Integrated Play-Back, Sensing, and Networked Control

Vincenzo Liberatore
Division of Computer Science
Case Western Reserve University
10900 Euclid Avenue
Cleveland, Ohio 44106-7071, USA
E-mail: `vl@case.edu`
URL: `http://vincenzo.liberatore.org/NetBots/`

***Abstract*— Sensing, actuation, and decision units can control a remote physical environment and enable physical actions regardless of distance. However, the effectiveness of networked control depends on its ability to tolerate network non-determinism, which in turn can be enhanced by the use of play-back buffers. Although play-back has been intensively studied in multi-media applications, play-back schemes differ significantly in networked control, which is characterized by different performance metrics and a different sequence of communication events. The primary contribution of this paper is an end-to-end algorithm that integrates play-back buffering, sensor sampling, and control. The algorithm is extensively validated on simulations and real-time wide-area emulations. The integrated algorithm canceled the effect of disturbances as much as a proportional controller under ideal network conditions.**

I. Introduction

Sensing, actuation, and decision units can control a remote physical environment (Fig. 1). Distributed sense-and-respond systems enable physical actions to take place regardless of distance, with fundamental epistemological and social consequences [8]. Applications include, for example, industrial automation (e.g., [26]), distributed instrumentation (e.g., [1]), unmanned vehicles (e.g., [11]), home robotics (e.g., [19]), distributed virtual environments [13], power distribution [24], and building structure control [30]. Sense-and-respond operates in real physical time and late or missing signals can jeopardize the stability, safety, and performance of the controlled physical environment. A central issue is to bridge the timeliness requirements of networked control with the non-determinism of a communication network.

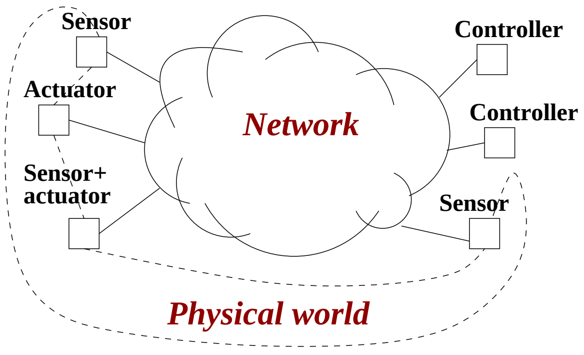

Fig. 1. The *networked control* approach.

The primary contribution of this paper is an end-to-end algorithm that integrates play-back buffers, sensor sampling, and control. The algorithm objective is to enable end-points to adapt to varying levels of network service. The end-point play-back method smooths out packet losses, delays, and jitter by applying control signals to the physical environment during predictable time intervals. By its nature, the play-back must be also integrated with the sampling and control processes that ultimately govern the physical dynamics. In other words, the algorithm takes on the intricate interaction of requirements stemming from non-deterministic networks and the target physics.

Sense-and-respond can partly borrow from multimedia play-back. However, sense-and-respond differs substantially from multimedia applications in behavior and objectives, and play-back strategies will differ as well. First, sense-and-respond performance is based on the real-time behavior of the controlled physical system, and there is in general no clear correspondence between the properties of the physics and multimedia replay quality. A critical difference is thus the integration between buffering and the underlying physics. Second, in multimedia applications, the play-back delay can often be determined upon the reception of a signal, whereas in sense-and-respond play-back intervals are always determined in advance because they affect the computation of the control signal. Consequently, certain multimedia play-back schemes cannot be immediately applied. For example, multimedia play-back sometimes employs the strategy of following delay spikes [25], but the controller can in general detect a spike only after an RTT, so that the original algorithm cannot be applied directly. Another difference is that most distributed multimedia are concerned with one- way delays, whereas sense-and-respond is primarily based on round-trip times. Play-back is also related to the problem of estimating TCP's RTO [23]. The RTO tends to be conservative because of the tremendous cost of a TCP time-out. An equally conservative play-back delay would effectively introduce a large delay in the feedback loop, with well-known control-theoretical complications (for exam-

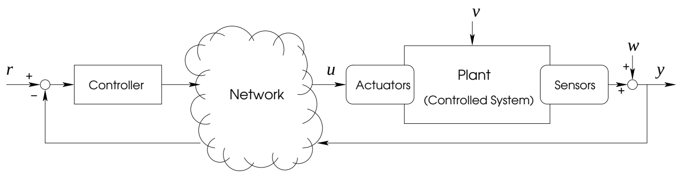

Fig. 2. Network-based control.

ple, the physical dynamics would be dominated by exogenous disturbances).

In other related areas, sensor networks are also concerned with quality of sensor data, but largely focus on the first part of the sense-and-respond loop. Furthermore, sensor networks tend to be more focused on energy savings than most sense-and-respond applications, in which total energy consumption is typically dominated by actuation. Finally, sense-and-respond has also been the subject of extensive control-theoretical work [14], which has revolved around the problems of scheduling packet transmission in closed networks, of stochastic compensation for network non-determinism, and of stability analysis for networked control systems. In general, control-theoretical approaches can exploit a buffer as a service provided transparently by an underlying communication channel.

Sec. II reviews the background on networked control. Sec. III introduces the play-back scheme. Sec. IV describes the evaluation. Sec. V reports on simulations and emulations. Sec. VI reviews related work and Sec. VII concludes the paper.

II. Networked Control: Background

The literature commonly denotes as *plant* the physical system to be controlled. A controller acquires data from the *sensors* that monitor the plant. The controller uses the sensor readings to issues commands to *actuators* so as to change appropriately the plant state.

Example 1: The textbook example is the regulation of ambient temperature. A thermostat (the controller) reads the temperature in the ambient (the plant) from a thermometer (the sensor) and modulates the heating (the actuator) so as to bring the room to the desired temperature.
In general, the plant state depends on the actuator operations and also on exogenous disturbances (e.g., people unpredictably open and close doors). Furthermore, the sensors are subject to measurement errors w_i. A *networked control system* involves a plant that is controlled remotely, and networked control is complicated by delays, jitter, and packet losses. In the first place, network latency can delay the reception of sensor data at the controller site and the application of a control signal at the plant site. Furthermore, packets can be lost either because they are dropped out in the network infrastructure or because they are discarded by the communication end-points, for example, if they arrive late.

Networked control is essentially the problem of reducing the uncertainty in the state of a system (Fig. 2). The plant is in state x_i at time t_i but, due to measurement errors w, x_i is known only approximately as a sensor reading y_i. The sample y_i is sent to the controller but y_i can be lost or be late. If y_i does reach the controller, it can be used to specify a control action u_i, but the signal u_i is either lost or is applied at an instant that is not known exactly due to delays and jitter. The plant state then evolves as a function of u_i and of a random disturbance $v(t)$. The goal is, for example, to make $y(t)$ equal to a perfectly specified set point r. In summary, the objective is to overcome the uncertainty due to network non-determinism, exogenous disturbances, and errors to bring the system to a known and desirable state.

Network latency can be partly mitigated by the use of an *observer*, a standard component that numerically simulates the plant to predict its state forward in the future [6], [21]. Observers have been extensively studied so as to recover to some extent from exogenous disturbances. However, prediction is subject to additional inaccuracies in the presence of delay jitter and packet losses. In the first place, the observer should extrapolate the plant state at the time when the control u_i is received by the plant, but the reception time is not deterministic. Second, the plant can receive outstanding control signals during the simulation interval, but the observer does not know the exact sequence and timing of these pending events. Another consequence of jitter is that it can prevent the use of large control signals because a large u_i can compromise the plant state if applied for an unpredictable interval. On the other hand, aggressive controllers would typically lead to closer tracking of a desired set point if delays were completely deterministic [6], [21]. On the good side, delay jitter can be eliminated with a *play-back buffer* that applies the control u_i exactly at time $t_i + \tau_i$ if u_i arrived before that time and drops the control if it was received too late. The τ_i value is called the *play-back delay*. The play-back delay τ_i must be carefully chosen. If τ_i is too large, various control-theoretical complications arise, such as, for example, the dynamics are dominated by disturbances and plant evolution becomes unpredictable. If τ_i is too small, the control signals would be lost and the plant cannot be controlled accurately. Therefore, an appropriate value of τ_i should strike a balance between delay and jitter.

Another consideration is that certain networked control systems involve plants that are computationally-constrained devices. In these cases, the plant algorithms should be simple, and the complexity should be moved to the controller as much as possible.

III. Play-Back and Sampling

A. Overview

The main contribution of this paper is an end-point algorithm that integrates play-back, sampling, and control. The intuition is that the control signals should be applied during a predictable interval. If the control is applied too early, the plant will not have reached yet the state for which the control was meant. Symmetrically, a control signal u_i can negatively affect a plant if it is applied continuously well after the sampling time t_i because the plant state would in general have changed significantly in the meanwhile.

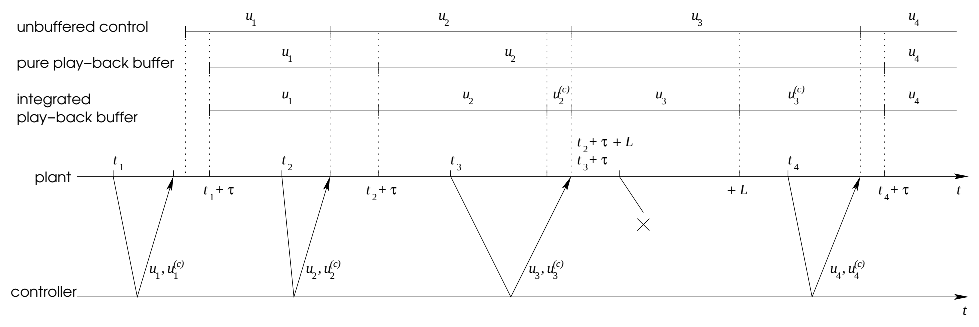

Fig. 3. The timeline of network events in the remote control of a plant. In this example, τ_i and $L_i = T$ are constants.

In our algorithm, 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 play-back time τ_i is used to implement a relaxed play-back buffer: the regular control is applied at time $t_i + \tau_i$, or immediately if the play-back time has already passed. Thus, the play-back delay only fixes the earliest time at which a control can be applied. After a duration L_i since the time when the control was applied, if a new control has not been received, the plant switches to the contingency action $u_i^{(c)}$. The duration L_i and the contingency control $u_i^{(c)}$ prevent the plant from applying the regular control for an unpredictably long period of time, thereby damaging the system. Therefore, the contingency control should be a low performance but presumably safe action. Additionally, the controller sends the plant a *sampling period* T_i, which denotes the time interval until the plant collects the next sample from the sensor and sends it to the controller. A predictable sampling schedule T_i effectively establishes time-out protection against losses.

Example 2: Fig. 3 exemplifies a possible signal exchange. In this example, the plant sends samples to the controller at regular intervals of length T. A sample was dropped out after t_3 but the plant kept sampling every T seconds and, in this example, the next signal was successfully delivered. The unbuffered control is commonly adopted in related work [14], [20] and applies received control signals from the time when they are received until the reception of a new signal. The other two methods hold the signal u_1, u_2, and u_4 that are received before their play-back time and apply the corresponding control signal only at the planned instant. The pure buffer discards u_3 because it is late and applies u_2 until otherwise instructed by a new control signal, whereas the final algorithm times out and switches to $u_2^{(c)}$ when u_3 is late and to $u_3^{(c)}$ when a sample is lost and no control signal is forthcoming.

Algorithm 1 describes the plant in pseudo-code, and Algorithm 2 describes the controller. The plant algorithm (Algorithm 1) basically implements the play-back and the contingency mechanisms, and it also collects statistics for future use by the controller. The plant algorithm is simpler than the controller's in that it avoids various numerical computations. 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 $(u, u^{(c)}, L)$ and $(u', u'^{(c)}, L')$. Additionally, the short plant memory simplifies the design of an observer (Algorithm 4, Line 8) At the controller site, the control law (Algorithm 2, Line 5) is parametric to the algorithms and can be designed with control-theoretical methods depending on plant characteristics and on control objectives. After this general overview, the details are discussed next.

B. Signal Discard

Both plant and controller discard signals unless they are received in-order. The intuition is that out-of-order controls are older than a previously received in-order control and thus less likely to be as effective. 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 [23], [29]). Moreover, the plant (Algorithm 1, Line 11) discards old signals and, specifically, a signal is discarded if its reception time is more than its expiration time $t' + \tau' + L'$. The assumption is that such a late signal does not contain any longer an appropriate control. Additionally, the play-back buffer has limited capacity and, if the buffer is full and a new control arrives, Algorithm 2 discards either the new signal or the previously buffered control. The algorithm discards the new signal if its application time $t' + \tau'$ is more than the current expiration value of T_P (Line 11 under the label of "quashed" signal). Intuitively, the newly received signal is discarded because the buffered signal is fresher (its application time is earlier). Moreover, quashed signals 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, where RTT_i is the ith round-trip time. In this case, signal $i+1$ resets the play-back timer T_P and effectively cancels the reception of the previous signal i. If this event occurs repeatedly, signals would cancel each other in turn and no control would be applied. This pathological scenario is avoided if a new signal must be replayed no later than the outstanding play-back delay in T_P.

Algorithm 1 Plant

```
Require: $T$ is initially set to the plant clock resolution
 1: Set timer $T_S$ to expire now
 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 a signal to the controller
 8:     Set $T_S$ to expire in $T$ seconds
 9:   else if an in-order control $(u', u'^{(c)}, t', \tau', L', T')$ is received then
10:     RTT-sample $\leftarrow$ current time$-t'$
11:     if $t' + \tau' > T_P$'s expiration time (quashed signal) and current time $\geq t' + \tau' + L'$ (old signal) then
12:       Discard the received signal
13:     else
14:       hiseqno $\leftarrow$ sequence number of the received control
15:       $T \leftarrow T'$
16:       if current time $< t' + \tau'$ then {Arm a play-back timer}
17:         Cancel $T_P$, and reset $T_P$ to expire at time $t' + \tau'$
18:       else {Play-back}
19:         $(u, u^{(c)}, L) \leftarrow (u', u'^{(c)}, L')$
20:         Cancel $T_L$ and $T_P$ and reset $T_L$ to expire in $L$ seconds
21:         Apply the control $u$
22:       end if
23:     end if
24:   else if $T_P$ expired then
25:     $(u, u^{(c)}, L) \leftarrow (u', u'^{(c)}, L')$
26:     Cancel $T_L$ and reset $T_L$ to expire in $L$ seconds
27:     Apply the control $u$
28:   else {$T_L$ expired}
29:     Apply $u^{(c)}$
30:   end if
31: end loop
```

Algorithm 2 Controller

```
1: loop {Main Loop}
2:   Read an in-order sample from the plant with $y$, the control action at time $t'$, and the local plant variable values (such as the current sampling interval $T'$) from the plant
3:   Update $\tau$ and $T$ as a function of RTT-sample and $T'$ (Algorithm 3)
4:   Predict $y(t' + \tau)$ with the observer (Algorithm 4)
5:   Calculate $u, u^{(c)}$, and $L$ as a function of $\tau$ and $T$
6:   Marshal $(u, u^{(c)}, t', \tau, L, T)$ into a signal to the plant
7:   Record the signal in the log $\mathcal{L}$
8: end loop
```

C. Play-Back Delays

The play-back delay τ mandates the nominal time when a control is applied as the plant input. The delay τ is computed at the controller because the controller needs it to generate u, $u^{(c)}$, and L. In principle, the value of τ could also be computed at the plant and sent to the controller, but with an additional computational load on the plant. In both cases, the play-back delay is computed before the control u reaches the plant.

Algorithm 3 is a modular component in the controller (Algorithm 2). It borrows from the recent peak-hopper method [7] its general structure, its initializations, and the values of the constant factors. The long-term component has been eliminated since the resulting τ was too conservative: although large RTO's are appropriate in the context of TCP to avoid unnecessary time-outs, the same large values would let disturbances dominate the plant's physical dynamics. The most noticeable addition is an upper bound on τ to avoid quashed signals. It is easy to see that signal $i+1$ will not be quashed if

$$\tau_i \leq T' + \mathrm{RTT}_{\min} , \tag{1}$$

where $\mathrm{RTT}_{\min}$ is the smallest round-trip time between plant and controller. If Eq. (1) is violated, the algorithm resets τ correspondingly (Line 18) and also adjusts the value of T so as to avoid the problem in the future (Line 17). Otherwise, the algorithm attempts to use a more aggressive sampling rate (Line 20) because faster sampling rates typically improve the performance of digital control [6]. The dynamic setting of τ and T requires an adaptive estimation of $\mathrm{RTT}_{\min}$. Furthermore, a slight underestimate of $\mathrm{RTT}_{\min}$ is preferable to an overestimate because Eq. (1) is an upper bound. The $\mathrm{RTT}_{\min}$ value is estimated with a symmetric version of the peak-hopper applied to find a lower bound on the RTT. In Algorithm 3, $r_{\min}$ is an adaptive estimate of the minimum round-trip, it is adjusted whenever a smaller round-trip time is detected, and it is 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. In turn, the C term is an exponential moving average of the negative variability and it is calculated with the same weights as in TCP's RTTVAR. It can be easily shown that, in Algorithm 3, $\tau - r_{\min}, T > 0$.

A final remark is that the dynamic setting of T is optional and can be replaced with a fixed sampling period, as in Fig. 3. However, a fixed T would introduce the burden of manual administration and does not necessarily avoid quashed signals.

D. Observer

The controller predicts the value of $y(t' + \tau)$ by invoking an observer (Algorithm 2, Line 4) [20]. In principle, the observer should predict the plant state by simulating its dynamics but,

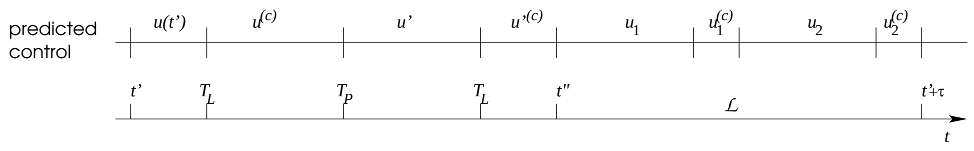

Fig. 4. Example of a timeline predicted by the observer, using the plant state $(u(t'), u^{(c)}, u', u'^{(c)}, T_P, L)$, and a log $\mathcal{L}$ containing $u_1, u_1^{(c)}, u_2, u_2^{(c)}$.

Algorithm 3 Play-Back Delay

```
Require: If available, $r_1$ is the new RTT sample, $r_0$ is the previous RTT sample, $T_{\min}$ is the plant timer resolution. Constant factors are adapted from previous algorithms and, in particular, $B$ is initially set to $0.25$ and $C$ is initially set to $-0.25$.
Ensure: Computes a new value of the play-back delay $\tau$ and the sampling interval $T$ in seconds.
 1: if there is no RTT sample then
 2:     $\tau \leftarrow 3$; $T \leftarrow$ plant nominal sampling time
 3: else if there is only one RTT sample $r_1$ then
 4:     $\tau \leftarrow 3r_1$; $T \leftarrow 2r_1$
 5: else if there are only two RTT samples $r_1$ and $r_0$ then
 6:     $\tau \leftarrow 1.25r_1$; $r_{\min} \leftarrow \min\{r_1, r_0\}$; $T \leftarrow \tau - r_{\min}$
 7: else
 8:     $\delta \leftarrow \frac{r_1 - r_0}{r_0}$ {Calculate current RTT variability}
 9:     $B \leftarrow \min\{\max\{2\delta, 0.9375B\}, 1\}$ {Update positive variability coefficient $B$}
10:     if $\delta < 0$ then {Update negative variability coefficient $C$}
11:         $C \leftarrow \max\{3C/4 + \delta/4, -1/2\}$
12:     end if
13:     $r' \leftarrow (1 + C)\min\{r_1, r_0\}$ {Update estimate $r_{\min}$ of RTT$_{\min}$}
14:     $r_{\min} \leftarrow r_{\min} < r'$ ? $r_{\min} + \frac{r' - r_{\min}}{16}$ : $r_{\min} \leftarrow r'$
15:     $\tau \leftarrow (1 + B)\max\{r_1, r_0\}$ {Update $\tau$ and $T$}
16:     if $\tau > T' + r_{\min}$ then
17:         $T \leftarrow \tau - r_{\min}$
18:         $\tau \leftarrow T' + r_{\min}$
19:     else
20:         $T \leftarrow T - \frac{T' - \tau + r_{\min}}{16}$
21:     end if
22: end if
23: $T \leftarrow \max\{T, T_{\min}\}$
```

in reality, the prediction is hampered by a variety of factors. The observer (Algorithm 4) uses known factors whenever they are available, and it resorts to assumptions otherwise.

The value of $y(t' + \tau)$ is determined by the state of the system at time t' and by its evolution in the interval $(t', t'+\tau)$. The system state at time t' obviously consists of the plant state x, but also of the value of the local variables in Algorithm 1. The evolution in $(t', t' + \tau)$ depends on disturbances that are inherently unpredictable but are often assumed to have 0 mean [6], so that $y(t' + \tau)$ will be estimated from a plant simulation with no disturbances. Furthermore, state evolution depends also on the messages that are in transit [5] and that will be received between time t' and $t' + \tau$. However, signal 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, their occurrence and timing cannot be predicted accurately. The observer must make an assumption and, for the most part, it assumes that outstanding in-order regular and contingency actions will be applied at their nominal time. Specifically, the controller maintains a log $\mathcal{L}$ of outstanding control signals (Algorithm 2, Line 7) 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 plant. First, the observer discards from $\mathcal{L}$ the entries of all messages whose sequence number is less than `hiseqno` (Line 1) because those messages either (a) have already been received and processed at the plant and therefore are already reflected in the plant state, or (b) will never be received, or (c) will be received in the future and discarded as out-of-order. Second, the observer times out old signals, 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. An example is shown in Fig. 4.

Algorithm 4 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: Estimate $y(t' + \tau)$ with $u(t)$ starting from $y(t')$
```

The next two paragraphs elaborate on the algorithm's prop-

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 G
p_{spike}	.99	Probability of a delay spike
q	0.2	Probability that the Gilbert model remains in the lossy state L
$\text{RTT}_{\min}$	50ms	Minimum RTT
$T_{\min}$	1ms	Shortest sampling period

TABLE I
PARAMETERS THAT DEFINE THE SYNTHETIC TRACES.

erties.

E. Pipes

The sampling rate $1/T$ normalized by the packet size is the instantaneous bandwidth used by networked control. Since T is related to bandwidth and τ to communication delays, T and τ are largely independent of each other. In particular, nothing prevents $T \ll \tau$, 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*. Notions similar to pipes were discussed in [15], [31]. Although the pipe is, in some sense, stored in the network, the plant (Algorithm 1) does not keep explicit state for the signals in the pipe.

An advantage of pipes was mentioned earlier (Sec. III-C): shorter sampling periods tend to improve control performance. 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 in the next round. If $t_{i+1} + \tau_{i+1} < t_i + \tau_i$, then u_i is discarded by the plant (Algorithm 1, Line 19 and 25). 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. However, Eq. (1) and Algorithm 3 still limit the sampling rate and, consequently, the pipe depth.

F. Implementation

Software is available on-line with source code, design documents, and reference and user's manuals. The code has been extensively tested in simulations and emulations. The software does not need real-time scheduling, but it can exploit it if it is available. Clock synchronization is unnecessary because the algorithm is based solely on RTT's.

IV. Experimental Methodology

A. Network

The algorithms are evaluated both on simulation and on real-time network emulations.

First, 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 [25], and a Gilbert model (Fig. 5) for losses [33]. The sensitivity to all parameters was investigated, but only an outline of the results is reported here.

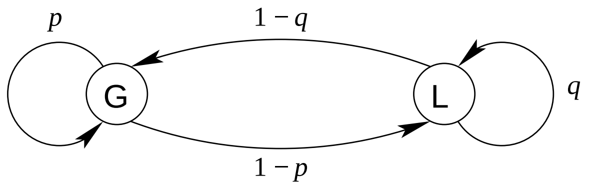
$$\begin{tikzcd} G \arrow[loop left, "p"] \arrow[rr, bend right, "1-p"'] & & L \arrow[ll, bend right, "1-q"'] \arrow[loop right, "q"] \end{tikzcd}$$

Fig. 5. The Gilbert model for packet losses [33].

A sample is lost if the Gilbert process is in the lossy state L (Fig. 5). 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 [18] with parameters $\text{RTT}_{\min}$, 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 default value of K is chosen so as to be sufficiently separated from the mode $(r-1)/\lambda$ and from the mean r/λ of the gamma distribution (body of the delay distribution). 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, an arbitrary 0.1ms) with probability p_{spike} to simulate a burst of almost simultaneous in-order arrivals [25], and it is left to its previously calculated value otherwise to simulate out-of-order delivery. The simulation parameters are summarized in Table I along with their default values.

Second, plants were remotely controlled in real-time over the Internet. In the choice of the end-hosts for the plant and the controller, we adopt the following principles. First, the algorithms were run only on machines where they were able to obtain either real-time status or a reasonable slice of system resources so that process timing was not skewed by resource starvation. Consequently, we ruled out PlanetLab [22] and resorted to independently obtained accounts. Second, we made an effort to capture a wide variety of heterogeneous conditions and we were successful in including disjoint end-to-end paths

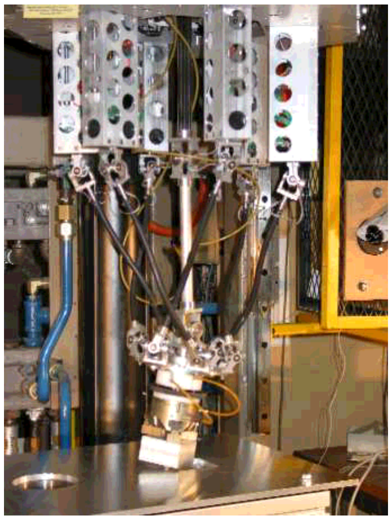

Fig. 6. The ParaDex robot is an Internet-controllable industrial manipulator whose reflex control interface is a scalar linear system (2) in each degree of freedom [1].

ranging from the same campus to overseas. We also repeated experiments at different times of the week and of the day.

B. Plants and Controllers

The primary evaluation environment is the class of scalar linear plants due to their generality and relative simplicity. However, results can be easily extended to higher-dimensional linear systems, as briefly sketched below. 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 scalar stochastic processes. The stochastic processes $v(t)$ and $w(t)$ model exogenous disturbances and noise. In a scalar linear plant, the state, input, output, and disturbances are related by the equations:

$$\left\{ \begin{array}{lcl} \dot{x}(t) & = & ax(t) + bu(t) + v(t) \\ y(t) & = & x(t) + w(t) \end{array} \right. , \tag{2}$$

where a, b are real numbers. In any one realization of Eq. (2), a and b are constants that represent the relationship between physical quantities in that particular plant. Eq. (2) is generic across applications because it models parametrically any physical system characterized exactly or approximately by a differential equation. Linear plants have numerous applications [1], [11], [26], [19] and any textbook on control engineering would present hundreds of examples (e.g., [6], [21]). 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 [32].

Example 3: The ParaDex robot (Fig. 6) is a manipulator with 6 degrees of freedom. In each degree of freedom, the interface of the NAC reflex controller is a scalar linear system (2), where x is the velocity in that direction, bu is the force applied by the manipulator's motors, and $a < 0$ can be initialized in software. Disturbances v and w are mostly due to unexpected contact forces. The ParaDex RPC interface was used to remotely control compliant manipulation tasks [1].

We will consider the objective of achieving $y(t) = 0$ in spite of disturbances and network non-determinism. A *deadbeat controller* attempts to obtain $y = 0$ by setting $L = T$ and $u = -ky$, where the feedback gain k is equal to

$$k = \left\{ \begin{array}{ll} \frac{a}{b}\frac{e^{aL}}{e^{aL}-1} & \text{if } a \neq 0 \\ -1/L & \text{otherwise} \end{array} \right. .$$

The deadbeat controller owes its name to the fact that, in the absence of disturbances, the system output converges to a reference set point in one sampling interval with no oscillation. If the actuator then stops ($u^{(c)} = 0$), in the absence of disturbances, the plant remains at the desired set point $y = 0$ [9]. The deadbeat controller and the corresponding contingency actions can be generalized to multi-dimensional linear plants via pole placement (details omitted). The deadbeat control was chosen to highlight two features of play-back. First, the deadbeat's nominal operations require that a second control be applied at a predictable time, and such a predictable timing is available with Algorithm 1 but not under an unbuffered controller. Second, the deadbeat controller is extremely aggressive, and so it highlights how timing predictability can support high-performance controllers.

The evaluation will use numerical simulations of plants. Although physical experiments are possible, numerical simulations are vastly more flexible and have been validated extensively over the course of the decades. Even though the plant is simulated, the real-time wide-area experiments (Sec. IV-A) are run over a real network. Simulations were executed for various values of a. The evaluation used only non-positive values of $a \leq 0$, which make the corresponding plant (2) stable because open-loop stability is required to achieve safe and fault-tolerant operations under network non-determinism. Several values of a were tried, and the default value is $a = -1$. We set the default value of $b = 1$, which basically normalizes the input units. A common model for $w(t)$ are independent normal random variables with mean 0 and variance W^2 [6]. In our evaluation, the default is $W = 1$, which basically normalizes the output units around the variability of the output disturbance. A common model for $v(t)$ is the formal derivative of Brownian motion and it is simulated as follows. The state $x(t)$ can be expressed as $x(t) = x_0(t) + x_N(t)$, where $x_0(t)$ obeys

$$\left\{ \begin{array}{lcl} \dot{x}_0(t) & = & ax_0(t) + bu(t) \\ x_0(t') & = & x(t') \end{array} \right. \tag{3}$$

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

$$\left\{ \begin{array}{lcl} \dot{x}_N(t) & = & ax_N(t) + v(t) \\ x_N(t') & = & 0 \end{array} \right. \tag{4}$$

The function $x_0(t)$ was integrated analytically for a generic piecewise linear $u(t)$ and calculated numerically by the plant

	Open-Loop	**Unbuffered**	**Pure Delays**	**Play-Back**
m_2	14.1657	5.8171	5.7105	5.2881
$\tilde{y}$	36.4254	14.9209	14.7639	13.7987
$\max \lvert y \rvert$	76.7715	74.6532	34.7771	32.3791

TABLE II
SUMMARY STATISTICS OF TWO WEEKS OF SIMULATED TIME. THE 99-PERCENTILE OF $|y|$ IS DENOTED AS $\tilde{y}$ ($S = 900$S). PURE DELAYS IS AN IDEAL (NON-IMPLEMENTABLE) BASELINE.

simulator. The stochastic process $x_N(t)$ was approximated by finite differences with the formula [4]:

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

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 default is $V = 20$ but several other values were tried as well. It is easy to see that the accuracy of the simulation requires that the step h be a small fraction of $\min\{1/|a|, 1/V^2\}$ and we found good accuracy with $h = \min\{1/|a|, 1/V^2\}/20$. The simulator uses the Mersenne twister to generate uniformly distributed pseudo-random numbers [17] and simulates Gaussian pseudo-random variables with the Box-Muller method [27].

C. 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. Summary metrics will be based on the time series $Y = \{y(iS) : i > 0\}$, which gives the plant output at regular intervals of length S seconds. The evaluation metrics will express the variability around the set point $y(t) = 0$ and, consequently, the methods' tolerance to disturbances and network non-determinism. The first metric is the root-mean-square sample output $m_2 = \sqrt{\sum_{y \in Y} y^2}$, which is a classical measure of the size of a signal [3]. Second, the 99-percentile of $|y|$ is the value $\tilde{y}$ for which $\Pr[|y| > \tilde{y} : y \in Y] \leq 0.01$ and gives an indication of the tail of the variability around the set point. Finally, we consider the maximum deviation $\max\{|y| : y \in Y\}$. In general, an important property is robustness because it directly impacts the safety and stability of a physical environment. Hence, performance metrics will be related to variations of experimental conditions, and especially to deteriorating connectivity, so as to assert whether a method's performance degrades gracefully.

Comparisons and benchmarks are complicated by the novelty of networked control [12]. For example, previous control-theoretical algorithms address disturbances, losses, delay, jitter, and pipes but not the combination of all of these factors [14]. In this paper, the evaluation benchmarks are adapted from more classical control theory, and the play-back controller is compared with three baselines. Two baselines can be physically implemented and the other one cannot, but is a useful term of comparison. The first baseline is the output generated by the *open-loop* plant (2) with $u(t) = 0$ and expresses the system behavior in the absence of any controller. The open-loop plant can obviously implemented in that it requires no communication and no controller. The open-loop scenario is as good as any controller in the case of network partitions and, conversely, a controller should aim to be no worse than open-loop under high loss rates. The second baseline is a closed-loop controller motivated by the observation that, in spite of network non-determinism, a play-back algorithm should regularize signal delivery. The algorithm is thus compared with a simpler proportional controller (fixed k, no observer, small fixed $T = 10$ms) but with the simulation of a perfect network with no losses, no jitter, and constant delays equal to $\text{RTT}_{\min}$. This second scenario is eminently ideal and will be denoted as *pure delays*. In the pure delay scenario, m_2 was minimized by $k = 11$, as this value struck a balance between the transfer functions of the two disturbances on the state and on the output. The gain $k = 11$ will be used in all our experiments. Incidentally, $k = 11$ leads to a gain margin of 9dB and a phase margin of $60°$ in the corresponding continuous-time system with constant delays. Pure delays can in principle be improved upon by the more aggressive deadbeat controller, but only if paired with buffers that ensure predictable timing, as discussed in Section IV-B. The *proportional unbuffered* baseline is an implementation of $k = 11$ with the unbuffered controller (Fig. 3, [20]).

V. EVALUATION

A. Observations

Default Simulation: Table II reports summary statistics for 2 weeks of continuous simulated time, and Fig. 7 shows a five second snapshot of the plant output in the middle of the simulation. The open-loop output $y(t)$ is mostly affected by the well-known behavior of Brownian motion (state disturbance $v(t)$), which can take the state away from the reference for long transients (Fig. 7(a)). By contrast, the controllers react when the state starts to drift, and rapidly bring the plant back to the reference (Fig. 7(b)(c)(d)). In the long run (Table II), all the measured play-back metrics are close to those under the ideal pure delay scenario and to the unbuffered controller.

Sensitivity: The simulations explored the sensitivity to all parameters. In general, if the network conditions deteriorate (packet losses, jitter), the play-back controller performance gracefully degraded and, as the levels of service approximated a network partition, the play-back performance progressively approached that of the open-loop plant. The unbuffered proportional controller also degraded but its performance was

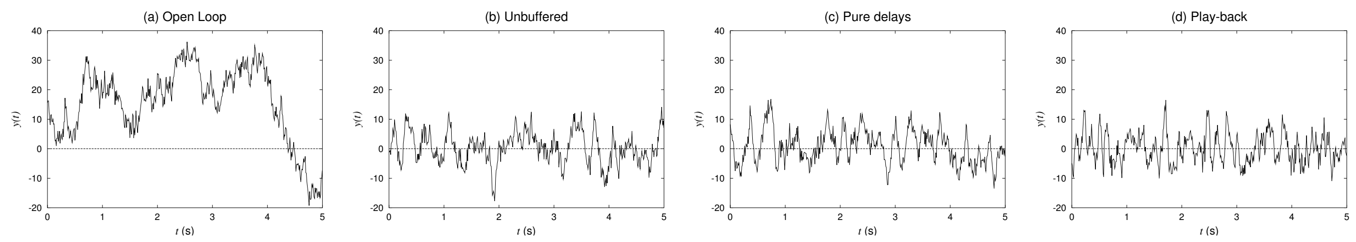

Fig. 7. Plant output $y(t)$ under the default simulation parameters (five second snapshot of Table II). Pure delays denote a proportional controller under ideal network behavior.

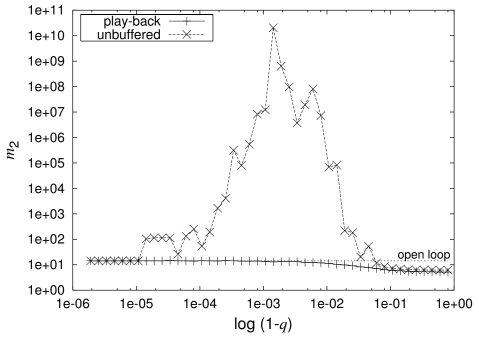

Fig. 8. Root-mean-square m_2 as a function of the probability q of remaining in the lossy state.

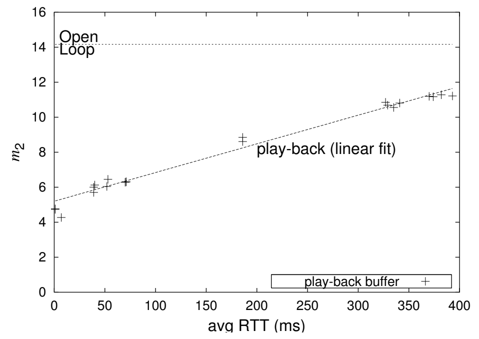

Fig. 9. Real-time wide-area experiments. The open-loop performance is ideal, cannot be implemented, but can be simulated (Table II). The pure delay scenario cannot be implemented in wide-area experiments. The unbuffered controller had comparable results only when RTT $\leq$ 60ms and small loss rates.

often significantly worse than that of an open-loop plant. For example, Fig. 8 shows m_2 as a function of the probability q of remaining in the lossy state. As q increases, the play-back controller smoothly degrades into an open-loop plant. However, the unbuffered m_2 becomes 10 orders of magnitude larger than the open-loop. Eventually, the loss probability is so severe that the proportional controller is effectively disconnected for the entire simulation and cannot cause any harm any longer. The unbuffered controller was also sensitive to longer RTT's and m_2 was practically infinite when $\text{RTT}_{\min} \geq 150$ms.

Wide-Area: As for the real-time wide-area experiments, results were close to those captured in simulations once the simulation parameters are set to the values measured in the corresponding wide-area experiment. Thus, wide-area evaluation validates simulations. Fig. 9 plots m_2 as a function of the average RTT (the charts obtained for metrics other than m_2 are qualitatively similar and omitted). The play-back m_2 gracefully degraded as the RTT increased. As for the unbuffered controller, in several wide-area experiments, it had practically infinite values of m_2 and $\tilde{y}$. The failure of the proportional controller was associated either with delays longer than 60ms or with high and bursty loss rates in spite of fairly contained RTTs (average RTT of 40ms). In the same scenarios, the play-back avoided any particular problem (i.e., $m_2 \simeq 6$).

Summary: The play-back algorithms resulted into higher reliability. It achieves ideal performance in the benign default scenario and, unlike the unbuffered controller, its performance gracefully degrades as communication conditions deteriorate.

B. Factors

The play-back behaviors can be better understood by isolating the impact of its components, as discussed next.

Play-Back and Expiration: A control signal u_i is applied during a time interval whose bounds are defined by the play-back delay and by the expiration time. The expiration time was especially effective in the case of losses in that it prevented the continued application of an old control that was appropriate at the time it was originally applied but that later on would take the plant output away from its reference value. Symmetrically, the play-back buffer prevented the application of a signal when it was not appropriate yet for the current plant state (numerical details omitted).

The buffer immediately plays back a packet received after the play-back time $t' + \tau'$ (Algorithm 1, Lines 18–22). An advantage of the relaxed play-back is that it does not introduce additional losses. The late replay improved performance, especially because it avoided occasionally large values of y. For example, Fig. 10 shows one such a spike at the very beginning of a wide-area experiment. The intuitive reason can be explained with Fig. 3: u_3 is not old (Algorithm 1, Line 11), and in fact it is only slightly late, and it is typically more helpful than the continued application of the contingency control $u_2^{(c)}$.

Minimum RTT: The value of $r_{\min}$ should be a close approximation of the RTT but $r_{\min}$ should also underestimate the RTT (Sec. III-C). For example, Fig. 11 shows the RTT

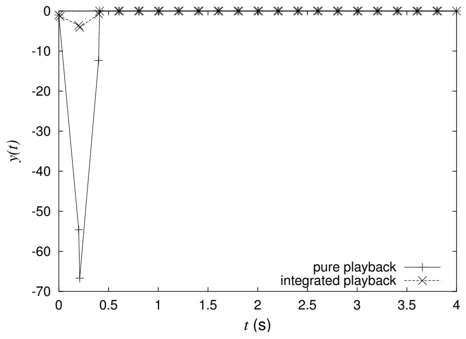

Fig. 10. Plant output $y(t)$ in a wide-area experiment. In this experiment $V = W = 0$ to highlight the effect of the buffers.

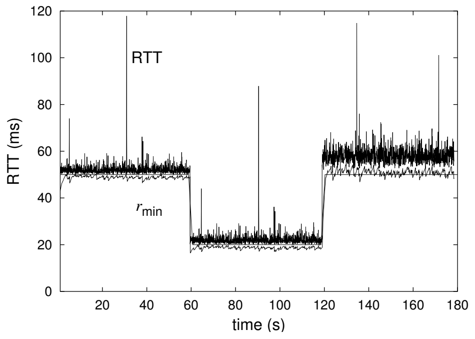

Fig. 11. Estimate of the minimum round-trip time. In the first minute, the delay distribution is the default. In the second minute, $\text{RTT}_{\min} = 20$ms. In the third minute, $r = 8$.

and the $r_{\min}$ in a simulation where the delay distribution is stable for one minute and it then changes abruptly and significantly. The average $r_{\min}$ was 0.7ms less than the RTT and underestimates it in all but 0.3% of the samples. The $r_{\min}$ estimate adapted quickly to changes in the RTT distribution.

Play-Back Delay: It was often the case that $T < \tau$, which means that a control pipe was established. Fig. 12 plots an estimate of the average number of packets in flight in the round-trip path between plant and controller as a function of $\text{RTT}_{\min}$. Since $\text{RTT}_{\min}$ is the only variable simulation parameter, T remains almost constant, and the control pipe becomes linearly deeper.

As for the value of the play-back delay τ, it should be a close approximation of the RTT, but should also overestimate it. Fig. 13 gives the difference $\tau -$ RTT under the default simulation parameters. The play-back delay overestimated the round-trip time in all but 3.3% of the packets. The mean overestimation $\tau -$ RTT was 4.8ms and its standard deviation 3.7ms. In other words, buffers led to the trade-off between a 96.7% confidence on the exact application time at the price of an additional average delay of 4.8ms. The underestimation was however severe in correspondence to occasional delay spikes because the algorithm does not attempt to predict spikes. The sensitivity to the jitter was investigated by keeping r constant and decreasing λ. As jitter increases, the algorithm generally increased the average $\tau -$ RTT but kept the underestimation rate constant.

Sampling Period: The impact of T will be clarified by setting T to a fixed value throughout simulations and showing the sensitivity of the results to the constant value of T. Fig. 14 shows m_2 as a function of T. In the first place, the root-mean-square error m_2 increases with T, as predicted in Sec. III-C. However, m_2 was significantly larger also at the smallest values of T (leftmost side of Fig. 14). Consequently, the fastest sampling rates should be avoided. Fig. 15 shows that m_2 is large when T is small relative to jitter. In the figure, T is normalized to the standard deviation of the gamma distribution (delay body). 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. When T is small (relative to jitter), m_2 is large. The reason is that, if T is small, τ is

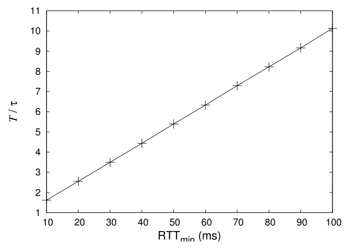

Fig. 12. The average ratio T/τ is an estimate of the number of packets in flight and is reported as a function of $\text{RTT}_{\min}$.

fundamentally defined by $T + r_{\min}$ (Algorithm 3, Line 18), 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 point is further illustrated in Fig. 16, which summarizes the imperfections of the control pipe as a function of the sampling period T. Late signals are almost 40% of all packets when $T = 2$ms, but they drop rapidly. The number of quashed signals is always less than 1% and decreases with larger sampling rates. 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 is derived from further increasing T. As for the dynamic setting of T (Algorithm 3), it slowly attempts to choose progressively faster sampling rates, but it backs off if it detects a potential for imperfections in the control pipe. The resulting late and quashed rates were close to the best that can be achieved with a constant value of T (Fig. 16).

Observer: The observer's prediction are subject to inaccuracies due to network non-determinism and to assumptions in the algorithm. Moreover, when the observer makes a prediction on the future plant state, it cannot anticipate τ seconds in advance the inherently unpredictable evolution of the stochastic processes v and w. The prediction inaccuracy was expressed as the mean-square difference between the predicted and true plant state, and it was basically equal to the variance of v and w over an interval of length τ. Hence,

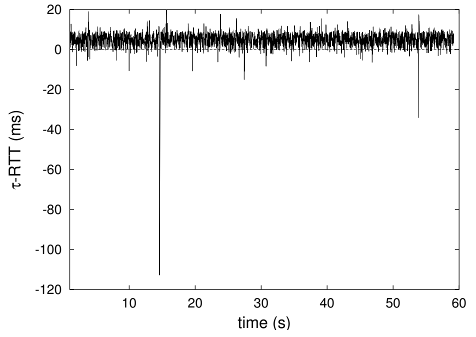

Fig. 13. Difference $\tau_i - \mathrm{RTT}_i$ (60 second snapshot, default parameter values).

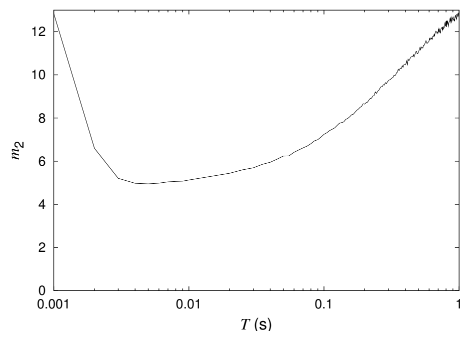

Fig. 14. Root-mean-square error m_2 of the output y as a function of a fixed sampling period T.

measured discrepancies can be attributed almost exclusively to inherently unpredictable plant disturbances.

Summary: The algorithm behavior cannot be attributed to any single component. Rather, the algorithm's modules worked appropriately individually as well as in combination with each other.

VI. Related Work

Remote control is a multi-disciplinary endeavor [8] and has been studied in control theory [14], robotics (the theme of the IROS 2005 conference is "Networked Sensors and Robots for the Improvement of the Quality of Life"), real-time systems, and middleware (e.g., [28]). Several control-theoretical algorithms can benefit from a more deterministic communication channel. For example, LQG controllers ([20], [14]) become more aggressive if the channel quality improves due, for example, to an underlying play-back buffer or to explicit QoS provisioning. In general, control-theoretical approaches are complementary to Networks methods. Network research has investigated the real-time capabilities afforded by shared Ethernet for manufacturing automation [10]. Networked Infomechanical Systems (NIMS) extend the capabilities of fixed networked sensors with mobile robots that explore an environment. In networked control, congestion control has been approached within an optimization framework [2]. Additional references (e.g., on play-back buffers) have been introduced in the previous sections of the paper.

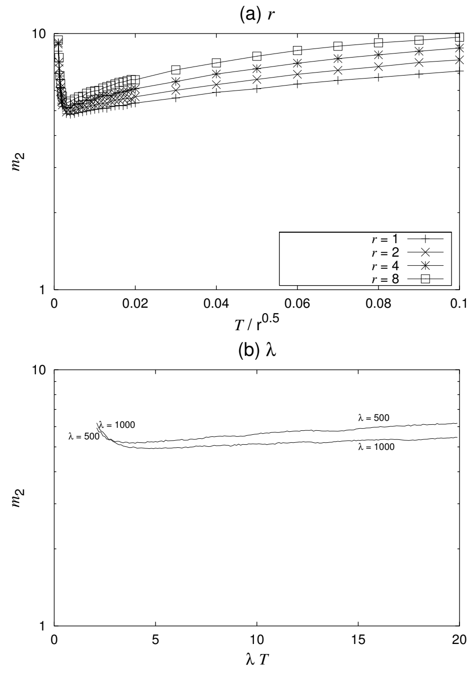

Fig. 15. Second moment m_2 of the output y as a function of the sampling period T for various jitter levels.

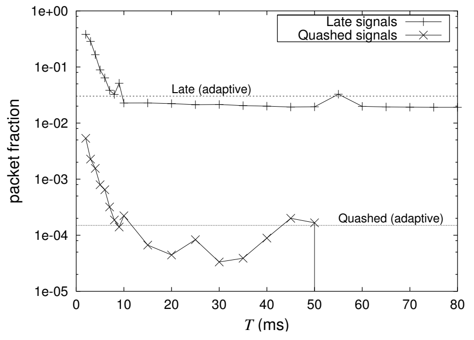

Fig. 16. Imperfections of the control pipe as a function of the fixed sampling period T.

VII. Conclusions

The main contribution of this paper is a play-back algorithm for tolerant and adaptive networked control. The play-back algorithm is integrated with sampling (through the dynamic setting of T) and control (through expiration times and performance metrics). Packet losses are dealt with by reverting to the inefficient but safer open-loop plant by means of the contingency control. Otherwise, the play-back delay τ adapted to network conditions and the dynamic sampling period T avoided most imperfections of the control pipe. The plant play-back is simple and its buffer is short. The algorithm was extensively simulated and emulated. It had low variability around the set point and it was robust to varying levels of connectivity.

Acknowledgments

We thank Michael S. Branicky, Ahmad Al-Hammouri, Huthaifa Al-Omari, and Stephen M. Phillips for many useful discussions. We thank Panos K. Chrysanthis, Brian D. Davison, and Stephen M. Phillips for providing some of the machines used in the wide-area experiments. Work supported in part under NSF CCR-0329910, Department of Commerce TOP 39-60-04003, and NASA NNC04AA12A.

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] Ahmad Al-Hammouri and Vincenzo Liberatore. Decentralized and dynamic bandwidth allocation in networked control systems. In *14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2006.

[3] Stephen P. Boyd and Craig H. Barratt. *Linear Controller Design. Limits of Performance*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

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

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

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

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

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

[9] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proc. of the IEEE Conference on Decision and Control*, 2002.

[10] Seok-Kyu Kweon, Kang G. Shin, and Gary Workman. Achieving real-time communication over Ethernet with adaptive traffic smoothing. In *Proceedings of the Sixth IEEE Real Time Technology and Application Symposium (RTAS)*, 2000.

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

[12] Vincenzo Liberatore. Panel on research challenges of end-to-end sense 'n respond solutions. In *Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services (EESR)*, 2005.

[13] Vincenzo Liberatore, M. Cenk Çavuşoğlu, Qingbo Cai, and Nikitha Kondapally. GiPSiNet: A middleware for haptic human-computer interaction. In *Telehealth*, 2005.

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

[15] B. Lincoln and B. Bernhardsson. Optimal control over networks with long random delays. In *Proc. International Symposium on Mathematical Theory of Networks and Systems*, 2000.

[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] 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.

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

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

[22] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the Internet. In *First Workshop on Hot Topics in Networking (HotNets-I)*, October 2002.

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

[24] Robert G. Pratt. The GridWise project. In *Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services (EESR)*, 2005.

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

[26] 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.

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

[28] D. Schmidt, A. Gokhale, T. Harrison, D. Levine, and C. Cleeland. TAO: A high-performance endsystem architecture for real-time CORBA. *IEEE Communications Magazine*, February 1997.

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

[30] Sangati Seth, Jerome P. Lynch, and Dawn Tilbury. Wirelessly networked distributed controllers for real-time control of civil structures. In *Proceedings of the American Controls Conference (ACC)*, 2005.

[31] A. Tzes, G. Nikolakopoulos, and I. Koutroulis. Development and experimental verification of a mobile client-centric networked controlled system. To appear.

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

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