Improving Performance of Networked Control Systems by using Adaptive Buffering

Luisa Repele, Riccardo Muradore, *Member, IEEE,* Davide Quaglia, *Member, IEEE,* and Paolo Fiorini, *Fellow, IEEE*

***Abstract*— Performance of Networked Control Systems is strongly affected by time-varying transmission delays. A traditional solution to this problem consists of storing arriving packets in a buffer which smooths delay jitter at the cost of an increased constant delay. The size of the buffer is based on either a long-term or worst-case analysis of network behavior leading to poor performance when the instantaneous network behavior is different. To overcome this problem, this work proposes 1) to adapt the buffer size according to the actual delay variation; 2) to re–size buffer content by using cubic spline smoothing which also reduces the signal noise; and 3) to use a Smith predictor at the controller side. Simulation results show that the adaptive buffering strategy reduces delay and packet loss probability while the spline smoothing process improves control performance even in case of constant-size buffers.**

***Index Terms*— Play-out buffer, adaptive buffer, networked control systems, spline smoothing.**

I. INTRODUCTION

NETWORKED Control Systems (NCS's) are spatially distributed systems for remote control applications often designed to operate in dangerous environment (e.g. nuclear plant maintenance) or to improve accuracy and safety in tele-operated plant (e.g. robotic surgery), [1]. Fig. 1 shows the basic block diagram of an NCS where the continuous-time plant $P(s)$ and the digital controller $C(z)$ are connected through a wired/wireless packet-based network. Both controller and plant are represented using a linear model whose transfer functions are given by $C(z)$ and $P(s)$ where z is the Z-transform variable and s is the Laplace variable, respectively. As usual, the controller sends commands u to the plant aiming at keeping measurement y as close as possible to reference r. In general, NCS's present many challenges for the design of stability–preserving controllers due to potential time-varying delays and packet dropouts, [2], [3].

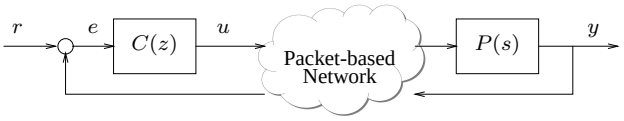


Fig. 1. Block diagram of a networked control system.

Several solutions appeared in literature during the last decade as reported in the survey paper [4]. The main strategies are based on linear–quadratic control (e.g. [5], [6]), on model predictive control (e.g. [7], [8], [9]), on gain scheduling (e.g. [10]), and on passivity theory (e.g. [11]). Other approaches use the co–design and quality–of–services/control paradigms (e.g. [12], [13], [14], [15]) as well as linear matrix inequalities (e.g. [16]). Due to the very different assumptions on network behavior, it is not easy to compare the different solutions in a fair way.

Even when the reliability of the network infrastructure avoids packet dropouts, time-varying transmission delays are always possible and they can degrade controller performance by compromising the assumptions on the loop delay. A possible way to address this issue consists of the introduction of a *playout buffer* which stores received packets and uses them after a fixed amount of time. This way, transmission delay variations and out-of-order packet arrivals can be compensated at the cost of an increased (but constant and known) delay. Such approach is quite common in multimedia (from the pioneer work in [17], to the improvements in [18], [19]), and recently has been introduced also within the control community ([20], [21], [22], [23]). Furthermore, the resulting constant delay can be effectively exploited by specific control techniques, e.g., by including a Smith predictor in the loop, [24].

The crucial point in using buffers is the choice of their size which determines the loop delay. Buffer re-sizing based on worst-case analysis of the network delay variation is rather conservative but leads to poor performance when the network behavior is better. Another possible approach uses long-term statistics: unfortunately sudden variations of the delay are not faced leading again to poor performance.

The main contribution of this work is to improve the buffer-based control architecture by

1) estimating the actual delay variation at run-time,
2) adapting the buffer size according to the estimated delay variation,
3) re–sizing the buffer content by using a smoothing technique based on cubic spline ([25]) which also reduces the signal noise,
4) adapting the delay in the Smith predictor according to the actual size of the buffers at the controller and plant side.

In this way, we aim at improving the performance by guaranteeing that the closed loop system works close to its optimal condition. The novelty of this approach is the run-time adaptation of the buffers according to the current network condition.

The paper is organized as follows. Section II introduces the concept of buffering and the challenges to be solved to use it

L. Repele, R. Muradore, D. Quaglia and P. Fiorini are with the Department of Computer Science, University of Verona, Italy.

in NCSs. Section III presents the proposed solution based on an adaptive buffer/controller architecture. Spline smoothing is the basic element of the proposed approach and is described in Section IV. Simulation results are reported in Section V, and conclusions are drawn in Section VI.

II. PROBLEM DESCRIPTION

The classic NCS with buffers at the controller and plant side can be represented as in Fig. 2. Let T_s be the sample time of the system. Each command and measurement is sent on the network in a different data packet at $t = kT_s$, $k \in \mathbb{N}$. Let u_s and u_r be the commands sent/received to/from the network, respectively, and let y_s and y_r be the measurements sent/received to/from the network, respectively. In the following we focus on the delay introduced in the packet transmission; however the algorithms proposed in Sections III and IV will cope also with packet losses. Let δ_{C2P} be the delay in the forward path and δ_{P2C} in the backward path, respectively. The following equations hold:

$$u_r(t) = u_s(t - \delta_{C2P}(t))$$
$$y_r(t) = y_s(t - \delta_{P2C}(t))$$

The network delay is due to different contributions:

- *access delay* due to transmission channel arbitration and possible re-transmissions;
- *packet coding delay* depending on the length of the packet and the physical speed of the link;
- *propagation delay* due to the signal propagation over the physical medium;
- *processing delay* by intermediate systems, e.g., switch enqueueing; this contribution affects multi-hop transmissions.

The first and last contributions depend on network conditions and therefore they can change over time. In case of multi-path networks, delay variation may lead to out-of-order packet delivery, i.e., $u_s(k'T_s)$ may arrive before $u_s(kT_s)$ even though $k' > k$.

In Fig. 2, the blocks named *PBuffer* and *CBuffer* are first-in-first-out queues, which host a given number of packets. As in multimedia, to reduce the probability of underflow/overflow, buffers are initially filled up to half their size (pre-buffering) and then packets are extracted at $1/T_s$ rate. If the buffer is large enough, it compensates delay variations and sorts out-of-order packets according to the original sequence. Therefore, the PBuffer/CBuffer avoids packet dropouts due to late and out-of-sequence arrivals and allows applying the commands/measurements with the original sampling rate thus simplifying the design of the discrete-time controller $C(z)$. The presence of the buffer makes the total delay constant but larger than the actual network delay because of the presence of the pre-buffering delay. Let u_B and y_B the commands and measurements extracted from the PBuffer (of size S_P) and CBuffer (of size S_C), respectively. The following equations hold:

$$u_B(t) = u_s(t - \tau_P) \quad (1)$$
$$y_B(t) = y_s(t - \tau_C) \quad (2)$$

where τ_P and τ_C are the constant forward and backward path delays. They are given by:

$$\tau_P = \mu_{C2P} + \frac{S_P}{2}T_s \quad (3)$$
$$\tau_C = \mu_{P2C} + \frac{S_C}{2}T_s \quad (4)$$

where μ_{C2P} and μ_{P2C} are the average values of the network delay from controller-to-plant and from plant-to-controller, respectively, while $\frac{S_P}{2}T_s$ and $\frac{S_C}{2}T_s$ are the pre-buffering delays of PBuffer and CBuffer, respectively. Therefore, the resulting overall loop delay (i.e., round-trip delay) is constant and equal to

$$h = \tau_P + \tau_C \quad (5)$$

To reduce packet dropouts due to buffer overflow and underflow, the value of S_P and S_C should be proportional to the variation of network delay

$$S_P = \left\lceil \frac{B\sigma_{C2P}}{T_s} \right\rceil, \qquad S_C = \left\lceil \frac{B\sigma_{P2C}}{T_s} \right\rceil. \quad (6)$$

where σ_{C2P} and σ_{P2C} are the standard deviations for the controller-to-plant and plant-to-controller paths, respectively, and B is a number depending on the delay statistics [17].

Since buffers lead to a constant loop delay, a Smith predictor with parameter h given by Equation (5) can be used at controller side [24] as shown in Fig. 2. It is well known that the Smith predictor allows improving the response to command signals when the plant $P(s)$ is asymptotically stable. The inner controller $C(z)$ can be designed as if the transmission delay were absent. For this reason, the same tuning of $C(z)$ can be used for different values of loop delay provided that it is *constant* and *known* at controller side so that h can be modified accordingly. Even though the delay is still in the loop, the output y follows quite well the reference r with a fixed time-shift which depends on h.

Network condition may vary over time and the long-term value of delay variation is usually different from the short-term value. If buffer sizes are decided based on long-term delay variation, then they can lead to packet losses in the worst-case periods and to unnecessary high delays in the other cases. If buffer size is derived from the highest delay variation, then no packet loss occurs but unnecessary delays are applied most of the time. Both packet losses and high delay (even if constant) compromise the tracking performance between the output of the plant and the reference signal. To overcome this problem we propose to *adapt the size of the buffers* according to the short-term value of delay variation. When the variance decreases, the buffer will be shrunk to reduce the delay; when the variance increases, the buffer will be increased to keep the dropout probability low. This way, the introduced delay is piecewise constant and its value is minimized; furthermore, the possibility to use the Smith predictor in parallel to the controller is preserved.

To create this novel solution the following issues have to be addressed:

- *estimation of delay variation*: mean and standard deviation of network delay need to be estimated at run-time

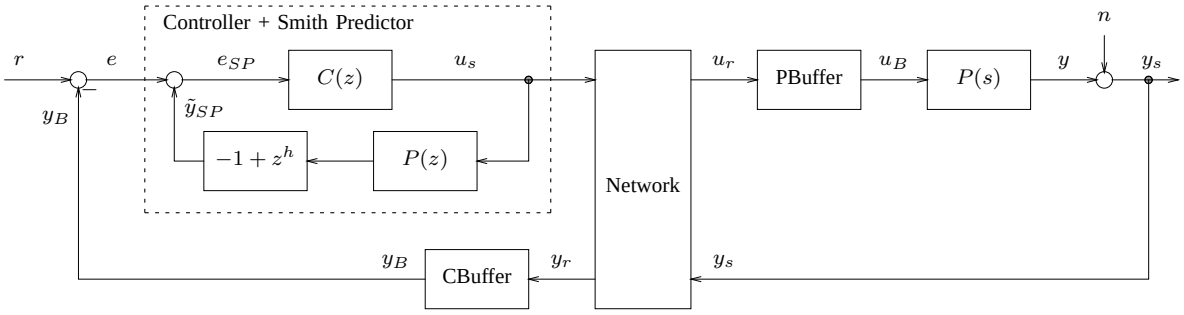


Fig. 2. Standard buffer architecture for a networked control system.

by inspecting arriving packets or relying on a network management component;

- *buffer re-sizing*: while in adaptive Voice-over-IP applications playout time is shifted in the silence intervals between talk-spurts [17], in control applications buffer content has to be re-sized to continuously feed plant and controller with commands and measurements, respectively;
- *communication about buffer re-sizing*: Smith predictor-based controller is quite sensitive to the accuracy of the loop delay h; therefore its new value, due to buffer re-sizing, has to be communicated to the controller as soon as possible.

All these issues are addressed in the next Section, which presents a methodology and an architecture to jointly adapt buffers and controller according to network condition.

The model in Fig. 2 also considers noise n which affects plant measurements and thus also control performance. In Section IV we will show that the algorithm for buffer content re-sizing can also be used to reduce this noise.

III. ADAPTIVE BUFFER/CONTROLLER APPROACH

As presented in Fig. 3, the resulting architecture contains new blocks with respect to the traditional one (Fig. 2) to perform the following operations:

- computation of network statistics
- adaptation of buffer size
- re-sizing and smoothing of buffer content.

The first two items will be described in this Section while the third will be addressed in Section IV.

A. Computation of Network Statistics

In the proposed architecture, the size of PBuffer and CBuffer is not constant but depends on the short-period value of the *standard deviation of network delay* according to Equation 6. Furthermore, the Smith predictor used at controller side needs to know the *loop delay* h. There are different ways to estimate these statistics:

- They can be provided by the network management component; this feature is network-dependent and it is not always available in all network implementations.
- They can be estimated as suggested by the RTP/RTCP standard [26]; standard deviation on both sides of the network can be estimated from the inter-arrival jitter of the packets while loop delay can be estimated from the round-trip time of RTCP packets.
- Network delay for each packet can be computed by storing the transmission timestamp in the packet header and subtracting it from the reception time; loop delay is computed by summing up the delay in both directions while standard deviation is estimated from the delay. This approach assumes that controller and plant are synchronized; this assumption can be satisfied by using either a synchronization protocol found in literature (e.g., [27]) or a common synchronization source such as GPS [28].

In this work we follow the last approach since it is simple to be presented and implemented in the experiments. It is worth noting that the proposed adaptive buffering methodology is independent of the way delay variations and loop delay are estimated. Furthermore, such estimation protocols use a constant amount of network capacity and thus they do not interfere with the adaptive buffering approach which is sensible to variations of network condition.

The average delay is estimated as follows:

$$\delta_{C2P}(kT_s) = t - t^k_{u_r} \quad (7)$$

$$\delta_{P2C}(kT_s) = t - t^k_{y_r} \quad (8)$$

$$\hat{\mu}_{C2P}(kT_s) = \frac{1}{W}\sum_{\ell=0}^{W-1}\delta_{C2P}(kT_s - \ell T_s) \quad (9)$$

$$\hat{\mu}_{P2C}(kT_s) = \frac{1}{W}\sum_{\ell=0}^{W-1}\delta_{P2C}(kT_s - \ell T_s) \quad (10)$$

where $t^k_{u_r}$ and $t^k_{y_r}$ are the k-th transmission timestamps of the received commands and measurements, respectively, while W is the length of a sliding window. The window length W (WT_s in sec) has to be large enough to have "statistically reasonable" estimates of mean and variance. However W should not be too large otherwise the system would not adapt promptly. From the tuning viewpoint, this parameter is selected by looking at past time series of the communication delay. In literature, there is a wide discussion on the estimation of network statistics: another possible solution is the implementation of a low-pass filter [26], [17].

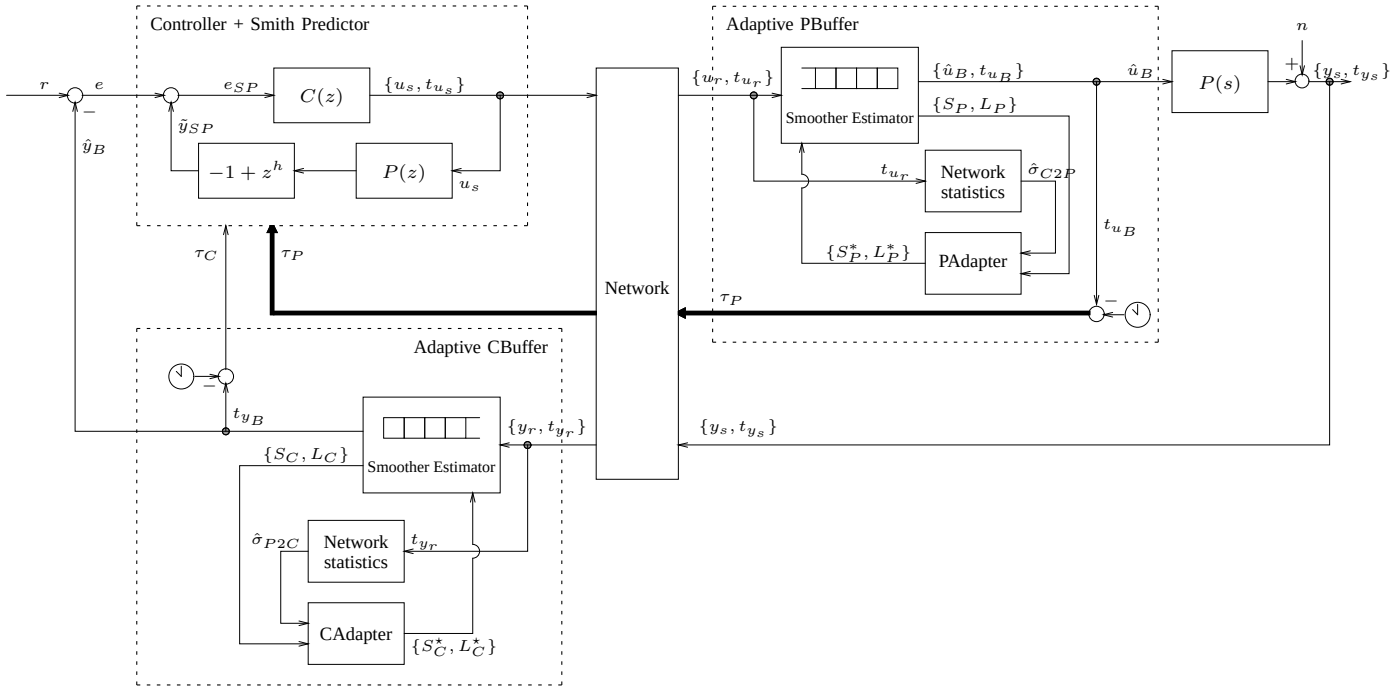


Fig. 3. Adaptive Buffer/Controller architecture. The bold line means a high-priority channel.

The standard deviation is estimated as usual:

$$\hat{\sigma}_{C2P}(kT_s) = \sqrt{\frac{1}{W-1}\sum_{\ell=0}^{W-1}\left(\delta_{C2P}(kT_s - \ell T_s) - \hat{\mu}_{C2P}(kT_s)\right)^2}$$

$$\hat{\sigma}_{P2C}(kT_s) = \sqrt{\frac{1}{W-1}\sum_{\ell=0}^{W-1}\left(\delta_{P2C}(kT_s - \ell T_s) - \hat{\mu}_{P2C}(kT_s)\right)^2}$$

Two blocks called *Network Statistics* are introduced at controller and plant sides to compute $\hat{\mu}_{C2P}$, $\hat{\sigma}_{C2P}$ and $\hat{\mu}_{P2C}$, $\hat{\sigma}_{P2C}$, respectively.

The end-to-end delay is made constant by the presence of the buffer and its variation depends only on the variation of buffer size according to Equations (3)-(4). Therefore the delay has to be computed after buffer re-sizing to update the loop delay. The end-to-end delay in both directions is computed as the difference between the current time and the timestamp of the samples exiting the PBuffer/CBuffer as follows:

$$\tau_P(kT_s) = t - t_{u_B}^k \quad (11)$$

$$\tau_C(kT_s) = t - t_{y_B}^k \quad (12)$$

Such delay values are summed up at the controller side to obtain the loop delay needed by the h parameter of the Smith predictor. It is worth remarking that the τ_P delay information has to travel through the network thus being affected by delay and data packet losses. In this case, the h parameter of the controller is not aligned promptly with the actual loop delay thus leading to a transitory loss of performance. For simplicity's sake in this work we assume that a high-priority channel is used to send τ_P thus reducing the possibility of delayed arrival and packet losses (see bold line in Fig. 3).

Average and standard deviation are not updated if packets do not arrive at destination or the delay is very high. To take this case into account, the adaptation of buffer size does not depend only on such statistics but also on the current amount of data within the buffer as described in the next Section.

B. Adaptation of Buffer Size

In this section we analyze the dynamic change of the size of PBuffer and CBuffer. According to Equation 6, the choice of buffer size is driven mainly by the standard deviation of the delay of the incoming packets. As explained in the previous Section, this value is estimated at each packet arrival over a window of W packets. Buffer re-sizing requires some operations on the buffer content (see Section IV) and therefore we decided to trigger it when the relative difference between the desired value (i.e., according to standard deviation) of buffer size and the current value is higher than a given threshold Th_S. When the buffer size is changed, also its content has to be modified to guarantee that the buffer occupancy degree is preserved. In particular, it is worth recalling here that the buffer works in its optimal condition when it is half filled because the probability of buffer under/overflow is minimized and, at the same time, the delay is kept constant. If the new size is larger than the actual size, "virtual" data should be put in the buffer. On the other hand, if it is smaller then in the buffer there are more data than needed and a strategy to throw away some of them is required. The solution to this problem is described in Section IV.

In case of a long sequence of packet loss events or if the delay is increasing, statistics computation is delayed and the delay in buffer adaptation could lead to potential buffer underflow. To address this case, buffer size adaptation is also

triggered if the relative difference between the current buffer level L_P (L_C) and the optimal one (i.e., half of the current buffer size) is higher than a given threshold Th_L. In this case, only the buffer content is re-sized to bring the buffer level to half of buffer size while the buffer size is not changed.

The thresholds Th_S and Th_L are application- and network-dependent: the choice of these parameters is part of the tuning of the control architecture (such as the tuning of the PID controller $C(z)$).

As a result new blocks are introduced in the architecture both at controller and plant side (namely, *PAdapter* and *CAdapter*) to implement the following adaptation algorithm.

procedure ADAPTBUFFER($\hat{\sigma}$, S, L, Th_S, Th_L)
  ▷ $\hat{\sigma}$ is the new estimation of standard deviation
  ▷ S and S^{opt} are the current and the optimal buffer sizes
  ▷ $S^\star$ is the new buffer size
  ▷ L and L^{opt} are the current and the optimal buffer levels
  ▷ $L^\star$ is the new buffer level
  ▷ Th_S is the threshold on size changes
  ▷ Th_L is the threshold on buffer level changes

  $S^{opt} = \left\lceil \frac{6\hat{\sigma}}{T_s} \right\rceil$
  $L^{opt} = \frac{S}{2}$
  if $|\frac{S^{opt}-S}{S}| > Th_S$ **then**
    $L^\star = L\frac{S^{opt}}{S}$
    $S^\star = S^{opt}$
    resizeBuffer($L^\star$,$S^\star$) ▷ Change buffer size/level
  else if $|\frac{L^{opt}-L}{L}| > Th_L$ **then**
    $L^\star = L^{opt}$
    $S^\star = S$
    resizeBuffer($L^\star$,$S^\star$) ▷ Change buffer level
  end if
end procedure

IV. CONTENT RE-SIZING AND SMOOTHING

The problem of buffer content re-sizing has been formulated as the problem of obtaining an analytical curve from the samples currently in the buffer and then re-sampling it with a different sampling rate. This approach has the following advantages:

- missing samples (due to packet loss or late arrival) can be estimated by interpolation;
- noise (e.g., measurement noise) can be eliminated by introducing a smoothing effect in the generation of the analytical curve.

Among the different solutions available in literature to obtain an analytical curve from a sequence of samples, we chose the spline approach [25].

A. Spline Smoothing

Let $B_{j,k,\mathbf{t}}$ be the j-th B-spline of order k for the knot sequence $\mathbf{t}$. According to [25], the spline space $\$_{k,\mathbf{t}}$ is the space of any linear combinations of $B_{j,k,\mathbf{t}}$ splines

$$\$_{k,\mathbf{t}} = \left\{ f \,\middle|\, f = \sum_j a_j B_{j,k,\mathbf{t}}, a_i \in \mathbb{R}, \forall i \right\}.$$

Let $Y_{[1,N]}$ be a set of N values $\begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}$ obtained from

$$y(t_i) = g(t_i) + e(t_i)$$

where g is an unknown smooth function, t_i is the time, and e is additive noise. Let $\mathbf{t}$ be the knot sequence related to the sample times $t_1, \ldots, t_i, \ldots, t_N$. The problem of estimating the function g within the space $\$_{k,\mathbf{t}}$ can be formalized as finding

$$f^\star = \arg \min_{f \in \$_{k,\mathbf{t}}} W(f) \tag{13}$$

where

$$W(f) = \alpha \underbrace{\sum_{i=1}^{N} w_i (y_i - f(t_i))}_{W_1} + (1-\alpha) \underbrace{\int_{t_1}^{t_N} \frac{d^m f(t)}{dt^m}}_{W_2}. \tag{14}$$

The performance index $W(f)$ is due to Schoenber, Reinsch and Whittaker, [25] and it takes into account the "interpolation" part W_1 (the estimated curve should stay close to the sampled data) and the "regularization" part W_2 (a smooth curve is desired to filter out the noise). The choice of the coefficient α is crucial for the trade-off between the two conflicting terms. The coefficients w_i are used to weight each sample y_i according to its importance.

In this work we use cubic splines, i.e. $k = 3$, and the derivative order in Equation (14) is $m = 2$.

B. Data Smoothing and Estimation of Missing Packets

In the present context, the sampled data y_i are the sequence of commands/measurements currently in the PBuffer/CBuffer. We will focus on the processing of the data in the PBuffer; the same line of reasoning holds for the CBuffer.

We start by assuming that all commands $u_s(t)$ uniformly sampled at $t = kT_s$ are received, i.e.

$$u_r(kT_s) = u_s(kT_s - \delta_{C2P}(kT_s)).$$

Thanks to the buffering, it is possible to re-order the out-of-sequence commands and to retrieve the original sampling order. Let L_P be the current buffer level. This means that the buffer adds to any commands a fixed delay equal to $T_s L_P$. At time t, $kT_s \leq t < (k+1)T_s$ the following elements are within the PBuffer

$$u_B(i) = \begin{cases} u_s(kT_s + (L_P - i)T_s), & \text{for } i = 1, \ldots, L_P \\ \text{NaN}, & \text{otherwise.} \end{cases}$$

The data in the buffer can be seen as the commands that will be applied in the future. To smooth or estimate the "actual" command (in case it is delayed or lost) it is important to keep a window of past commands to be processed together with the future ones.

Let $\hat{U}_B$ and U_B be the p past commands and the commands in the buffer, respectively:

$$\begin{aligned}\hat{U}_B &= \begin{bmatrix}\hat{u}_B(k-p) & \hat{u}_B(k-p+1) & \cdots & \hat{u}_B(k-1)\end{bmatrix}\\ U_B &= \begin{bmatrix}u_B(k) & u_B(k+1) & \cdots & u_B(k+L_P)\end{bmatrix}\end{aligned}$$

with the corresponding sampling sequences

$$\begin{aligned}\hat{\mathbf{t}}_B &= \begin{bmatrix}(k-p)T_s & (k-p+1)T_s & \cdots & (k-1)T_s\end{bmatrix}\\ \mathbf{t}_B &= \begin{bmatrix}kT_s & (k+1)T_s & \cdots & (k+L_P)T_s\end{bmatrix}.\end{aligned}$$

Since the sampling is uniform there is no need to keep the k index: what matters is the inter-distance between consecutive points, i.e.

$$\hat{\mathbf{t}}_B = \begin{bmatrix}-pT_s & (-p+1)T_s & \cdots & -T_s\end{bmatrix} \quad (15)$$
$$\mathbf{t}_B = \begin{bmatrix}0 & T_s & \cdots & L_PT_s\end{bmatrix}. \quad (16)$$

To estimate the value $\hat{u}_B(kT_s)$ (the empty square in Fig. 4), it is necessary to go through the following steps:

1) solve the problem (13)-(14) with
$$\begin{aligned}y_i &\in [\hat{U}_B \quad U_B]\\ \mathbf{t} &= [\hat{\mathbf{t}}_B \quad \mathbf{t}_B],\end{aligned}$$
2) let $g(t) = \sum_j a_j B_{j,\mathbf{t}}$ be the solution of the previous step, then
$$\hat{u}_B(kT_s) = g(0).$$

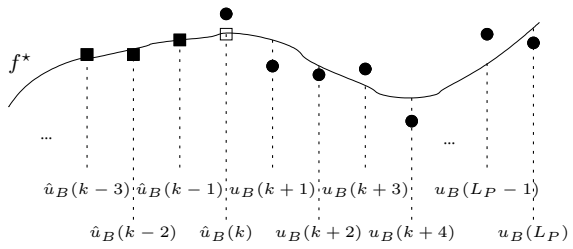


Fig. 4. Example of sample estimation and smoothing (it is worth noting that $\hat{u}_B(k \pm i)$ is a short writing for $\hat{u}_B(kT_s \pm iT_s)$). ■ past data; • data within the buffer; □ estimated/smoothed sample.

This procedure can be used even though some points in U_B are missing and so also when the missing point is the command needed at current time, i.e. $u_B(kT_s)$. As we will show in Section V, this algorithm allows to:

- remove noise from the data contained in the buffer (which is especially important at the controller side since the received measurements are affected by noise);
- estimate missing commands and measurements from past and future values,

C. Re-sizing the buffers

When the buffer size has to be changed, also the buffer content need to be adapted to preserve buffer level; this Section explains the data decimation when the buffer dimension is decreased and the introduction of virtual data when the buffer dimension is increased. The procedure is still based on the cubic spline smoothing and goes through the following steps:

1) let $g(t) = \sum_j a_j B_{j,\mathbf{t}^{old}}(t)$ be the curve which is the solution of the problem (13)-(14) with
$$\begin{aligned}y_i &\in [\hat{U}_B^{old} \quad U_B^{old}]\\ \mathbf{t}^{old} &= [\hat{\mathbf{t}}_B^{old} \quad \mathbf{t}_B^{old}],\end{aligned}$$
2) let S_P^{new} and S_P^{old} be the new size of the buffer and the current one, respectively; then "stretch" the time according to the ratio $\kappa := \lceil S_P^{new}/S_P^{old} \rceil$ as
$$f^\star(t') = \sum_j a_j B_{j,\mathbf{t}^{old}}(t')$$
with $t' = \kappa t$,
3) down-sampling the curve if $S_P^{new} < S_P^{old}$ or up-sampling the curve if $S_P^{new} > S_P^{old}$
$$\begin{aligned}\mathbf{t}_B^{new} &= \begin{bmatrix}0 & T_s & \cdots & S_P^{new}T_s\end{bmatrix}.\\ \hat{U}_B &= \begin{bmatrix}\hat{u}_B(k) & \hat{u}_B(k+1) & \cdots & \hat{u}_B(k+S_P^{new})\end{bmatrix}\end{aligned}$$
with
$$\hat{U}_B = f^\star\left(\kappa \mathbf{t}_B^{new}\right).$$

Fig. 5 shows how this procedure works in the two cases.

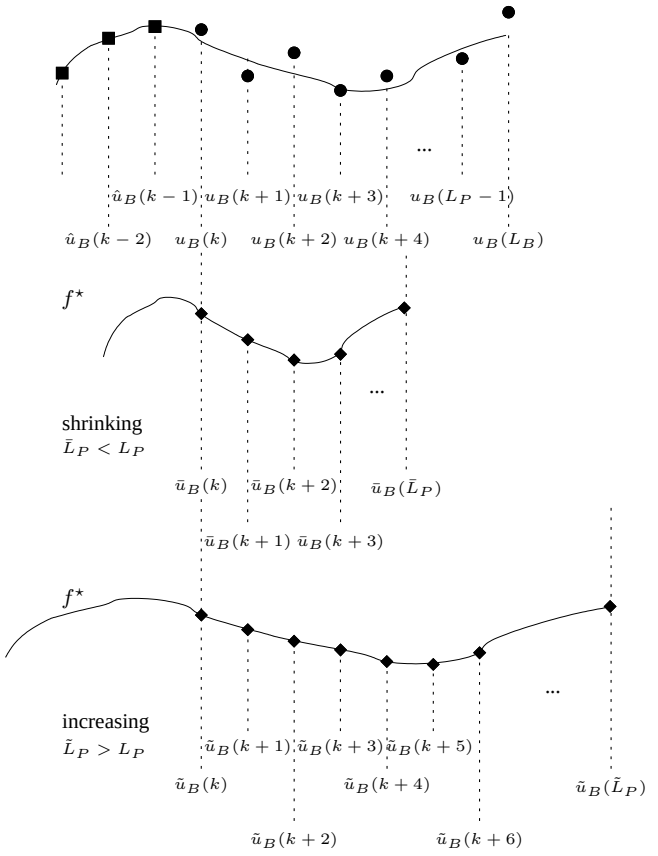


Fig. 5. Buffer re-sizing. Top: smoothing of the original data; Middle: shrinking of buffer content; Bottom: increasing of buffer content. Symbols: ■ past data; • data within the buffer; ♦ estimated data.

V. SIMULATION RESULTS

The proposed adaptive buffer/controller strategy has been validated on a Matlab/Simulink scenario with a first order plant

$$P(s) = \frac{1}{1 + 0.02s}$$

and a PI controller

$$C(z) = K_P e(k) + \frac{K_i T_s}{1 - z^{-1}} e(k).$$

The sample time T_s for commands and measurements has been set to 1 ms, whereas the noise variance is 0.01. At each sample time, a packet is generated for each command/measurement. The window W is set to 100 (i.e. 0.1 s) and the thresholds Th_L, Th_S in the ADAPTBUFFER( ) procedure are both equal to 0.2.

The following strategies have been compared:

- *Un-buffered*, i.e., there is no buffer at both sides; the plant applies the most recently received command only if its sequence number is greater than the previous one and considers the other packets as lost; in this case the previously applied data is held. The similar strategy is adopted at the controller side.
- *Constant-size buffer (CB)*, i.e., a simple queue to store data; out-of-sequence packets can be sorted inside the buffer provided that they arrive before their playout time otherwise the last applied data is held; buffers containing 20, 30, 45, and 90 packets have been considered.
- *Smoothing Buffer (SB)*, i.e., a constant-size buffer in which data smoothing and estimation of missing packets is performed; buffers containing 20, 30, 45, and 90 packets have been considered.
- *Adaptive Smoothing Buffer (ASB)* performing data smoothing, estimation of missing packets, adaptation of buffer size and buffer level; referring to Equation (6), $B = 3$ has been used.

The controller parameters K_P, K_I have been chosen so that the overshoot of the step response is less than or equal to 10%. Their values are the same for all the cases in which a buffer is present, either fixed or adaptive, since the Smith predictor makes the choice of K_P, K_I independent of the delay, provided that it is known (thanks to the presence of the buffer). The Smith predictor uses the discrete-time approximation of $P(s)$ at sample time T_s. In the un-buffered case (without the Smith predictor), the controller parameters have been chosen by considering an average delay over the whole simulation.

A. Real Wireless Scenario

The proposed approach has been applied on a real network scenario featuring a constant-bitrate transmission between a mobile node and a stationary node, [29]. The blue dots in Fig. 6 represents the network delay. The dashed red line is the almost piecewise-constant delay obtained by using our adaptive buffer technique. It is worth highlighting that the proposed approach is able to follow also sharp spikes.

In the following tests, the network has been simulated by using a data structure which sorts packets according to their associated delay. The sequence of delay values has been generated through a stochastic process based on a Gaussian distribution; values less than a minimum delay τ_{min} are replaced by τ_{min}. Since we are interested in time-varying network conditions, the sequence of delay values has been merged from three sub-sequences obtained by using different sets of generation parameters as reported in Table I. Blue dots in Figs. 8 and 10 show two examples of network behavior used in the experiments.

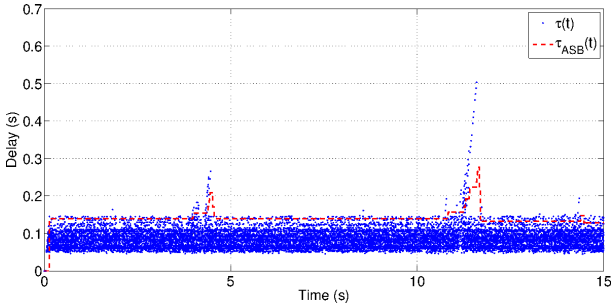


Fig. 6. Packet delay in a real wireless scenario.

TABLE I
PARAMETERS FOR THE GENERATION OF DELAY VALUES.

Interval	μ (ms)	σ (ms)	τ_{min} (ms)
0-5 s	15	5	5
5-10 s	35	15	5
10-15 s	20	10	5

The motivation behind this choice is twofold:

- in the short period there are many independent sources of delay that can be approximated by a Gaussian variable with constant mean and variance (Central Limit Theorem),
- in the long period the causes of big changes in mean and variance are phenomena with span over a significant number of packet transmission intervals (e.g., a competing traffic is switched on).

For sake of simplicity, the statistics on the two paths (plant to controller, and controller to plant) are assumed equal.

B. Regulation Test Case

In this test case the controller has to keep the output of the plant as close as possible to zero in spite of measurement noise and network problems. Table V-B reports the performance of the different buffering strategies in this case. The second and third columns report mean and standard deviation of the tracking error over ten simulations with different initialization of the random number generator (used to obtain packet delay values). Tracking error is computed as the standard deviation of the difference between the reference signal r and the measurement y. Two different tracking error metrics have been considered, i.e., at plant side $(r - y_s)$ and at controller side $(r - \hat{y}_B)$. As expected the latter is higher due to measurement noise and network problems. The last three columns show the packet loss rate during three different simulation intervals characterized by different network condition.

TABLE II
PERFORMANCE OF DIFFERENT BUFFERING STRATEGIES FOR THE REGULATION TEST CASE.

Buffering strategy	Tracking error (mean/st.dev.) $r - y_s$	$r - \dot{y}_B$	Packet Loss Rate (%) [0-5]s	[5-10]s	[10-15]s
Un-buffered	0.0371/0.0018	0.0804/0.0024	73.27	43.43	26.63
CB $S = 20$	0.0281/0.0054	0.0613/0.0234	1.44	77.11	29.00
CB $S = 30$	0.0269/0.0019	0.0769/0.0016	0.08	64.51	14.18
CB $S = 45$	0.0204/0.0008	0.0740/0.0009	0.00	43.39	3.88
CB $S = 90$	0.0156/0.0005	0.0725/0.0007	0.00	4.14	0.06
SB $S = 20$	0.0221/0.0013	0.0341/0.0020	1.44	77.11	29.00
SB $S = 30$	0.0170/0.0011	0.0258/0.0015	0.08	64.42	14.25
SB $S = 45$	0.0129/0.0008	0.0194/0.0008	0.00	43.39	3.88
SB $S = 90$	0.0142/0.0008	0.0201/0.0010	0.00	4.14	0.06
ASB	0.0133/0.0010	0.0192/0.0012	0.08	1.65	0.09

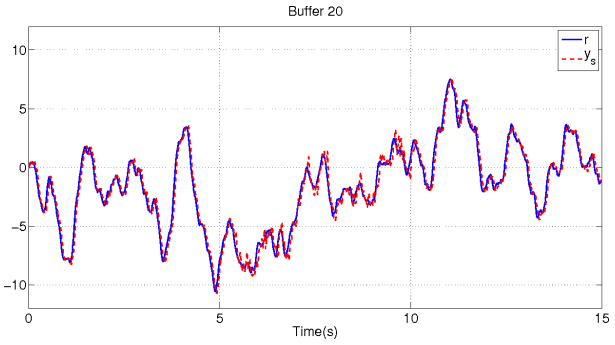
Results show that the ASB strategy provides performance close to the best constant-size buffer without the need of *a priori* knowledge of the delay statistics. Even if it does not provide the best performance in each interval, it guarantees a smooth behavior between different network condition. It is worth remarking that a first significant improvement is given by the proposed data smoothing and estimation function even if buffer size is kept fixed (i.e., with the SB strategy). In fact, this pre-processing decreases the tracking error (in particular at the controller side) by decreasing the measurement noise and estimating lost data packets.

C. Tracking

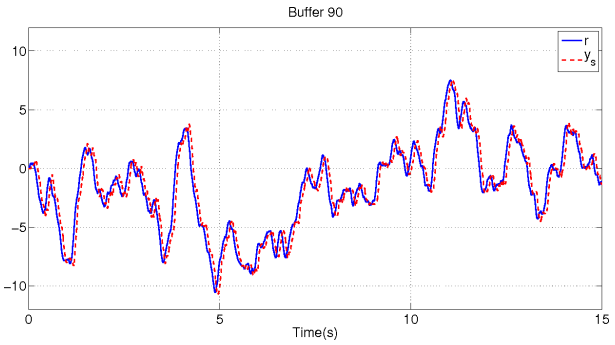
The second test case evaluates the tracking of a reference signal r. Fig. 7 shows the tracking performance when the SB strategy is used. The behavior of the network is still consistent with the parameters in Table I. In the case with $S_P = S_C = 20$ (Fig. 7.a), the tracking error increases between 5 to 10 seconds, when the standard deviation of the delay is high leading to several packet dropouts. Vice versa, in the case with $S_P = S_C = 90$ (Fig. 7.b), the tracking performance is good for the whole simulation but with a higher delay. Fig. 8 shows the network delay (blue dots) and the controller-to-plant delay when $S_P = 20$ (dashed red line) and $S_P = 90$ (dashed black line): the blue dots above the two lines correspond to packets dropouts. As expected, a larger buffer reduces the packet loss rate but increases the time lag between reference r and output y.

Fig. 9 shows the tracking performance with the ASB strategy. With respect to Fig. 7, the output y follows accurately the reference r also in presence of high variations of the network delay and the displacement between the two curves is minimized. Fig. 10 shows the delay introduced by the ASB buffer (red dashed line) with respect to the network delay (blue dots). We can see that the former changes over time according to the network condition. In this way the number of blue dots over the line (which correspond to packet dropouts) is minimized. The buffer delay behavior is piecewise constant to minimize computation effort and artifacts due to buffer content re-sizing.

Table III compares the behavior of the different buffering strategies for the Tracking Test Case. The second column reports the controller-to-plant delay which depends on buffer



(a) SB $S_P = S_C = 20$



(b) SB $S_P = S_C = 90$

Fig. 7. Tracking performance for the SB strategy with different buffer size values: reference r (blue solid line), output y (red dashed line).

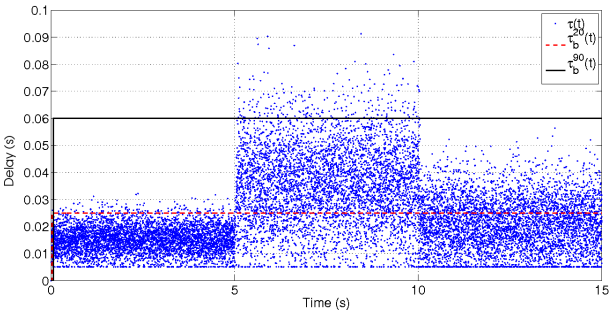


Fig. 8. Network Delays (blue dots), controller-to-plant delay with constant-size buffers of 20 (red dashed line) and 90 (black dashed line) packets.

size; in case of ASB strategy an average delay has been computed for each period of the simulation characterized by different network condition. The third column shows mean and standard deviation of the tracking error over ten experiments. Tracking performance is reported for each period of the simulation. To emphasize the tracking artifacts over the simple effect of buffer delay (which is known in advance when the buffer size is decided), the tracking error has been computed by shifting the reference signal with the delay reported in the second column.

The SB strategy outperforms the Pure Buffer strategy thanks to the smoothing/estimation function that reduces noise and the effect of packet loss. The ASB strategy provides results which

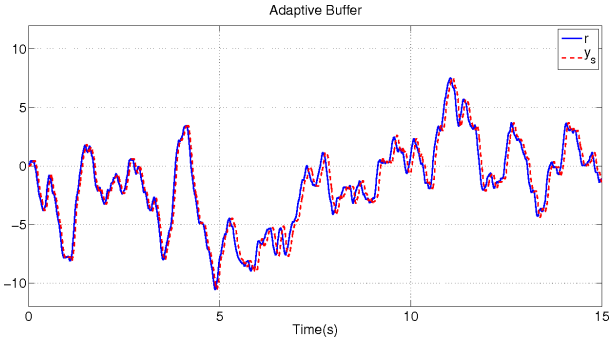


Fig. 9. Tracking performance with the ASB strategy: reference (blue solid line), output y (red dashed line).

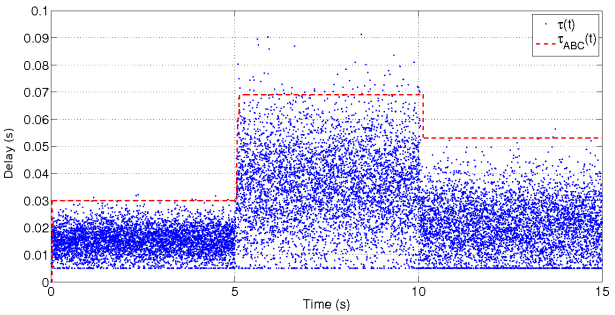


Fig. 10. Network Delays (blue dots), controller-to-plant delay with the ASB strategy (red dashed line).

are close to the best results of the SB strategy in the different periods. Even if the ASB strategy may not provide the best results, it has the advantage of adapting to network condition without any *a priori* knowledge. It is worth remarking that such slightly worse result (compared to SB $S = 90$) within the [5-10] range (higher variability interval) is due to the transient phase during the adaptation. This means that in longer periods this effect is averaged out.

D. Noise Reduction and Tracking Error

Table IV reports the effect of different amount of noise on tracking error in presence of the smoothing feature of the SB strategy (also within the ASB strategy). The tracking error is reported as the increase percentage with respect to the noiseless case. The first two rows show, as expected, an error increase corresponding to the increase of the noise variance. For the same values of noise variance, the smoothing feature of the buffering strategy reduces significantly the effect of noise on the tracking error.

VI. CONCLUSIONS

An improved buffer management strategy has been presented for networked control systems with time-varying communication delay. At run time the buffer size is changed to match the variation level of the estimated network delay. Also the buffer level can be changed when it may lead to underflow or overflow. In these cases buffer content must be re-sized to avoid gaps in the control/measurement flow; a methodology based on spline smoothing has been proposed which also reduces measurement noise and estimates lost packets. Simulation results show that the adaptive buffering strategy reduces delay and packet loss probability while the spline smoothing process improves control performance even in case of constant-size buffers. Future work will present the proof of system stability which takes into account the presence of piecewise constant delay and shrinking/increasing of the buffer content.

TABLE III

PERFORMANCE OF DIFFERENT BUFFERING STRATEGIES FOR THE TRACKING TEST CASE.

Buffering Strategy	Delay (ms)	Tracking error $r - y_s$ (mean/st.dev.) [0-5] s	[5-10] s	[10-15] s
CB $S = 20$	25	0.026/0.002	0.068/0.004	0.034/0.002
CB $S = 30$	30	0.026/0.002	0.054/0.002	0.029/0.003
CB $S = 45$	37	0.025/0.002	0.039/0.002	0.026/0.002
CB $S = 90$	60	0.026/0.002	0.028/0.003	0.026/0.003
SB $S = 20$	25	0.025/0.003	0.045/0.003	0.026/0.003
SB $S = 30$	30	0.024/0.003	0.035/0.003	0.025/0.003
SB $S = 45$	37	0.024/0.002	0.029/0.003	0.024/0.003
SB $S = 90$	60	0.026/0.002	0.025/0.003	0.024/0.003
ASB	30 - 68 - 53	0.025/0.003	0.031/0.004	0.024/0.003

TABLE IV

EFFECT OF NOISE REDUCTION ON TRACKING ERROR.

Buffering strategy	Noise Variance	Tracking error (%)
CB (no smoothing, $S = 45$)	0.04	+57.4
CB (no smoothing, $S = 45$)	0.02	+32.5
CB ($S = 45$)	0	0
SB (smoothing, $S = 45$)	0.02	+0.35
SB (smoothing, $S = 45$)	0.04	+3.9

REFERENCES

[1] R. Gupta and M. Chow, "Networked control system: Overview and research trends," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 7, pp. 2527–2535, 2010.
[2] X. Luan, P. Shi, and F. Liu, "Stabilization of networked control systems with random delays," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 9, pp. 4323–4330, 2011.
[3] H. Zhang, Y. Shi, and A. Mehr, "Robust static output feedback control and remote pid design for networked motor systems," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 12, pp. 5396–5405, 2011.
[4] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.
[5] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. Sastry, "Foundations of control and estimation over lossy networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 163–187, 2007.
[6] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Kalman filtering with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.
[7] G. Pin and T. Parisini, "Networked predictive control of uncertain constrained nonlinear systems: Recursive feasibility and input-to-state stability analysis," *Automatic Control, IEEE Transactions on*, vol. 56, no. 1, pp. 72–87, 2011.
[8] G. Liu, Y. Xia, J. Chen, D. Rees, and W. Hu, "Networked predictive control of systems with random network delays in both forward and feedback channels," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 3, pp. 1282–1297, 2007.
[9] A. Onat, T. Naskali, E. Parlakay, and O. Mutluer, "Control over imperfect networks: Model-based predictive networked control systems," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 3, pp. 905–913, 2011.
[10] H. Li, Z. Sun, M. Chow, and F. Sun, "Gain-scheduling-based state feedback integral control for networked control systems," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 6, pp. 2465–2472, 2011.

[11] T. Matiakis, S. Hirche, and M. Buss, "Control of networked systems using the scattering transformation," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 1, pp. 60–67, 2009.
[12] M. Tabbara, A. Rantzer, and D. Nešić, "On controller & capacity allocation co-design for networked control systems," *Systems & Control Letters*, vol. 58, no. 9, pp. 672–676, 2009.
[13] R. Muradore, D. Quaglia, and P. Fiorini, "Adaptive LQ Control over Differentiated Service Lossy Networks," in *World Congress of the International Federation of Automatic Control, (IFAC), Milan*, 2011.
[14] ——, "Predictive control of networked control systems over differentiated services lossy networks," in *Design, Automation & Test in Europe, (DATE), Dresden*, 2012.
[15] P. Martí, J. Yépez, M. Velasco, R. Villà, and J. Fuertes, "Managing quality-of-control in network-based control systems by controller and message scheduling co-design," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 6, pp. 1159–1167, 2004.
[16] M. Cloosterman, L. Hetel, N. Van De Wouw, W. Heemels, J. Daafouz, and H. Nijmeijer, "Controller synthesis for networked control systems," *Automatica*, vol. 46, no. 10, pp. 1584–1594, 2010.
[17] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. IEEE Computer Society Press, 1994, pp. 680–688.
[18] M. Narbutt, A. Kelly, L. Murphy, and P. Perry, "Adaptative VoIP playout scheduling: Assesssing user satisfaction," *IEEE Internet Computing*, vol. 9, no. 4, pp. 28–34, 2005.
[19] L. Atzori and M. L. Lobina, "Playout buffering IP telephony: A survey discussing problems and approaches," *IEEE Communications Surveys&Tutorials*, vol. 8, no. 3, pp. 36–46, 2006.
[20] S. Tavakoli and M. Tavakoli, "Optimal tuning of PID controllers for first order plus time delay models using dimensional analysis," in *Control and Automation, 2003. ICCA'03. Proceedings. 4th International Conference on*. IEEE, 2003, pp. 942–946.
[21] D. Quevedo and D. Nešić, "Input-to-state stability of packetized predictive control over unreliable networks affected by packet-dropouts," *Automatic Control, IEEE Transactions on*, vol. 56, no. 2, pp. 370–375, feb. 2011.
[22] G. Alldredge, M. Branicky, and V. Liberatore, "Play-back buffers in networked control systems: Evaluation and design," in *American Control Conference, 2008*. IEEE, 2008, pp. 3106–3113.
[23] V. Liberatore, "Integrated play-back, sensing, and networked control," in *IEEE Infocom*, 2006.
[24] K. Åström and B. Wittenmark, *Computer-controlled systems*. Prentice-Hall, Inc., 1997.
[25] C. De Boor, *A practical guide to splines*. Springer Verlag, 2001, vol. 27.
[26] R. F. H. Schulzrinne, S. Casner and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.
[27] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. c1–269, 24 2008.
[28] M. Baldi, Y. Ofek, and B. Yener, "Adaptive group multicast with time-driven priority," *Networking, IEEE/ACM Transactions on*, vol. 8, no. 1, pp. 31–43, feb 2000.
[29] M. Han, Y. Lee, S. B. Moon, K. Jang, and D. Lee, "CRAWDAD trace set kaist/wibro/seoul (v. 2008-06-04)," Downloaded from http://crawdad.cs.dartmouth.edu/kaist/wibro/seoul, June 2008.