels impose a bound on the fastest physical dynamics that can be supported in the target environment. At the same time, control applications can be rendered more tolerant and adaptive by play-back algorithms. In summary, the *rules of the game* are to design end-system geographically-scalable play-back algorithms that make control applications more tolerant and adaptive to network vagaries and thus able to cancel stronger disturbances for a broad class of physics.

The paper will focus on the concepts and explanations. Furthermore, the paper will strive to be as self-contained as possible. Details are available in the appendices and software. Section 2 reviews the background on networked control. Section 4 introduces the play-back scheme. Section 5 describes the evaluation. Section 7 reviews related work and Section 8 concludes the paper.

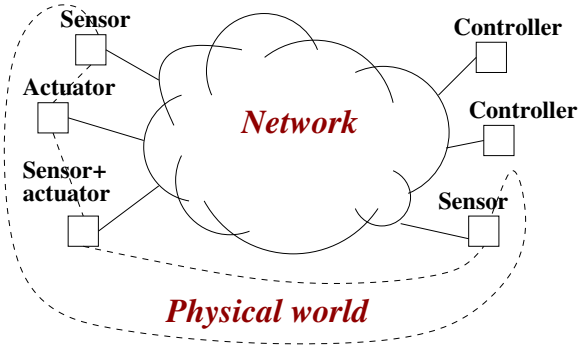

Figure 1: The *networked control* vision.

2 Networked Control: Background

The definition of a *plant* captures the notion of a physical system where sensor data are collected and control signals are delivered with the objective of affecting the physical environment. The term "plant" originated from the industrial automation area, but it has since been applied to express a generic unit of sensing and actuation (e.g., [33]). Section 2.1 discusses an abstract plant and Section 2.2 introduces networked plants. This section reviews previous work in the area, and its contents are often based verbatim on previous papers (e.g., []).

2.1 The Scalar Linear Plant

The scalar linear plant is the primary type of plant that will be considered in this paper due to its generality and relative simplicity. However, results can be generalized to higher-dimensional linear systems The focus on scalar linear plants is thus mostly for concreteness and ease of exposition. A scalar plant is characterized by its *state* $x(t)$, an *input* $u(t)$ (the control signal), an *output* $y(t)$ (the sensor data), a *state disturbance* $v(t)$, and an *output disturbance* $w(t)$, which are continuous-time stochastic processes. In a stable scalar linear plant, the state, input, output, and disturbances are

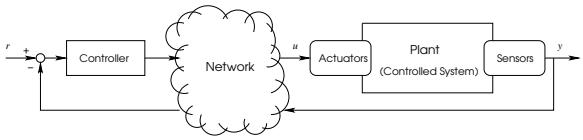

Figure 2: Network-based control.

related by the equations:

$$\begin{cases} \dot{x}(t) = ax(t) + bu(t) + v(t) \\ y(t) = x(t) + w(t) \end{cases}, \tag{1}$$

where $a \leq 0, b$ are real numbers. Equation (1) is generic across applications because it models any physical system characterized exactly or approximately by a differential equation. Linear plants have numerous applications [1, 13, 28, 21] and any textbook on control engineering would present hundreds of examples (e.g., [7, 23]). Furthermore, linear systems describe the behavior of hybrid linear systems during each mode of operation, and as such they are a fundamental building block for control paradigms that overcome the limitations of classical control theory [31].

The dynamics of plant (1) depend on the value of a. For example, if $u = v = 0$, then $x(t) = x(0)e^{at}$, which increases more rapidly for larger values of a. In general, larger values of a are thought to denote faster plant dynamics [7, 23], and a unit increase in the value of a denotes an exponentially faster plant evolution.

The stochastic processes $v(t)$ and $w(t)$ model exogenous disturbances and noise. The impact of noise can be attenuated by filters. Since $v(t)$ appears in the same equation as the input, it sums up the disturbances on the state and on the input. A common model for $v(t)$ is the formal derivative of Brownian motion. A common model for $w(t)$ are independent normal random variables with mean 0 [7]. In these cases, $E[\dot{x}(t)] = ax(t) + bu(t)$, and $E[y(t)] = x(t)$, which partly accounts for the common practice of disregarding the disturbances during an initial design and analysis phase (e.g., [7, 23]). Of course, disturbances are present in a physical systems and have to be reintroduced later on for a comprehensive evaluation of control algorithms. If $v = w = 0$ is assumed, then it becomes analytically possible to exploit a perfect *observer* to accurately predict the system state forward in the future.

The control design problem is basically to achieve certain system properties by appropriately setting the input $u(t)$. For example, the design could specify a *reference output* $r(t)$ and the problem is to make $y(t)$ as close as possible to $r(t)$. In this case, a design objective could be to achieve *tracking*, which is, informally, the distance between $y(t)$ and $r(t)$. If $r(t) = r$ is a constant, then r is called a *set point*. This paper will consider $r = 0$, but the same arguments hold for any other set point with a slight notational complication.

In a *digital control system*, the plant output $y(t)$ is sampled at time $t_1, t_2, \ldots$ to generate the time series $y(t_1), y(t_2), \ldots$. *Even sampling* is the particular case in which the output is being sampled at regular intervals $t_{i+1} = t_i + T$, where T is the constant duration of a *sampling period*. This paper will assume even sampling. The inverse of the sampling period $1/T$ is called the *sampling rate* or *sampling frequency*. Returning to the general case of arbitrary sampling, a *controller* sets the value $u(t) = u_i$ in the interval $[t_i, t_{i+1})$ as a function of $y(t_1), y(t_2), \ldots, y(t_i)$ so as to achieve, say, tracking of a reference set point r.

2.2 Networked Control

A *networked control system* is a digital control system in which the sensor data $y(t_i)$ and control signals u_i ($i \geq 1$) are delivered over a network [22, 15]. Networked control typically involves a plant that is controlled remotely, and Figure 2 shows the corresponding information flows.

Networked control is complicated by the presence of delays, jitter, and packet losses. In the first place, packets can be lost either because they are dropped out in the network infrastructure or because they discarded by the communication end-points, for example, if they arrive out of order. Furthermore, network latency can delay the application of a control signal at the plant site.

In traditional networked control (e.g., [22, 33]), the ith sensor reading $y(t_i)$ reaches the controller after a delay, the controller computes u_i as a function of $y(t_1), y(t_2), \ldots, y(t_i)$, sends u_i back to the plant, which will apply the control u_i in the interval $[\theta_i, \theta_{i+1})$, where $\theta_i = t_i + \mathrm{RTT}_i$ and RTT_i is the value of the round-trip time at the ith step. It will be convenient to divide the ith RTT into two terms as $\mathrm{RTT}_i = \tau_i + \xi_i$, where τ_i is a *nominal RTT* between the plant and the controller and ξ_i is the corresponding *jitter*. The nominal RTT τ_i can be chosen in a variety of ways depending on its role in the sampling and control scheme. The value of τ_i is often a constant or a slowly varying quantity, whereas ξ_i accounts for short-term fluctuations. If either $y(t_i)$ or u_i is lost in the loop from sensor to actuator, then plant behavior is equivalent to the case when y was not sampled at all at time t_i. Hence, packet losses effectively alter the sampling schedule. In particular, packet losses can introduce uneven sampling even if the original plant's sampling was evenly spaced. As a result, the sampling period T must be replaced by a time-varying quantity $\Delta t_i = t_{i+1} - t_i$. Since out-of-order signals are discarded, it can always be assumed that $\theta_{i+1} \geq \theta_i$. In-order delivery can be enforced with sequence numbers, as discussed at length, for example, in [26, 30].

A *proportional controller* uses the last sample $y(t_i)$ to predict $y(t_i + \tau_i)$ with a perfect observer and then generates the control $u_i = -ky(t_i + \tau_i)$ for the interval $[\theta_i, \theta_{i+1})$. A *deadbeat controller* [] is a proportional controller that sets k equal to

$$k_d = \begin{cases} \dfrac{a}{b}\dfrac{e^{aT}}{e^{aT}-1} & \text{if } a \neq 0 \\ -1/T & \text{otherwise} \end{cases} .$$

In the absence of disturbances and network vagaries, the k_d controller converges to $r = 0$ in a time interval of length T. In a less ideal setting, k_d can still have excellent performance []. The k_d controller will be used as the baseline for the evaluation of other controllers.

The property $\xi_i = 0$ (no jitter) can be guaranteed with a *play-back buffer* that applies the control u_i exactly at time $t_i + \tau_i$ if it arrived before that time and drops the control if it was received too late. The τ_i value is called the *play-back* delay. A play-back buffer can increase the loss rate. A relaxation is a *reverse play-back buffer*, which works exactly like a regular play-back buffer for early signals, i.e., it holds them until the play-back time $t_i + \tau_i$, but it does play back also late non-reordered signals as soon as they arrive (cite paper). The reverse play-back method is based on the intuition that control signals are better received late than never (this intuition does not necessarily carry over to other real-time applications). Moreover, the reverse play-back scheme does not introduce additional losses, but can retain a certain level of jitter.

3 The Networked Control Problem

Networked control is essentially the problem of reducing the uncertainty in the state of a system. The plant starts in a state $x(t_i)$, which is known only approximately as $y(t_i)$. An action u_i is

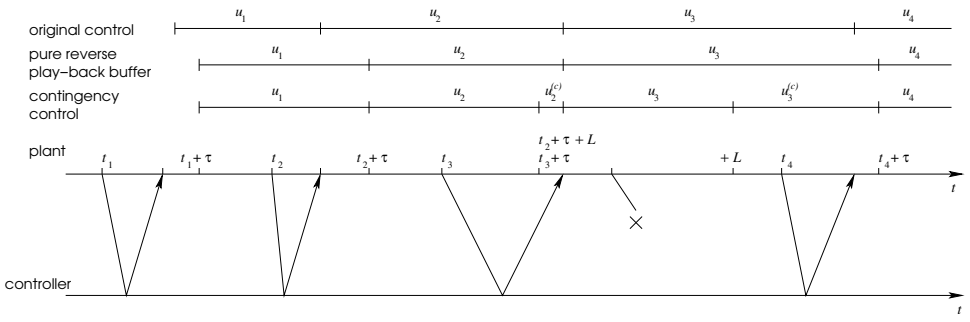

Figure 3: The timeline of network events in the remote control of a plant. In this example, τ_i and $L_i = T$ are constants.

specified and applied at an instant θ_i, which is not known exactly due to network vagaries. The system state then evolves as a function of u_i and of a random disturbance $v(t)$. The goal is to make $y(t) = r$ for a perfectly specified set point r. In summary, the objective is to overcome multiple sources of uncertainty to bring the system to a known and desirable set point. Uncertainty stems from two types of sources:

- The stochastic processes $v(t)$ and $w(t)$, which model measurement errors and exogenous disturbances, and
- Network vagaries, which cause control signals to be applied at non-deterministic instants.

Network vagaries are similar to disturbances in that both can introduce random time-varying deviations of the system state from its nominal value. However, network vagaries and disturbances are also distinct phenomena: disturbances model the effect of physical behaviors on the continuous-time dynamics, whereas network vagaries are imperfections of the digital feedback loop.

Certain networked control systems involve plants that are resource-constrained devices. In these cases, the plant algorithms should be simple, and most of the complexity should be moved to the controller.

4 Play-Back Algorithm

4.1 Sampling, Control, and Communication

The main contribution of this paper is an end-point play-back algorithm for sampling and control. Our scheme is a departure from the proportional controllers (Section 2) in that the controller sends to the plant *two* control signals, which will be called the *regular control* (denoted by u_i) and the *contingency control* (denoted by $u_i^{(c)}$). Additionally, the controller sends also a *play-back delay* τ_i and the *duration* L_i of the regular control. The start time basically implements a (reverse) play-back buffer: the regular control is applied at time $t_i + \tau_i$, or immediately if the play-back time has already passed. The duration L_i forces the plant to switch to the contingency control $u_i^{(c)}$ if a new regular control has not been received after L_i seconds since the time when the control was applied. Although L_i could change over time, it would be presumably fixed as a tunable parameter and change only occasionally. Figure 3 exemplifies a representative timeline. The regular control action u_i is generated by a deterministic deadbeat controller. The contingency action is $u_i^{(c)} = 0$

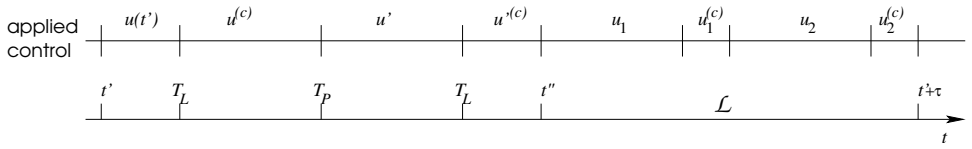

Figure 4: Example of a timeline generated by the observer, using the plant state $(u(t'), u^{(c)}, u', u'^{(c)}, T_P, L)$, and a log $\mathcal{L}$ containing $u_1, u_1^{(c)}, u_2, u_2^{(c)}$.

in the case of scalar plants, but $u_i^{(c)}$ is in general an action sequence for multi-dimensional plants (details omitted). The intuition is that deadbeat control makes $x(t_i + \tau_i + L_i) = r$ if controls are played back at predictable times and in the absence of disturbances, and then $u_i^{(c)}$ would result into $x(t) = r$ for all $t \geq L_i$. In Figure 3, this nominal behavior occurs in correspondence to u_1 and u_2. The reverse play-back can enforce a similar behavior in correspondence of u_1 but not of u_2 and, in general, it is improved upon in the case of heavier jitter or packet losses. The nominal behavior requires that $L_i \leq T$, and for simplicity the rest of the paper fixes $L_i = T$.

Both plant and controller discard signals unless they are received in-order. In this context, in-order delivery means that the sequence number is higher than any previously received one, even if it shows a gap in the received sequence (special consideration are made, as usual, for the case when the sequence numbers wrap around [30, 26]). Additionally, packets are discarded if $t' + \tau'$ is more than the current expiration value of T_P to avoid the pathological scenario in which $t_i + \mathrm{RTT}_i < t_{i+1} + \mathrm{RTT}_{i+1} < t_i + \tau_i$ for all i's. In this case, packet $i+1$ resets the play-back timer T_P and effectively cancels the reception of the previous packet i. If this event occurs repeatedly, packets would cancel each other in turn and no control would be applied. This pathological scenario is avoided if a new packet must be replayed no later than the outstanding play-back delay in T_P. If $t' + \tau'$ is more than the expiration value of the play-back timer T_P, the discarded packet will be said to be a *quashed* packet. Finally, a packet is discarded if it is too old, and specifically if the packet reception time is more than $t' + \tau' + L'$.

Algorithm 1 describes the plant in pseudo-code, and Algorithm 2 describes the controller. The plant algorithm (Algorithm 1) basically implements the reverse play-back and the contingency mechanisms, and it also collects statistics for future use by the controller. The controller assumes various black-box components that are well-known in real-time distributed systems. High-resolution timers are available in real-time Operating Systems, can be implemented either with active waiting states or with signals, and efficient data structures exist to support multiple outstanding timers [32]. RTP is implicitly used to send the value of the current timestamp as well as the packet sequence number. The plant algorithm is short (less than 70 statements of C code) and simpler than the controller's in that it avoids various numerical computations. Furthermore, the timer T_S can be eliminated if the controller polls the plant. The simplicity of the plant algorithm stems in part from the short memory of the algorithm, which keeps track of at most two control signals. As we shall see, the small footprint comes at the price of reduced performance at high sampling rates.

4.2 Observer

The controller predicts the value of $y(t' + \tau)$ by invoking an *observer* [7, 23]. A perfect observer would return an accurate prediction but, in reality, the prediction is hampered by a variety of factors. The observer uses known factors whenever they are available, but it resorts to assumptions otherwise.

Algorithm 1 Plant

Require: T is the sampling interval in seconds.

1: Set timer T_S to expire every T seconds
2: `RTT-sample`, `hiseqno` $\leftarrow$ `undef`
3: **loop** {Main Loop}
4: Enter a synchronous multiplexed read from the controller which blocks either until a control is received or until a timer expires.
5: **if** T_S expired **then**
6: Sample system output y
7: Marshal y, the current control action, and the value of local variables into an (RTP) packet to the controller
8: **else if** an in-order control $(u', u'^{(c)}, t', \tau', L')$ is received **then**
9: `RTT-sample` $\leftarrow$ current time$-t'$
10: **if** $t' + \tau' \leq T_P$'s expiration time and current time $< t' + \tau' + L'$ **then**
11: `hiseqno` $\leftarrow$ sequence number of the received control
12: **if** current time $< t' + \tau'$ **then** {Arm a play-back timer}
13: Cancel T_P
14: Set T_P to expire at time $t' + \tau'$
15: **else** {Reverse play-back}
16: Cancel T_L and T_P
17: $u \leftarrow u'$
18: $u^{(c)} \leftarrow u'^{(c)}$
19: $L \leftarrow L'$
20: Set T_L to expire in L seconds
21: Apply the control u
22: **end if**
23: **end if**
24: **else if** T_P expired **then**
25: Cancel T_L
26: $u \leftarrow u'$
27: $u^{(c)} \leftarrow u'^{(c)}$
28: $L \leftarrow L'$
29: Set T_L to expire in L seconds
30: Apply the control u
31: **else** {T_L expired}
32: Apply $u^{(c)}$
33: **end if**
34: **end loop**

Algorithm 2 Controller

Require: T is the sampling interval in seconds.

1: $L \leftarrow T$
2: **if** $a \neq 0$ **then**
3: $\quad k \leftarrow \frac{a}{b} \frac{e^{aL}}{e^{aL} - 1}$
4: **else**
5: $\quad k \leftarrow -\frac{1}{L}$
6: **end if**
7: **loop** {Main Loop}
8: $\quad$ Read an in-order sample from the plant with y, the control action at time t', and the local plant variable values from the plant
9: $\quad$ Update τ as a function of `RTT-sample` (Algorithm 4)
10: $\quad$ Predict $y(t' + \tau)$ with the observer (Algorithm 3)
11: $\quad u \leftarrow -ky$
12: $\quad u^{(c)} \leftarrow 0$
13: $\quad$ Marshal $(u, u^{(c)}, t', \tau, L)$ into an (RTP) packet to the plant
14: $\quad$ Record the packet in the log $\mathcal{L}$
15: **end loop**

The value of $y(t' + \tau)$ is based on the state of the system at time t' and on its evolution in the interval $(t', t' + \tau)$. The evolution in $(t', t' + \tau)$ depends on disturbances that are inherently unpredictable. However, $E[v(t)] = E[w(t)] = 0$ and an estimate of $y(t' + \tau)$ can be obtained by ignoring disturbances. As for the system state at time t', it obviously consists of the plant state $x(t') \simeq y(t')$, but also of the value of the local variables in Algorithm 1. Furthermore, the state of a distributed system depends also on the messages that are in transit [6]. However, packet arrivals in $(t', t' + \tau)$ are events that lie in the future at the time t' when the sample left the plant and, in general, these events cannot be predicted accurately. The prediction of $y(t' + \tau)$ requires an estimate of $u(t)$ in the interval $[t', t' + \tau)$. The value of $u(t)$ depends on factors that are known exactly and on factors that are not. The known factors are $u(t')$ and the values of the local plant variables at time t'. The unpredictable factors are the messages that will be delivered to the plant in the interval $(t', t' + \tau)$ and their arrival times. The observer must make an assumption and it assumes that outstanding in-order regular and contingency actions will be applied at their nominal time. An example is shown in Figure 4.

The controller maintains a log $\mathcal{L}$ of outstanding control signals and uses it to predict $u(t)$. The log $\mathcal{L}$ keeps a record only of the control signals that are likely to be used by the plants. First, the observer discards from $\mathcal{L}$ the entries of all messages whose sequence number is less than `hiseqno` because those messages either

- Have already been received and processed at the plant and therefore are already reflected in the plant state $x(t')$, or
- Will never be received, or
- Will be received in the future and discarded as out-of-order.

Second, the observer times out old packets, and specifically it discards those whose activation time is less than $t' - \tau$. Finally, the observer discards the logged signals whose activation time is greater

than or equal to $t' + \tau$ under the assumption that the newly generated control will overtake the logged one, which will be then discarded by the plant. The resulting observer is denoted Algorithm 3.

Algorithm 3 Observer

Require: $\mathcal{L}$ is a log of outstanding control messages that the controller sent the plant.

1: Prune $\mathcal{L}$ of all messages whose sequence number is less than `hiseqno` or whose activation time is smaller than $t' - \tau$ or greater than $t' + \tau$.
2: $t'' \leftarrow$ minimum nominal application time of a control in $\mathcal{L}$ that falls in $(t', t' + \tau)$
3: **if** t'' exists **then** {Outstanding controls can affect $y(t' + \tau)$}
4: Generate $u(t)$ $(t'' \leq t \leq t' + \tau)$ assuming that regular and contingency controls in $\mathcal{L}$ are applied at their nominal time
5: **else**
6: $t'' \leftarrow t' + \tau$
7: **end if**
8: Generate $u(t)$ $(t' \leq t \leq t'')$ from the plant state variable ($y(t')$, $u(t')$, the value of the variables $u^{(c)}$, u', $u'^{(c)}$, and the expiration time of T_P and T_L)
9: Simulate the plant dynamics with $u(t)$ starting from $y(t')$ to obtain an estimate of $y(t' + \tau)$

4.3 Play-Back Delay

The play-back delay τ_i mandates the nominal time when a control is applied as the plant input. If τ_i is too large, the dynamics are dominated by disturbances and plant evolution becomes unpredictable. If τ_i is too small, the control signals would be applied at unpredictable times and the system dynamics cannot be controlled accurately. Therefore, an appropriate value of τ_i should strike a balance between delay and jitter. The estimation of τ_i is similar to play-back problems in distributed multimedia. The main differences are that most distributed multimedia are concerned with one-way delays and that, in most of those applications, the play-back delays are determined by the same end-point that plays the signal out. Networked control is primarily based on round-trip times and play-back intervals are determined by the controller. These differences prevent the immediate application of certain multimedia play-back schemes to networked control. For example, multimedia play-back sometimes employ the strategy of following delay spikes [27], but the controller can in general detect a spike only after an RTT, so that the original algorithm cannot be applied directly. Play-back values are also related to the problem of estimating TCP's RTO [26]. The RTO tends to be conservative because of the tremendous cost of a TCP time-out. An equally conservative τ_i would effectively mean a larger delay in the feedback loop, with well-known control-theoretical complications (for example, the plant dynamics would be dominated by disturbances). Additionally, if τ_i is too conservative, the plant would apply a signal much later than its reception. Therefore, it would have to buffer packets for a relatively long time, and a longer buffer interval would progressively increase the likelihood that more packets would be quashed. As a result, an overly conservative τ_i could increase the loss rate and negatively impact the effectiveness of control.

Although different methods can be used, this paper adapts for concreteness the short-term component of the recent *peak-hopper* algorithm [8]. The play-back algorithm (Algorithm 4) is a modular component within the controller (Algorithm 2) and the estimation of τ_i can be easily replaced with a different implementation. Algorithm 4 borrows from [8] the general structure of the algorithm, its initializations, and the values of the constant factors. As compared to the original

	Open-Loop	**Pure Delays**	**Play-Back**	**Inherent**
σ_y	14.1657	5.7105	5.1507	2.2447
99-percentile $\lvert y \rvert$	36.4254	14.7639	13.3336	5.7968
Maximum $\lvert y \rvert$	76.7715	34.7771	46.0276	13.2214

Table 1: Summary statistics of two weeks of simulated time. Pure Delays and Inherent are ideal (non-implementable) baselines.

peak-hopper, the controller can avoid the complicate interactions with various TCP features such as, say, timestamps. Furthermore, the initializations are less critical than in the original algorithm since one would presumably warm it up before attempting to manipulate a remote plant (e.g., [11]). Finally, the long-term component has been eliminated since it was too conservative in the face of disturbances.

On the other hand, Algorithm 4 introduces an upper bound on τ to avoid quashed packet. A quashed packet is typically worse than a late one because the late packet can be replayed by the reverse play-back. Therefore, τ_i should preferably be set so as to avoid losing packets when $t' + \tau_i$ follows the play-back time T_P even if a few more packets arrive after their application time $t' + \tau_i$. It is easy to see that packet i will not be quashed if $\tau_i \leq T + \mathrm{RTT}_{\min}$, where $\mathrm{RTT}_{\min}$ is the smallest round-trip time between plant and controller. Packet losses can be avoided by setting an upper bound on τ, which in turn requires an estimation of $\mathrm{RTT}_{\min}$. In related literature, $\mathrm{RTT}_{\min}$ was captured from a sequence of samples. In networked control, $\mathrm{RTT}_{\min}$ should be adjusted dynamically by a long-lived and continuously running controller. Furthermore, it is preferable to obtain a slight underestimate of $\mathrm{RTT}_{\min}$ rather than an overestimate. The $\mathrm{RTT}_{\min}$ value is estimated with a symmetric version of the peak-hopper algorithm applied to find a lower bound on the RTT (details omitted). In the algorithm, $\mathrm{RTT}_{\min} \simeq (1 - C) r_{\min}$. In turn, $r_{\min}$ is an adaptive estimate of the minimum round-trip time and C is an estimate of the negative variability of round-trip times. First, the C term is an exponential moving average of the negative variability and it is calculated with the same weights as in TCP RTT variability. Second, $r_{\min}$ is adjusted whenever any smaller round-trip time is detected, and it is also aged progressively to adapt to longer-term changes. The aging factor is proportional to the absolute variability of round-trip times as captured by the $r' - r_{\min}$ term. Since $r_{\min}$ is aged, it tends to be based only on the past few samples and, consequently, it can overestimate $\mathrm{RTT}_{\min}$. Therefore, $r_{\min}$ is corrected by a term $1 - C$ that expresses the effect of the long-term negative variability of round-trip times.

4.4 Noise and Pipes

The sampling period T and the packet size determine the amount of bandwidth used by networked control. Since T is related to bandwidth and τ_i to communication delays, T and τ_i are largely independent of each other. In particular, nothing prevents $T \ll \tau_i$, in which case a sequence of samples and control signals are typically in flight in the network. Such a scenario will be called a *control pipe*. Although the pipe is, in some sense, stored in the network, the plant (Algorithm 1) does not keep state for the packets in the pipeline. Although the plant could in principle store multiple signals, its implementation would become more involved and would not be appropriate for resource-constrained systems.

Control pipes have advantages and disadvantages. A small T increases the amount of computation and storage required in $\mathcal{L}$, it complicates the computation of $u(t)$, and in general it increases

Algorithm 4 Play-Back Delay

Require: B is initially set to 0.25, C is initially set to 0.75. If available, r_1 is the new RTT sample, r_0 is the previous RTT sample

Ensure: Computes a new value of the play-back delay τ in seconds.

if there is no RTT sample **then**
  $\tau \leftarrow 3$
else if there is only one RTT sample r_1 **then**
  $\tau \leftarrow 3r_1$
else if there are only two RTT samples r_1 and r_0 **then**
  $\tau \leftarrow 1.25r_1$
  $r_{\min} \leftarrow \min\{r_1, r_0\}$
else
  $\delta \leftarrow \dfrac{r_1 - r_0}{r_0}$
  $B \leftarrow \min\{\max\{2\delta, 0.9375B\}, 1\}$
  if $\delta < 0$ **then**
    $C \leftarrow 3C/4 - \delta/4$
  end if
  $r' \leftarrow (1 - C)\min\{r_1, r_0\}$
  if $r_{\min} < r'$ **then**
    $r_{\min} \leftarrow r_{\min} + \dfrac{r' - r_{\min}}{16}$
  else
    $r_{\min} \leftarrow r'$
  end if
  $\tau \leftarrow \min\{(1 + B)\max\{r_1, r_0\}, T + r_{\min}\}$
end if

Parameter	Default Value	Short Description
α	1.5	Shape of the Pareto distribution (tail)
λ, r	$\lambda = 1000, r = 2$	Parameters of the gamma distribution (body)
k	$2r/\lambda$	Minimum of the Pareto distribution (tail)
p	0.95	Probability that the Gilbert model remains in the good state
p_{spike}	.99	Probability of a delay spike
q	0.2	Probability that the Gilbert model remains in the lossy state
$\text{RTT}_{\min}$	50ms	Minimum RTT
T	10ms	Sampling period

Table 2: Parameters that define the synthetic traces.

the computational demands on the observer. Moreover, if jitter is comparable with the sampling period T, the behavior of the pipe becomes unpredictable and it makes it harder for the observer to establish if or when control signals are going to be applied at the plant. Therefore, although the sampling period T is in principle independent of absolute delays, it is related to the delay variability. The most immediate advantage of a control pipe is that it provides a level of redundancy against packet losses. Specifically, if a control signal u_i did not arrive by time $t_i + \tau_i$, the controller would be able to detect u_i's late arrival when it receives the first sample that was generated after time $t_i + \tau_i$. A smaller sampling period T would make it more likely that the detection occurs earlier. A second advantage of pipes is that it enables a controller to quickly countermand a previous control signal. For example, a transient delay spike could increase considerably the value of τ_i only to deflate it on the next round. If $t_{i+1} + \tau_{i+1} < t_i + \tau_i$, then u_{i+1} is applied at time $t_{i+1} + \tau_{i+1}$, which also means that u_i will never be applied. In some sense, the controller has acted at first as if the spike represents a long-term change of RTT and issued the control u_i accordingly, only to realize afterwards that the delay spike was a transient phenomenon, and countermanded u_i with a more appropriate control action. Revocations are particularly helpful if the controller can issue the next command quickly, that is, if the control pipe has a small period T.

Another advantage of pipes is its tolerance to state disturbances and, as such, it is best exemplified in the ideal case in which $w(t) = 0$ and all packets are applied at their nominal times. In this case, it is easy to show that $\text{Var}[x(t_i + \tau_i + T)]$ is an increasing function of $\tau_i + T$. Hence, the plant dynamics can be made more predictable in the face of state disturbances by taking smaller values of T. In other words, larger bandwidth can partly compensate for long delays. The method could be especially useful given that bandwidth tends to improve faster than delays [24] and to ensure geographically scalable control (cite paper).

5 Experimental Methodology

5.1 Synthetic Traces

Synthetic traces of RTTs were generated by combining a shifted gamma distribution (as in [18]) for the body of the delay distribution, a Pareto distribution for the tail (as in [16]), a model for delay spikes as described in [27], and a Gilbert model (Figure 5) for packet losses [34].

A sample is generated every T seconds. The sample is lost if the Gilbert process is in the lossy state (The Gilbert model is a two-state Markov chain as shown in Figure 5 and is defined by p, the probability of remaining in the good state, and q, the probability of remaining in the

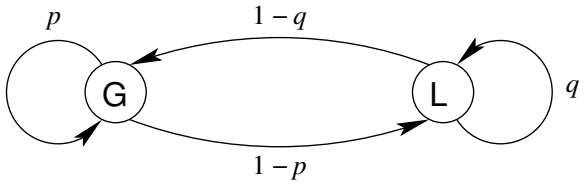

Figure 5: The Gilbert model for packet losses [34].

lossy state.) If the packet survives, its RTT belongs to the body with probability 99% and to the tail with the remaining 1% probability, as in [16]. If the packet falls in the body, its RTT is calculated from a shifted gamma distribution with parameters $\mathrm{RTT}_{\min}$, λ, and r. If the packet falls in the tail, then its RTT is calculated from a Pareto distribution of shape α and minimum k. The value of k is chosen so as to be sufficiently separated from the mode and from the mean of the gamma distribution (body). Delay spikes are simulated by checking whether the RTT would cause an out-of-order delivery. If it would, then the RTT is changed to the previous delivery time plus a small interval (in the simulations, 0.1ms) with probability p_{spike} to simulate a burst of almost simultaneous arrivals, and it is left to its simulated value otherwise. The default trace length is 15 minutes, although several simulations were run for up to two weeks of simulated time. The simulation parameters are summarized in Table 2 along with their default values.

The default value of $b = 1$, which fundamentally means that the input units have been normalized. The default plant has $a = -1$. The output disturbance is a Gaussian random variable with mean 0 and variance W^2, for a simulation parameter W. The default is $W = 1$, which fundamentally means that the output units are normalized around the variability of the output disturbance. As for state disturbances, $x(t)$ can be expressed as $x(t) = x_0(t) + x_N(t)$, where $x_0(t)$ obeys

$$\begin{cases} \dot{x}_0(t) = ax_0(t) + bu(t) \\ x_0(t') = x(t') \end{cases}$$

and captures the nominal disturbance-free evolution of the system, and $x_N(t)$ obeys

$$\begin{cases} \dot{x}_N(t) = ax_N(t) + v(t) \\ x_N(t') = 0 \end{cases}$$

The function $x_0(t)$ can be integrated analytically for any given $u(t)$. The stochastic process $x_N(t)$ can be approximated by finite differences with the formula [5]:

$$x_N(t' + (i+1)h) = (1 + ah)x_N(t' + ih) + \sqrt{h}v(i) \ ,$$

where h is the integration step and $v(i)$ is a Gaussian random variable with mean 0 and variance V^2, for a simulation parameter V. The simulator uses a step equal to $h = (t - t')/4$ for a sufficiently accurate integration of x_N. The default is $V = 20$. The simulator uses the Mersenne twister to generate uniformly distributed random numbers [17] and simulates Gaussian random variables with the Box-Muller method [29].

5.2 Metrics

The simulation outcome is most immediately expressed by the plant output $y(t)$. However, metrics must be more succinct if they are to capture long simulations and to compare different schemes.

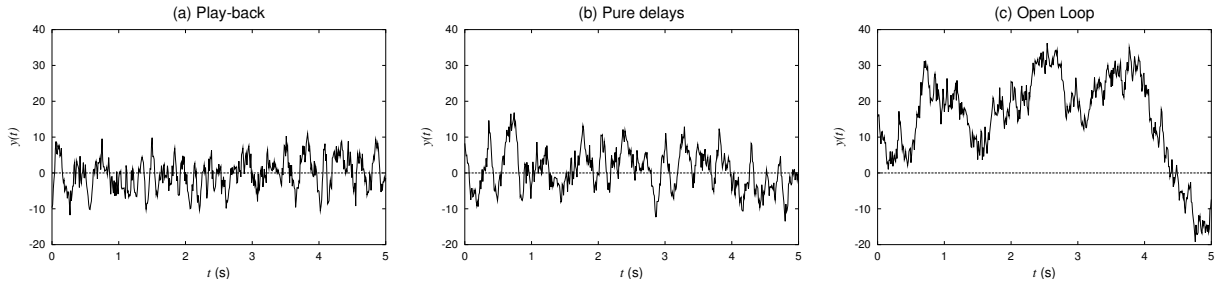

Figure 6: Plant output $y(t)$.

Summary metrics will be based on the time series $Y = \{y(i/100) : i > 0\}$, which captures the plant output at regular intervals of an arbitrary short length equal to 10ms. Our first metric is σ_y, the standard deviation of Y. For comparison, the expected value of Y is $E[y(t)] = 0$. Second, we consider the 99-percentile of the deviation, that is, the value $\tilde{y}$ for which $\Pr[|y| > \tilde{y} : y \in Y] \leq 0.01$. Finally, we consider the maximum deviation $\max\{|y| : y \in Y\}$.

The play-back controller is compared with three baseline simulations. The first one is the output generated by the open-loop plant (1) and expresses the system behavior in the absence of any controller. The first comparison aims at establishing the benefit of using a control and communication algorithm against the *laissez-faire* strategy. The second simulation assumes that the plant is connected to a proportional controller by a channel with constant delay equal to $\text{RTT}_{\min}$. The second comparison aims at providing a lower bound on the algorithm performance by assuming that the network is perfect (no losses, no jitter) and even if other proportional controllers could be used beside the deadbeat. However, the comparison assumes no observer, and will be denoted as the *pure delay* scenario. In the default scenario, σ_y was minimized by $k = 11$, as this value strikes a balance between the transfer functions of the two disturbances on state and on output. Incidentally, $k = 11$ leads to a gain margin of 9dB and a phase margin of 60° in the corresponding continuous time system with pure time delays. The third comparison is with an open-loop plant in which the state is reset to r after each sample is collected. The simulation allows the determination of the *inherent variability* of the plant output due to the combination of discrete control and continuous disturbances. The *additional variability* of a sampling and control method is defined as the difference between the method's σ_y and the inherent variability. Of course, the additional variability can be negative in any one simulation due to fortuitous random occurrences.

Trace-Based Simulations. Preliminary simulations were also conducted on a small set of relatively short RTT traces. The trace-based simulations should be extended but preliminary results are included here for general reference.

6 Evaluation

The simulations was continuously run for 2 weeks of simulated time. Table 1 reports summary statistics for the whole simulation and Figure 6 shows a five second snapshot of the plant output in the middle of the simulation. Figure 6(c) shows that the open-loop output $|y(t)|$ can be relatively large for a long period of time. The open-loop output $y(t)$ is mostly affected by the well-known behavior of Brownian motion (state disturbance), which can take the state away from the reference for long transients. By contrast, both the play-back and the ideal controller react when

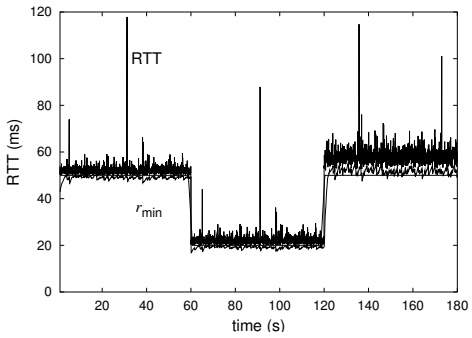

Figure 7: Estimate of the minimum round-trip time ($T = 50$ms). In the first minute, the delay distribution is the default. In the second minute, $\text{RTT}_{\min} = 20$ms. In the third minute, $r = 8$.

the state starts to drift, and rapidly bring the plant back to the reference. In the long run (Table 1), the play-back σ_y is close to that of the ideal controller. The play-back removed 75% of the additional variability of the open-loop output, which is comparable to the performance under pure delays. The play-back has smaller σ_y than the pure delay scenario primarily due to the use of an observer. Similarly, the 99-percentile and max $|y|$ are close to that of the ideal pure delay scenario. If the network conditions deteriorate (packet losses, jitter), the play-back controller performance gracefully degraded and, as the levels of services approximated a network partition, the play-back performance progressively approached that of the open-loop plant (details omitted).

Although the plant output $y(t)$ is the primary concern of a play-back strategy and ultimately determines its success, a play-back algorithm can be better understood by isolating the impact of each algorithmic component, as discussed next.

Minimum RTT. The value of $r_{\min}$ should be a close approximation of the RTT but $r_{\min}$ should always underestimate the RTT (Section 4.3). Figure 7 shows the RTT and the $r_{\min}$ when the delay distribution is stable for one minute and it then changes abruptly and significantly. The value of $r_{\min}$ was typically fairly close to the RTT and underestimates it in all but 0.2% of the samples. In general, $r_{\min}$ showed fast adaptiveness to changes in the RTT distribution.

Play-Back Delay. The value of the play-back delay τ_i should be a close approximation of the RTT, but should not overestimate it. Figure 8 gives the difference $\tau_i - \text{RTT}_i$. In Figure 8(a), the play-back delay overestimates the round-trip time in all but 2.3% of the packets. However, the underestimation was severe in a few cases, mostly in correspondence to delay spikes. The τ_i value was slightly larger than the RTT when jitter is heavier (last minute of the simulation). It was often the case that $T < \tau_i$, which means that a control pipe was established. At the nominal sampling frequency (Figure 8(b)), the value of τ becomes progressively dominated by the cap $T + r_{\min}$, and so $\tau - \text{RTT}$ does not grow much. However, τ underestimated the RTT in slightly more cases (2.8%) and almost as significantly as for the larger value of T. When the sampling frequency is very high (Figure 8(c)), τ has shorter peaks both in the positive and in the negative direction, but the most significant phenomenon is that the RTT is underestimated in more than 88% of the cases. Consequently, the plant applies almost all signals as soon as they are received, effectively disabling the play-back buffer. This simulation also highlights the importance of playing back late signals [] or else almost all packets would be discarded.

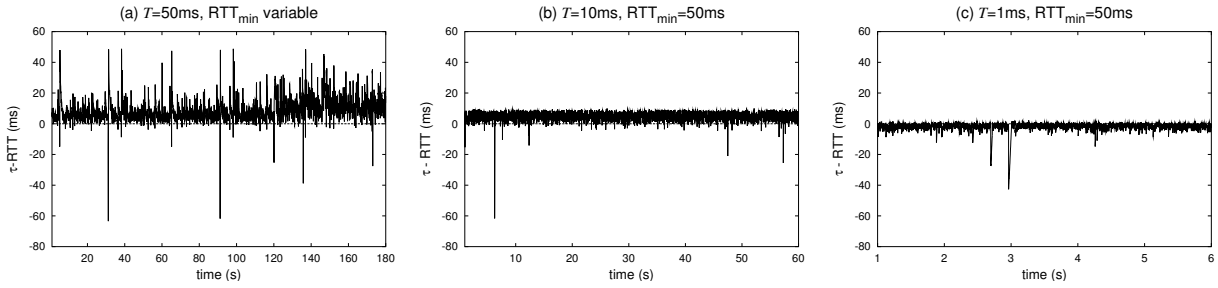

Figure 8: Difference $\tau_i - \mathrm{RTT}_i$ in the same scenario as in Figure 7 (left) and for $T = 10, 1\text{ms}$ (right).

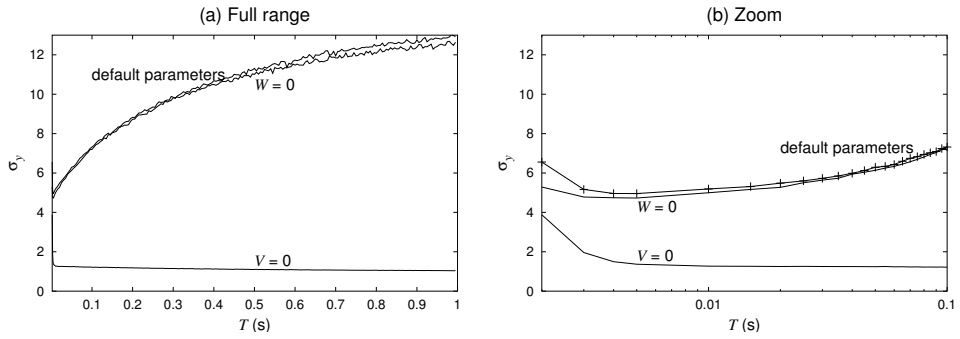

Figure 9: Standard deviation σ_y of the output y as a function of the sampling period T.

Sampling Period. Figure 9 shows σ_y as a function of T for the default case $V = 20$ and $W = 1$. For future discussion, the figure also gives the two cases $V = 0$, $W = 1$ and $V = 20$, $W = 0$.

If $V = 0$, the state evolves free of state disturbances. Since $x(0) = v(t) = 0$, if $u = 0$, then $x = 0$ always and $\sigma_y = W$. In other words, if $V = 0$, the best strategy would be to remove the controller or, which is equivalent, to set $T \to \infty$. The second case $W = 0$ is antithetic: the state grows progressively more unpredictable during a longer inter-sampling period. The optimal value of T should be chosen in the middle ground and depends on two factors. First, T depends on the relative strength V/W of the disturbances. If $W \gg V$, then long sampling periods are better (e.g., Figure 9, $V = 0$) and, conversely, if $W \ll V$, then short sampling periods are better (e.g., Figure 9, $W = 0$).

The second consideration is that the standard deviation of the output is significantly larger at the smallest values of T (leftmost side of Figure 9(b)) and, consequently, such values should be avoided. In general, the output variability σ_y was significantly larger when T is comparable to the RTT jitter. Figure 10 shows the standard deviation of y as a function of T. The first chart increases jitter with higher values of r and constant λ. The second chart increases jitter with lower values of λ and constant r. In both charts, T is normalized to the standard deviation of the gamma distribution (delay body). If T is small, τ was fundamentally defined by $T + r_{\min}$, and if T is less than the jitter, the nominal activation time $t' + \tau$ is likely to precede the reception of the signal by the plant, so that the buffer is effectively disabled. Consequently, the actual application times and the plant evolution are relatively unpredictable. The simulations suggest a value of T which is roughly 5 times the standard deviation of the delay distribution body. A possible justification is that, numerically, the probability that a Gamma random variable exceeds the mean plus 5 times the standard deviation is less than 1% for all $r > 0$. Figure 11 summarizes the imperfections of

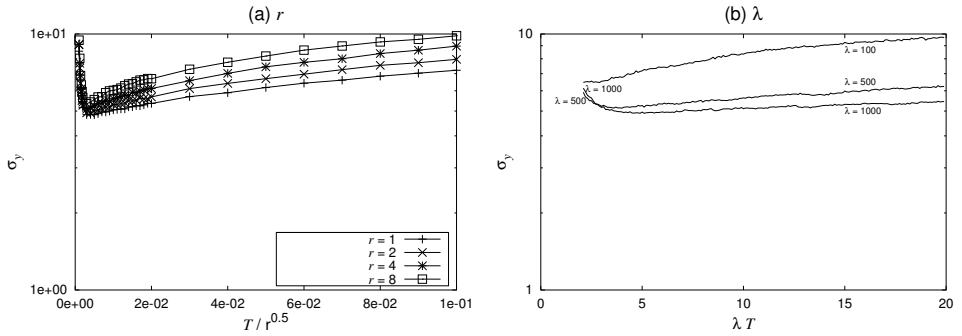

Figure 10: Standard deviation σ_y of the output y as a function of the sampling period T for various jitter levels.

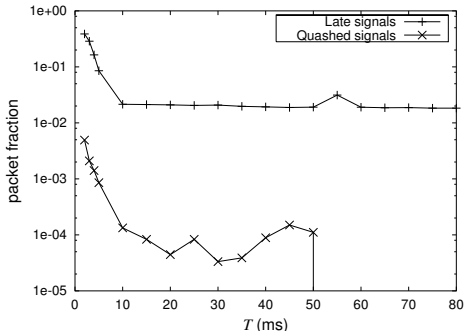

Figure 11: Imperfections of the control pipe as a function of the sampling period T.

the control pipe as a function of the sampling period T. Late signals are almost 65% of all packets when $T = 2$ms, but they drop rapidly and the curve shows a knee in correspondence to $T = 10$ms. The number of quashed signals is always less than 1% and decreases with larger sampling rates. The low incidence of quashed signals depended on the introduction of the upper bound $T + r_{\min}$ on τ (details omitted). In summary, if T is too small, the control pipe is unpredictable, mostly due to the fact that the buffer is effectively disabled and signals are applied late. However, if T is large enough, the pipe is fairly stable and little incremental predictability derived from further increasing T. The value of T should be larger than the round-trip jitter, and it should otherwise be chosen so as to balance the opposite effects of output and state disturbances.

This paper has assumed that T is a hardwired constant, as in most discrete-time plants. However, the sampling period T was found to be tied to the jitter, and we feel that future work should address the dynamic and adaptive setting of the sampling rate. In preliminary results, we have extrapolated the following consequences of the rule of thumb that T should be about 5 times the standard deviation of the delays. First, it is sometimes as easy to control a unit overseas as to control one that is nearby. The reason is that while long-haul delays tend to be larger, their jitter can be smaller, thus affording a smaller value of T and a finer grained control. Of course, this assumption is predicated on a model of the plant that is accurate enough for longer predictions, i.e., the observer needs an accurate value of the plant constant a. On the basis of route information, the intuitive reason is that our overseas paths have a similar number of hops as more local paths, and hence, while the delay increases, the jitter does not necessarily increase.

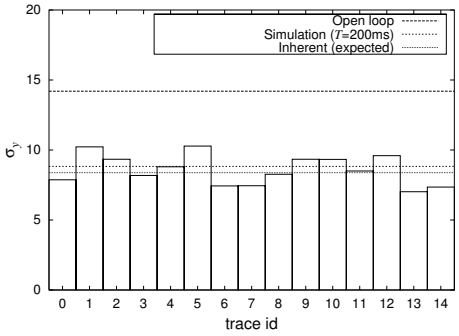

Figure 12: Standard deviation σ_y of the output y on trace-based simulations.

Trace-Based Simulations. Figure 12 gives preliminary results on a set of 15 RTT traces. The figure also gives the σ_y value for the open-loop plant and the expected value of the inherent variability. The figure reports for comparison the σ_y value obtained from simulations (Figure 9) even though the $\mathrm{RTT}_{\min}$, jitter, and drop rate differ sometimes significantly in the simulation and in the traces. In spite of these differences, σ_y hovers around the simulation value. The σ_y value is also close to the expected value of the inherent variability. The traces are collected at $T = 200$ms, and so these results suggest that comparably better performance would arise with faster sampling periods.

7 Related Work

Networked control is a relatively new field but already has a wealth of results [15] and is related to the area of real-time embedded distributed systems (e.g., [19]). The networked control model in Section 2 follows the original assumptions in which sensors are time-driven whereas controllers and actuators are event-driven [22]. The paper has focused on proportional controllers but, naturally, many other types of controllers exist [7, 23]. Networked control has been intensively investigated in the local area context, for example for the purpose of factory or building automation (e.g., [33] and references therein). This paper focuses on network vagaries that are more likely to occur on wide-area networks. Co-simulation for co-design is a novel approach in which the design of the control system and the design of the network are tackled together, and therefore should be analyzed (and simulated) together [4]. A taxonomy of real-time applications introduces the concepts of tolerance and adaptiveness. Moreover, other real-time applications, such as VoIP or video streaming, also use end-point play-back to achieve higher levels of tolerance or adaptiveness or both [26]. Since the controller sets an expiration date to the regular control, it is also suitable for adoption in event-driven sampling, where a sensor sends a sample only when the output has changed by at least a certain threshold value [2, 12]. Event-driven sampling is particularly suitable for energy savings in wireless networks. However, event-driven sampling creates an ambiguity for the more traditional controllers, which cannot ascertain whether a sample was dropped or simply not sent. The ambiguity is alleviated in the case of contingency control due to the presence of a fail-safe action $u_i^{(c)}$. Previous work on optimal loss compensation mostly addresses i.i.d. losses [3], which are not relevant in a Gilbert model.

8 Conclusions

In this paper, we have formalized the problem of making networked control tolerant and adaptive to network vagaries. In addition to the specific results, a main contribution of this paper is to formulate networked control in a way that is cleaner and more suitable for further Networks research. The paper also introduced novel performance metrics that are natural and appropriate for networked control. Although the focus has been on scalar plant, the same methods are generalizable to multi-dimensional plants.

The main contribution is a novel play-back algorithm for tolerant and adaptive networked control. The play-back algorithm is integrated with sampling and control. In the end, the algorithm involves a reverse play-back buffer and the estimation of play-back times, an expiration time to curb the regular aggressive deadbeat control, a contingency control to deal with loss episodes, and an observer that explicitly accounts for network vagaries. Extensive simulations showed that the combined approach was able to remove 75% of the additional (non-inherent) plant variability. Furthermore, the play-back control performed roughly as well as any proportional controller on an ideal network with pure delays. The algorithm was robust to parameter choice and its performance gracefully degraded when network connectivity worsened. The simulations show that the sampling period should be set as a function of jitter, and future work should address uneven sampling in networked control.

References

[1] A. Al-Hammouri, A. Covitch, D. Rosas, M. Kose, W. S. Newman, and V. Liberatore. Compliant control and software agents for Internet robotics. In *WORDS*, 2003.

[2] Karl Johan Åström and Bo Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. In *Preprints 14th World Congress of IFAC*, volume J, pages 301–306, Beijing, P.R. China, July 1999.

[3] Babak Azimi-Sadjadi. Stability of networked control systems in the presence of packet losses. In *CDC*, 2003.

[4] M. S. Branicky, V. Liberatore, and S. Phillips. Co-simulation for co-design of networked control systems. In *American Control Conference*, 2003.

[5] Roger Brockett. Stochastic control. Course Notes, Harvard University, 1992.

[6] K. Chandy and L. Lamport. Distributed snapshots: Determining global state of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, February 1985.

[7] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems.* Prentice-Hall, Upper Saddle River, NJ, 2001.

[8] H. Ekstroem and R. Ludwig. The peak-hopper: A new end-to-end retransmission timer for reliable unicast transport. In *IEEE Infocom*, 2004.

[9] Ken Goldberg, editor. *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet.* MIT Press, Cambridge, MA, 2001.

[10] Justin R. Hartman. Networked control systems co-simulation for co-design: Theory and experiments. Master's thesis, Case Western Reserve University, 2004.

[11] Gunnar Karlsson, Henrik Lundqvist, and Igacio Más Ivar. Single-service quality differentiation. In *IWQoS*, pages 265–272, 2004.

[12] M. Lemkin et al. Velocity estimation from widely spaced encoder pulses. In *American Control Conference*, June 1995.

[13] V. Liberatore et al. IP communication and distributed agents for unmanned autonomous vehicles. In *AIAA-UAV*, 2003.

[14] Vincenzo Liberatore, M. Cenk Çavuşoğlu, Qingbo Cai, and Nikitha Kondapally. GiPSiNet: A middleware for haptic human-computer interaction. Submitted.

[15] Vincenzo Liberatore et al. Networked Control Systems Repository. `http://home.cwru.edu/ncs/`.

[16] D. Loguinov and H. Radha. Measurement study of low-bitrate Internet video streaming. In *IMW*, November 2001.

[17] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[18] A. Mukherjee. On the dynamics and significance of low frequency components of Internet load. *Internetworking: Research and Experience*, 5:163–205, December 1994.

[19] D. J. Musliner et al. CIRCA: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), 1993.

[20] W. S. Newman et al. Design lessons for building agile manufacturing systems. *IEEE Trans. Robotics and Automation*, 16(3):228–238, June 2000.

[21] Marco L. Ngai, Vincenzo Liberatore, and Wyatt S. Newman. An experiment in remote robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2190–2195, 2002.

[22] J. Nilsson. *Real-Time Control Systems with Delays.* PhD thesis, Lund Institute of Technology, 1998.

[23] Katsuhiko Ogata. *Modern Control Engineering.* Prentice-Hall, Upper Saddle River, NJ, third edition, 1997.

[24] David A. Patterson. Latency lags bandwith. *Commun. ACM*, 47(10):71–75, 2004.

[25] B. Penaflor. Current status of DIII-D realtime digital plasma control. *IEEE Transactions on Nuclear Science*, 47(2):201–204, 2000.

[26] Larry L. Peterson and Bruce S. Davie. *Computer Networks.* Morgan Kaufmann, 2000.

[27] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *IEEE Infocom*, 1994.

[28] B. P. Robinson and V. Liberatore. On the impact of bursty cross-traffic on distributed real-time process control. In *Workshop on Factory Communication Systems (WFCS)*, 2004.

[29] Sheldon Ross. *A First Course in Probability.* Prentice Hall, Englewood Cliffs, NJ, fourth edition, 1994.

[30] Schulzrinne, Casner, Frederick, and Jacobson. RTP: A transport protocol for real-time applications. IETF Internet Draft, 2000.

[31] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems.* Springer, 2000.

[32] G. Varghese and A. Lauck. Hashed and hierarchical timing wheels: Efficient data structures for implementing a timer facility. *IEEE/ACM Transactions on Networking*, 5(6):824–834, December 1997.

[33] Wei Zhang. *Stability Analysis of Networked Control Systems.* PhD thesis, Case Western Reserve, August 2001.

[34] Yin Zhang, Nick Duffield, Vern Paxson, and Scott Shenker. On the constancy of Internet path properties. In *IMW*, 2001.

A Plant Dynamics

We integrate the differential equation (**??**) with a boundary condition on $x(\theta_i)$ to obtain

$$x(t) = \left(x(\theta_i) - \frac{bkx(\theta_i - \xi_i)}{a}\right) e^{a(t-\theta_i)} + \frac{bkx(\theta_i - \xi_i)}{a} . \tag{2}$$

In particular,

$$x(\theta_{i+1} - \xi_{i+1}) = \left(x(\theta_i) - \frac{bkx(\theta_i - \xi_i)}{a}\right) e^{a(\theta_{i+1}-\theta_i-\xi_{i+1})} + \frac{bkx(\theta_i - \xi_i)}{a}$$
$$x(\theta_{i+1}) = \left(x(\theta_i) - \frac{bkx(\theta_i - \xi_i)}{a}\right) e^{a(\theta_{i+1}-\theta_i)} + \frac{bkx(\theta_i - \xi_i)}{a} .$$

Define $\hat{x}_i = x(\theta_i)$. Observe that

$$\hat{x}_{i+1} = e^{a\xi_{i+1}} x_{i+1} + \frac{bkx_i}{a}\left(1 - e^{a\xi_{i+1}}\right) .$$

Hence, if $x_i = x_{i+1} = 0$, then also $\hat{x}_{i+1} = 0$, and, by Equation (2), $x(t) = 0$ for all $t \geq \theta_{i+1}$. Observe that $\Delta t_i = t_{i+1} - t_i = \theta_{i+1} - \theta_i - \xi_{i+1} + \xi_i$ and $\Delta s_i = \theta_{i+1} - \theta_i - \xi_{i+1}$. We now have

$$\begin{aligned}
x_{i+1} &= \left(e^{a\xi_i} x_i + \left(1 - e^{a\xi_i}\right)\frac{bkx_{i-1}}{a} - \frac{bkx_i}{a}\right) e^{a(\theta_{i+1}-\theta_i-\xi_{i+1})} + \frac{bkx_i}{a} \\
&= \frac{bk(x_{i-1} - x_i)}{a} e^{a\Delta s_i} + \left(x_i - \frac{bkx_{i-1}}{a}\right) e^{a\Delta t_i} + \frac{bkx_i}{a} \\
&= \left(e^{a\Delta t_i} - \frac{bk}{a} e^{a\Delta s_i} + \frac{bk}{a}\right) x_i + \frac{bk}{a}\left(e^{a\Delta s_i} - e^{a\Delta t_i}\right) x_{i-1} \\
&= \alpha_i x_i + \beta_i x_{i-1} ,
\end{aligned}$$

which is the homogeneous recurrence (**??**).

B Deterministic Deadbeat Control

An approach similar to the baseline can be used also when $\beta_i \neq 0$ is a constant. In this case, the jitter ξ_i is also constant, and we let $\xi_i = \xi$ for all i's. The intuitive interpretation is that either the constant RTT τ was estimated with an error or, as in [22, 33], the controller was designed for local operation but is being used even when delays are present in the feedback loop. In this case, the condition $\alpha_i = 0$ leads to

$$k'_d = \frac{a}{b} \frac{e^{aT}}{e^{a(T-\xi)} - 1} .$$

Proposition 1. *In the case of no losses, $\xi > 0$ and $k = k'_d$, the plant state converges to the reference $r = 0$ if and only if*

$$\xi < T - \frac{1}{a} \ln \frac{1 + e^{2aT}}{1 + e^{aT}} .$$

Proof. By design, k'_d implies $\alpha_i = 0$, so that $x_{i+1} = \beta x_{i-1}$. The characteristic equation of this recurrence is $\rho^2 = \beta$, and so the amplitude of x_i decreases with time if and only if $|\beta| < 1$. If $\xi > 0$, then

$$|\beta| = \frac{bk'_d}{a}\left(e^{aT} - e^{a(T-\xi)}\right) = \frac{e^{aT}}{e^{a(T-\xi)} - 1} e^{aT}\left(1 - e^{-a\xi}\right) = \frac{e^{2aT}}{e^{a(T-\xi)} - 1}\left(1 - e^{-a\xi}\right) .$$

Figure 13: The bound $\xi_{\max}$ on the jitter ξ as a function of the sampling period T (Proposition 1).

Figure 14: A case with jitter $\xi \neq 0$ that leads to a damped oscillation of the plant output.

Therefore, $|\beta| < 1$ occurs if and only if $e^{2aT}\left(1 - e^{-a\xi}\right) < e^{a(T-\xi)} - 1$, or $e^{-a\xi} > (1 + e^{2aT})/(e^{aT} + e^{2aT})$. The claim follows by taking the logarithm of both sides. □

Figure 13 shows the upper bound on ξ as a function of T for various values of a. In general, as the product aT increases, $\ln\left(1 + e^{2aT}\right) / \left(1 + e^{aT}\right) \sim aT$, and so little jitter can be tolerated with faster plants or slow sampling rates.

Corollary 2. *If $\xi > 0$ (undercompensation of delays), an oscillation is present in the plant output.*

Proof. If $\xi > 0$, then $\beta < 0$. The corollary follows from the characteristics equation derived in the proof of the previous result. □

Figure 14 shows the case $T = 0.2$, $\xi = 0.05$, $k_d' \simeq 6.88$, which leads to a damped oscillation in the plant output.

C I.i.d. Losses

If the sampling period is uniform and of unit length and the loss probability is $p < e^{-a}$, then Δt_i is a geometric random variable with parameter $1 - p$. Then,

$$M_T(a) = \frac{(1-p)e^a}{1 - pe^a} ,$$

and the difference

$$M_T(a) - e^a = pe^a \frac{e^a - 1}{1 - pe^a} > 0$$

is maximized by taking the derivative with respect to p. However, the derivative is $e^a(e^a - 1)/(1 - pe^a)^2$, which is always positive, and thus p should be taken as large as possible.

D Gilbert Model

The Gilbert model [34] (Figure 5) is a two-state Markov chain. The two states are labeled G (the "good" state) and L (the "lossy" state). Let p be the probability of remaining in the G state and q the probability of remaining in the L state. The chain generates a G or an L upon each state transition, where the jth symbol determines whether the control packet at step j was received at the plant (G) or not (L). For example, the string $GGLLG\ldots$ denotes losses at step 3 and 4, but good reception during the other steps. Assume even sampling and assume that time is normalized so that the sampling period is 1. Then, Δt_i is the length of the ith maximal subsequence of the form L^*G. For example, the string $GGLLG\ldots$ gives $\Delta t_i = \Delta t_2 = 1$ and $\Delta t_3 = 3$. At the end of each subsequence L^*G, the Markov chain is in the G state. Therefore, at the beginning of the next subsequence, the state is G with probability p and L with probability $1 - p$. If the initial state is G,

then $\Delta t_i = 1$. If the initial state is L, then $\Delta t_i = 1 + G$, where G is a geometric random variable with mean $1/(1-q)$. Therefore,

$$\overline{T} = E[\Delta t_i] = p + (1-p)\left(1 + \frac{1}{1-q}\right) = \frac{2-p-q}{1-q} ,$$

and, if $q < e^{-a}$

$$M_T(a) = E[e^{a\Delta t_i}] = p + (1-p)E[e^{a(G+1)}] = p + (1-p)e^a E[e^{aG}] = p + (1-p)(1-q)\frac{e^{2a}}{1-qe^a} .$$

Therefore,

$$\begin{aligned} k_L &= \frac{a}{b}\left(1 + \frac{1}{M_T(a) - 1}\right) \\ &= \frac{a}{b}\left(1 + \frac{1-qe^a}{p - pqe^a + (1-p)(1-q)e^{2a}}\right) \\ &= \frac{a}{b}\frac{(1-p)(1-q)e^{2a} - q(1+p)e^a + (1+p)}{(1-p)(1-q)e^{2a} - pqe^a + p} . \end{aligned}$$

E Effective Sampling Rate

The gain k_L is a function of $M_T(a)$, which in turn can be expressed as a McLaurin series of its derivatives $M^{(n)}(0) = E[(\Delta t_i)^n]$ calculated at the origin. Thus, k_L accounts for all of the moments of the distribution of Δt_i. However, if $M_T(a) \simeq M_T(0) + aM'_T(0) = 1 + a\overline{T}$, then

$$k_L \simeq k'_L = \frac{1}{b}\left(a + \frac{1}{\overline{T}}\right) .$$

The error in the approximation of $M_T(a)$ is $a^2 M''(\hat{a})/2 = a^2 E[(\Delta t_i)^2 e^{\hat{a}\Delta t_i}]/2 \geq 0$ for some $0 < \hat{a} < a$, and so $k'_L \geq k_L$. The control k'_L has the advantage of depending only on the effective sampling rate $1/\overline{T}$ rather than on the entire distribution of Δt_i. However, k'_L can only be applied if the error is small. Furthermore, k'_L can take a value $k'_L > k_1$ even when k_L does not exist. For example, in the case of an exponential distribution, $k'_L = (a+\lambda)/b$ even when $\lambda < a$. The effective sampling period $\overline{T}$ was the only factor considered in previous work [10, 33], with conclusions that were often difficult to explain analytically.

F Jitter

Proof of Proposition **??**. We first claim that α_i is independent of x_i. The quantity α_i is a function of constant terms and of the random variable ξ_i, which represents the jitter or the time difference between $t_i + \tau$ and θ_i. Meanwhile, x_i is the state at time $t_i + \tau$ and thus it is not affected by a future jitter ξ_i. By the same token, β_i is independent of x_{i-1}. Therefore, $E[x_{i+1}] = E[\alpha_i]E[x_i] + E[\beta_i]E[x_{i-1}] = E[\beta_i]E[x_{i-1}]$. Moreover, since the ξ_i's are i.i.d random variables, so are the β_i's, and so

$$\lim_{i\to\infty} x_{2i+1} = \lim_{i\to\infty} x_1 E\left[\prod_{j=1}^{i} \beta_{2j}\right] = \lim_{i\to\infty} x_1 \prod_{j=1}^{i} E[\beta_{2j}]$$

converges to $r = 0$ if and only if $|E[\beta_{2j}]| < 1$. Since the β_i's are identically distributed, the same condition also guarantees that $\lim_{i\to\infty} x_{2i} = 0$. We now have that

$$\begin{aligned} E[\beta_i] &= E\left[\frac{bk_J}{a}\left(e^{a(T-\xi)} - e^{aT}\right)\right] \\ &= \frac{e^{aT}}{e^{aT}M_J(-a) - 1} e^{aT} E[e^{-a\xi_i} - 1] \\ &= \frac{e^{2aT}}{e^{aT}M_J(-a) - 1}(M_J(-a) - 1) \ . \end{aligned}$$

Therefore, $E[\beta_i] > -1$ if and only if $M_J(-a) > (e^{2aT}+1)/(e^{2aT}+e^{aT})$, which proves the claim. □

Proof of Corollary **??**. It can be easily verified that $e^{aT}M_J(-a) - 1 > 0$ when $M_J(-a) > (e^{2aT} + 1)/(e^{2aT} + e^{aT})$. □

This section gives a sufficient condition for convergence in the presence of jitter.

Lemma 3. *If* $|\beta_i| \le \beta < 1$, *then* $\lim_{i\to\infty} E[x_i] = 0$.

Proof. We have that $|x_{i+1}| = |\beta_i||x_{i-1}| \le \beta|x_{i-1}|$, so that $E[|x_{i+1}|] \le \beta E[|x_{i-1}|]$. Therefore, $\lim_{i\to\infty} E[x_i] = 0$ if $\beta < 1$. □

In turn, such value of k_T and the condition that $|\beta| < 1$ will effectively put a bound on ξ_i and T.

Proposition 4. *If there is a* $\beta < 1$ *with*

$$0 \le \xi_i \le -\frac{1}{a}\ln\left(1 - \beta\frac{e^{aT}M_J(-a) - 1}{e^{2aT}}\right) \qquad (i \ge 1) \ ,$$

then x_i *converges to the reference* r *in the expectation.*

Proof. The condition on ξ_i ensures that $-1 \le \beta \le \beta_i \le 0$, and the proposition follows from the previous result. □

The proposition states that larger jitter ξ_i can be tolerated with shorter values of T.

G Contingency Control

It remains to establish the value of k, for which we first need to determine the plant dynamics.

Consider an interval $[\zeta_{i-1}, \theta_i)$. Since $u_i^{(c)} = 0$ is applied continuously in this interval, the plant is described by the equation $\dot{x}(t) = ax(t)$, which is integrated to give $x(t) = x(t_i + \tau)e^{a(t-t_i-\tau)}$ for all $t \in [\zeta_i, \theta_i)$. In particular, $x(\theta_i) = x(t_i + \tau)e^{a\xi_i}$. In the interval $[\theta_i, \zeta_i)$, the plant is described by the equation $\dot{x}(t) = ax(t) - bkx(t_i + \tau)$, which integrates to

$$x(t) = \left(x(\theta_i) - \frac{bkx(t_i+\tau)}{a}\right)e^{a(t-\theta_i)} + \frac{bkx(t_i+\tau)}{a} = x(t_i+\tau)\left(e^{a(t+\xi_i-\theta_i)} - \frac{bk}{a}\left(e^{a(t-\theta_i)} - 1\right)\right) \ .$$

Therefore,

$$x(\zeta_i) = x(t_i+\tau)\left(e^{a(T-\tau)} - \frac{bk}{a}\left(e^{a(T-\tau-\xi_i)} - 1\right)\right) \ ,$$

Model	Parameter	Values
Plant	a	3
	b	1
	$\overline{T}$	20ms
Network	τ	10ms
i.i.d. loss	p	$0.01, 0.02, 0.05$ $0.1, 0.15, 0.2, 0.5, 0.6$
Exponential loss	λ	5
Gilbert loss	p	$0.99, 0.98, 0.95,$ $0.9, 0.8, 0.5, 0.4$
	q	$0, 0.1, 0.2, 0.5$
Gamma jitter	λ	200, 500, 1000
	r	1, 2, 3

Table 3: Parameters of systems and networks in simulation based on probabilistic network models.

and

$$E[x(\zeta_i)] = x(t_i + \tau)\left(e^{a(T-\tau)} - \frac{bk}{a}\left(e^{a(T-\tau)}M_J(-a) - 1\right)\right) ,$$

which can be set to 0 by choosing k equal to

$$k_c = \frac{a}{b}\frac{e^{a(T-\tau)}}{e^{a(T-\tau)}M_J(-a) - 1} .$$

Suppose that $\xi_i = 0$ and that $x(t_i + \tau)$ is estimated with an error, due either to disturbances or to an imprecise system model. Then, the control u_i becomes $u_i = -k_c(1 + \epsilon)x(t_i + \tau)$ for some error term ϵ. The controller k_c makes $|x(\zeta_i)| < x(t_i + \tau)$ if and only if $-e^{a(T-\tau)} < \epsilon < e^{a(T-\tau)}$, which puts a bound on the combination of error, plant speed, and sampling period.

H Methodology

H.1 Simulations

In the first set of simulations, packet losses and delays were generated from various probabilistic models. For example, one such model would assume that there is no jitter and packet losses are generated with a Gilbert model, as in an example in Section **??**. In general, all probabilistic models in Section **??** were simulated (Table 3). Model-based simulations can be used to assess the accuracy of the dynamic estimators defined in Section **??**. The estimators should induce a plant output that is close to the output obtained with a controller that uses the exact value of the moment generating function for the same probabilistic model. The evaluation of the estimators starts with an initial unit value for the moments, so that the stochastic compensation is initially disabled. Thus, the simulation tested implicitly the simultaneous convergence of the plant and of the moment estimator.

H.2 Bounds on a

The distribution parameters λ and r are given in Table 4 and 5 and were estimated as $\lambda = \mu/\sigma_e^2$, and $r = \lambda\mu$, where μ is the average observed jitter and σ_e^2 is the sample variance of the jitter.

Name	a_L	$a_L^{(G)}$	λ	r	$a_{J,\min}$	$a_{J,\max}$
wopr.cwru.edu	∞	∞	70557.3	2.5	1460	33.6514
weatherhead.cwru.edu	∞	∞	52324.9	1.83	1485	33.7086
scp.grc.nasa.gov (Oct 2)	27.15	14.98	1575.3	3.00	27.14	16.9387
scp.grc.nasa.gov (Sep 30)	6.9	6.49	198.62	1.03	10.11	13.0512
www.ri.cmu.edu	15.65	12.55	1114.7	1.74	33.25	17.7509
www.comau.com	22.05	14.54	329.15	1.44	11.94	13.6788
www.icra-iros.com	12.7	6.44	945	1.28	38.5	18.3321
control-lab.et.tudelft.nl	13.25	12.7	309.71	1.28	12.71	13.9109

Table 4: Statistical and control-theoretical characteristics of the collected traces (Year 2002).

Name	a_L	$a_L^{(G)}$	λ	r	$a_{J,\min}$	$a_{J,\max}$
wopr.cwru.edu	∞	∞	27952.4	1.05	1550	33.3984
weatherhead.cwru.edu	∞	∞	356.7	0.10	1.81	24.9346
scp.grc.nasa.gov	28.88	13.20	46.5	1.39	3.44	6.1942
www.ri.cmu.edu	42.59	∞	2652.0	0.73	101.3	24.8884
www.comau.com	20.34	21.72	75.9	1.10	4.41	9.0900
www.icra-iros.com	20.90	∞	59.4	0.25	0.78	14.1837
control-lab.et.tudelft.nl	39.12	∞	91.9	0.06	0.30	21.7209

Table 5: Statistical and control-theoretical characteristics of the collected traces (Year 2004).

The bounds on a are predicted from the probabilistic models after they were fit to the traces and, as such, the bounds could differ from those obtained in a complete simulation. In particular, M_J is the moment generating function of the gamma distribution that was fit to the body of the jitter distribution.

I Probabilistic Evaluation

Figure 15 is an example of certain effects that are especially visible with high loss rates. The figure shows the plant output for $p = q = 0.5$ (i.e., an overall drop rate of 50%). The value of q is close to the critical threshold $q < e^{-aT} \simeq 0.55$, which is in the range where k_L and k_d differ more significantly. In the first place, the estimator is substantially worse than the exact $M_T(a)$, which follows the general trend for these large drop rate, as discussed above. The figure also compares the k_L and k_d controllers. The k_L controller has lower overshoot but larger rise time than k_d. The reason is that the k_d controller is designed for the ideal case of no losses and it severly suffers during periods of poor connectivity, so that it swings widely above and below the reference. However, the controller immediately snaps back to the reference as soon as connectivity resumes. By contrast, the k_L controller attempts to keep a behavior that balances between the periods of poor and good connectivity. The relative behavior of k_d and k_L basically held for all values of p and q, even though the stochastic nature of the network model occasionally resulted in anomalous outcomes with low probability. At any rate, the behavior of k_d and k_L justifies intuitively contingency control, in that contingency attempts to combine the good features of the former during good connectivity periods

Figure 15: Plant output for a network model with no jitter and losses generated by a Gilbert model with $p = q = 0.5$.

Figure 16: Convergence with no jitter for the case of no losses ($k = 7 \neq k_d = 6.47$) and of a constant loss probabilty of 10% ($k = k_L$).

and of the latter during bad connectivity. Finally, the estimator $M_{T,i}$ had lower overshoot than k_d even though estimators were not particularly accurate at these high drop rates.

J Evaluation

J.1 Probabilistic Network Models

The first set of simulations involves the generation of jitter and packet losses according to the various probabilistic models. The simulation confirmed the importance of setting the correct value for the gain k: for example, Figure 16 shows that a 10% loss rate with $k = k_L$ has comparable performance to the case when no packet is lost but k is not exactly equal to k_d. The dynamic estimation of $M_T(a)$ produced results comparable to those obtained with a prior knowledge of $M_T(a)$. For example, dynamic estimation increased the overshoot by less than 1% in a simulation with a 20% i.i.d. loss probability. The plant output deteriorated more significantly only for higher loss rates. Similarly, the dynamic estimation of $M_J(-a)$ led to overshoot within 1% of the exact value, and the estimate of $M_S(a)$ gave rise to behaviors analogous to those caused by a dynamic $M_T(a)$. Further detailed are given in Appendix I.

J.2 Contingency Control

In summary, the scp(9/30/02) trace with $a = 7$ and $D = e^{-2a}$ led to:

- The baseline k_d swang between $\pm 9 \cdot 10^5$.
- The contingency controller had 4 short spikes through a 15 minute simulation and the spike intensity ranged as $-1.6 \leq y \leq 0.7$. Its estimate of $M_J(-a)$ was close to 1 and no significant differences were introduced if the term was set to 1 throughout the simulation.
- The k_d controller had more spikes of large intensity (from $-11 \cdot 10^4$ to $5 \cdot 10^3$) even with the addition of a reverse play-back.
- The k_L controller (no play-back buffer) had significant spikes ($\pm 2 \cdot 10^5$), its rise time was 30 times longer than k_c (6 seconds versus 0.2 seconds), and it typically took longer to recover from a spike, as shown for example in Figure **??**.
- The k_L controller improves with the addition of a play-back buffer: it had a few spikes of intensity between -8 and 5. However, its behavior during these spikes resembled that of k_L with no buffer (Figure **??**), and its rise time was longer than k_L (7 seconds).