Control over Unreliable Networks affected by Packet Erasures and Variable Transmission Delays

Daniel E. Quevedo*, *Member, IEEE,* Eduardo I. Silva*, Graham C. Goodwin*, *Fellow, IEEE*

Index Terms— **Control over networks, packet erasures, time-delays, stability.**

Abstract— **This paper describes a novel control strategy aimed at achieving good performance over an unreliable communication network affected by packet loss and variable transmission delays. The key ingredient in the method described here is to use the large data packet frame size of typical modern communication protocols to transmit control sequences which cover multiple data-dropout and delay scenarios. Stability and performance of the resultant scheme are addressed under nominal networked conditions. Simulations verify that the strategy performs exceptionally well under realistic conditions with noise and unmeasured disturbances.**

I. Introduction

There has been a trend towards the implementation of closed loop control systems using digital networks; see, e.g. [1]. In particular, Ethernet in its wired (hub-based and switched) and wireless forms (IEEE 802.11) is increasingly being adopted as a low level control network technology, see [2], [3]. The reasons for this move towards *Networked Control Systems* (NCS's) are manifold, including lower cost, higher reliability, interoperability of devices, and easier installation and maintenance.

From a control design perspective, many interesting challenges are associated with NCS's. For example, due to the inherent bit-rate limitations associated with digital networks, signals need to be coded and quantized prior to transmission [4]. Furthermore, the network may induce variable delays and data-dropouts [5]. Not surprisingly, designs made for non-networked control systems (i.e., where communication links are transparent) will often give poor performance when used in NCS's. Consequently, successful NCS design methods need to consider both control and communication aspects.

Through a complete, and often complex, re-design of the control algorithm, good NCS performance can often be achieved; see, e.g., [6]. However, given the power and sophistication of design methods for non-networked control systems, there exists a strong incentive to extend existing non-networked designs to NCS situations.

In the present work we will show how control laws designed for non-networked control systems can be embellished so as to achieve good performance when used in an NCS. We focus on NCS's having an unreliable link affected by both data-dropouts and time delays. The delays are assumed stochastic in nature, possibly larger than one sampling period and unknown in advance. In our approach, we take advantage of the fact that in contemporary IP based networks over Ethernet, data is sent in large packets. Also, the data can be easily time-stamped, e.g., by invoking the Real Time Transport Protocol (RTP). Thus, rather than sending individual values, finite-length signal predictions can be transmitted. By taking into account multiple transmission outcomes, we show that the resultant NCS is equivalent to an NCS without time-delays and with a data-dropout probability which can be made arbitrary small by choice of design parameters. Thus, our proposal amounts to a special scheme for time-delay and dropout compensation tailored to contemporary communication technology. The method allows one to achieve NCS performance which is close to that of the corresponding non-networked system. An important feature of our proposal is its versatility: namely, it can be used with any plant and any controller.

To illustrate the key ideas, we consider a single-plant, single-controller NCS architecture where the communication network is placed in the control-link, i.e., between controller and plant input.[1] NCS's where both estimation- and control-link use a network can then be designed by using certainty equivalence (although, in general, separation does not hold), see also [6]. Design for unreliable estimation-links has also been studied, e.g., in [9]–[11]. In addition, it has been suggested [6], [12] that it may be advantageous to use architectures with distributed intelligence.

Our proposal is loosely related to various NCS design methods where signal predictions are transmitted, see, e.g., [13]–[20]. The distinguishing, and novel, aspect of our approach is that it incorporates an underlying closed loop control law which is formulated based on predictions of all possible future plant state scenarios (within our model). This allows us to address performance and stability for the associated NCS's directly in the design process. Our results apply to constrained non-linear unstable multiple-input plant models controlled over unreliable communication links affected by delays and data-dropouts, which are both unknown and variable. We believe this to represent a significant advance on existing packet-based NCS design strategies of the type surveyed above.

The motivation for the circle of ideas presented here comes from closed loop control as described in [21]. (Also called "planning with recourse" in some fields; see, e.g., [22], [23].) These strategies involve the determination of an optimal policy that relates the future information states to actions. These kinds of optimization problems are typically intractable due

*School of Electrical Engineering & Computer Science, The University of Newcastle, Callaghan, NSW 2308, Australia; Emails: dquevedo@ieee.org, eduardo.silva@studentmail.newcastle.edu.au, graham.goodwin@newcastle.edu.au

[1]Note that the estimation-link situation (i.e., where the network is situated between plant sensors and controller input) can be treated by adapting standard Kalman Filter techniques, see, e.g., [5], [7], [8].

to computational complexity. In the situation studied in this paper, however, the future scenarios (see [24], [25]) have finite cardinality since only packet loss and/or delays need to be considered. Indeed, the number of scenarios is relatively small, making it fairly straightforward to implement our proposal.

II. Networked Control System Configuration

We consider a discrete-time (possibly unstable) nonlinear plant model described in state-space form via:

$$x(\ell+1) = f(x(\ell), u(\ell)). \tag{1}$$

The plant input and state are constrained according to:

$$u(\ell) \in \mathbb{U} \subseteq \mathbb{R}^{\nu}, \quad x(\ell) \in \mathbb{X} \subseteq \mathbb{R}^{\eta}, \qquad \forall \ell \in \mathbb{N}_0 \triangleq \mathbb{N} \cup \{0\}, \tag{2}$$

where $\mathbb{N}$ denotes the set of positive integers, ν denotes the dimension of the plant input, whilst η refers to the state dimension.

To achieve some desired behaviour of the plant variables, the plant input $\{u(\ell)\}$ is manipulated by a controller. There exist many control strategies that can be utilized in a traditional (non-networked) setting, i.e., where the communication between controller and plant is transparent. In particular, discrete-time controllers operating at the same sampling rate as the plant can typically be designed to stabilize the plant model and make (parts of) $\{x(\ell)\}$ track a given time-varying reference $\{r(\ell)\}$; see, e.g., [26]–[30]. In the non-networked case, the plant input is given by a, possibly time-varying and dynamic, mapping of the plant state, say

$$u(\ell) = \kappa_\ell(x(\ell)), \quad \ell \in \mathbb{N}_0, \tag{3}$$

where $\kappa_\ell \colon \mathbb{X} \to \mathbb{U}$ is the control policy.

In the present work, we will address the more challenging situation, where a discrete-time controller communicates with the plant input via an unreliable network. This network is assumed to be able to carry large packets of data (say, a few kilobytes). However, each packet, say $U(\ell)$, may be delayed or even lost (due to buffer overflows and transmission errors). If quantization issues are to be explicitly taken into account, then $\kappa_\ell(\cdot)$ in (3) should be chosen as a quantized control policy, such as the finite set constrained controller of [31].[2]

The delay experienced by $U(\ell)$ has both a fixed component, which can be included in the plant model (1), and a time-varying component, say $\tau(\ell)$. The latter depends on several factors including network load. Since we will concentrate on a configuration where sensors, actuators and controller operate at the same sampling rate[3], it suffices to consider $\tau(\ell) \in \mathbb{N}_0$. Packets which are delayed by more than a given value, say $\tau_{\max}$, will be regarded as lost. Thus, we will use the term "lost" to refer to those packets which are effectively dropped by the network as well as to those which do not arrive during a prespecified time frame. Lost packets will not be used further in the NCS. The maximum delay $\tau_{\max}$ constitutes a "timeout" value which can be designed based upon network delay and dropout characteristics, see, e.g., [33]. (We will see in Section VI that the timeout value also has an impact on the complexity of the networked controller.) The model adopted above includes pure erasure channels (see, e.g., [7], [19], [20]), as a particular case, by setting $\tau_{\max} = 0$.

[2]Whether quantization aspects need to be explicitly taken into account or not, depends upon how large their effect on achieved closed loop performance is, when compared to other quantities, including the network induced delays and dropouts. Interestingly, simple additive quantization noise models often give good results even for rates as low as 2 bits per sample, see, e.g., [32]. Moreover it is known that when the quantization effect can be modeled as additive noise, then it suffices to use a non-quantized control law $\kappa_\ell(\cdot)$, see also [12]. Note that, if (1) is open-loop unstable, then to achieve closed loop stability, bit-rates need to be larger than the lower bounds described in [4].

[3]We consider time-driven rather than event-driven systems.

In the sequel, we will present a method to use available communication resources to control the plant (1). In particular, we will show how closed loop properties of a non-networked control system can be preserved when it is implemented as an NCS. We will focus on the link between controller output and plant input. Accordingly, we will assume that the networked controller has access to the plant state.

III. Scenario-Based Networked Control

To achieve acceptable performance, NCS's need to be designed taking into account properties of both the plant and the communication link. In particular, with an unreliable network situated between controller and plant input, the controller should be aware that, in addition to possible quantization constraints, data sent may be delayed or lost. Here, we will exploit the fact that, in todays' networks, data can be sent as large packets.

In our proposal, at each time instant k, the controller sends a packet $U(k)$ to the plant input side. To take account of possible future transmission errors, $U(k)$ contains not only a desired value for $u(k)$, but also predicted plant inputs to be implemented at a finite number of future time instants. In addition, since $U(k)$ may be delayed by $\tau(k)$, which is unknown to the controller at time k, the plant input sequences are calculated for all possible delay values up to the point where the packet is deemed lost, i.e., for $\tau(k) \in \{0, 1, \dots, \tau_{\max}\}$. At the plant input side, the received plant input sequences are buffered according to a specific selection logic. Fig. 1 illustrates the overall NCS configuration. The design of the buffer selection logic and control packet are closely related. We will describe both in the sequel.

A. Buffer Selection Logic

We will first describe the selection logic at the receiver side, i.e., at the plant input. Here, a buffer, say $\vec{b}(k)$, contains the values to be passed on to the actuator; fresh values replace values sent earlier. For example, let us suppose that packet $U(k)$ is received (without error). Thus, it arrives at time instant $k + \tau(k)$. If none of the packets

$$\{U(k+1), U(k+2), \dots, U(k+\tau(k))\} \tag{4}$$

have been received by time $k + \tau(k)$, then $U(k)$ is used when it arrives. Otherwise, if any of the more recent packets in (4) have been received by time $k + \tau(k)$, then $U(k)$ is discarded.

For further reference, we will denote the entire set of packets after discarding by $\mathcal{U}$, the sequence of time instants when these

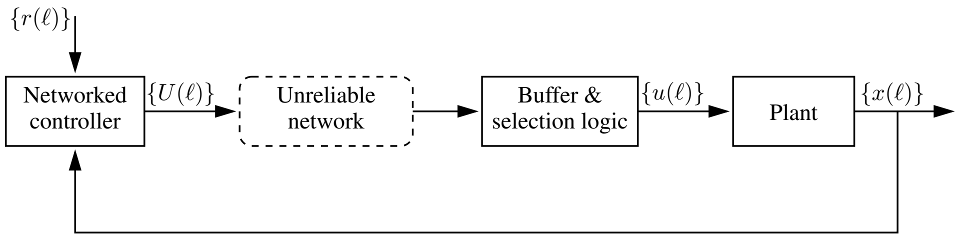

Fig. 1. Networked Control System Configuration.

fresh packets are generated via $\mathcal{K} = \{k_i\}_{i\in\mathbb{N}}$, and the arrival instants as $\mathcal{N} = \{n_i\}_{i\in\mathbb{N}}$, where

$$n_i = k_i + \tau(k_i), \quad \forall i \in \mathbb{N}. \tag{5}$$

We will assume that n_1 is finite.

B. Scenarios for Control Calculations

To design $U(k)$, the controller should ideally know the plant state at the arrival time $k + \tau(k)$. If there were no delay associated with $U(k)$, i.e., if $\tau(k) = 0$, then the plant state needed would be simply $x(k)$, which we assume known. However, for finite non-zero delays, $x(k+\tau(k))$ depends upon $x(k)$ and upon the future plant inputs contained in

$$\vec{u}(k) \triangleq \{u(k), u(k+1), \dots, u(k+\tau(k)-1)\}, \ \tau(k) \geq 1. \tag{6}$$

Given the buffer selection logic employed, $\vec{u}(k)$ is completely determined by $\vec{b}(k-1)$ and by packets other than $U(k)$. The relevant information for determining $U(k)$ is contained in the finite set of pairs:

$$\mathcal{F}(k) = \Big\{ \, (U(k_i), \tau(k_i)) : U(k_i) \in \mathcal{U}, \ k_i < k, \\ k_i + \tau(k_i) \in \{k, k+1, \dots, k+\tau(k)-1\}\Big\}.$$

If acknowledgments form part of the network protocol and if the one-way delay from the buffer to the controller is smaller than 1 sampling period, then $\vec{b}(k-1)$ will be known to the controller at time k. However, the controller cannot foresee the future. In particular, at time k, the controller does not know $\mathcal{F}(k)$ or $\tau(k)$.

The above observation motivates us to develop a control strategy which utilizes only the available information at the controller side at time k (namely, $\vec{b}(k-1)$ and previously sent packets), and determines $U(k)$ by taking into account all possibilities for $\tau(k) \in \{0, 1, \dots, \tau_{\max}\}$ and for $\mathcal{F}(k)$. We will denote the associated set of all possible plant input scenarios to be considered when calculating $U(k)$ via:

$$\mathcal{S}_\tau(k) \triangleq \{\vec{s}_\tau^{\,(i)}(k)\}_{i\in\{1,\dots,|\mathcal{S}_\tau(k)|\}}, \ \tau \in \{0, 1, \dots, \tau_{\max}\}, \tag{7}$$

where $|\mathcal{S}_\tau(k)|$ denotes the cardinality of $\mathcal{S}_\tau(k)$. Clearly, $|\mathcal{S}_\tau(k)|$ is always finite, and depends on τ and on when the content of the buffer at time instant $k-1$, i.e., $\vec{b}(k-1)$, was generated. Definition (7) implies that, if the actual delay were to be $\tau(k) \in \{0, 1, \dots, \tau_{\max}\}$ and the scenario were $i^\star \in \{1, 2, \dots, |\mathcal{S}_{\tau(k)}(k)|\}$, then the future plant inputs would be given by:

$$\vec{u}(k) = \vec{s}_{\tau(k)}^{\,(i^\star)}(k).$$

Therefore, the actual plant input sequence in (6) satisfies:

$$\vec{u}(k) \in \mathcal{S}_\tau(k), \quad \text{for some } \ \tau \in \{0, 1, \dots, \tau_{\max}\}.$$

In the undelayed case, where $\tau = 0$, there are no future plant inputs in $\vec{u}(k)$. Accordingly, we define (for future reference):

$$\mathcal{S}_0(k) \triangleq \{\vec{s}_0^{\,(1)}(k)\}, \quad \vec{s}_0^{\,(1)}(k) \triangleq \{\,\}, \quad |\mathcal{S}_0(k)| = 1. \tag{8}$$

Further characterizations of the total number of scenarios to be considered at each time instant will be given in Section VI.

Whenever necessary, i.e., when $\vec{b}(k-1)$ does not contain enough useful plant input values or not enough packets arrive in the situation considered, then the input scenarios in (7) will be determined by assuming that plant input values are provided by a given open-loop policy, say $u(\ell) = \kappa^f(\ell)$, which depends only on received packets up to time ℓ. Simple examples include keeping the current value or setting the plant input to zero.

To achieve good performance in the presence of unreliable communication, the networked controller should be aware of all possible plant input scenarios, $\mathcal{S}_\tau(k)$, in (7) and (8). In the following, we will present a strategy which embodies this requirement.

Remark 1 (Network Protocols without Acknowledgments): In the above, we have assumed that $\vec{b}(k-1)$ is known to the controller at time k. If the one-way delay from the buffer to the controller is larger than one sampling period or also if simple UDP-like protocols are used, this will not be the case. Here, a larger (although still finite) number of scenarios needs to be considered by the controller. Alternatively, $\vec{b}(k-1)$ could be estimated based on plant state information. We note that the simplification in communications technology, comes here at the expense of larger controller complexity. $\triangle$

C. Control Packet Design

To keep the exposition simple, we will assume in the sequel that the networked controller knows the previous buffer contents[4] $\vec{b}(k-1)$. We then propose that, at every time instant k, the controller calculates control sequences associated with each of the possible plant states $x(k+\tau)$, $\tau \in \{0, 1, \dots, \tau_{\max}\}$, resulting from the input scenarios in (7) and (8). The packet to be sent is formed as:

$$U(k) = \Big\{ \big\{\vec{u}(k; 0, \vec{s}_0^{\,(1)}(k))\big\}, \big\{\vec{u}(k; 1, \vec{s}_1^{\,(i)}(k))\big\}_i, \dots \\ \dots, \big\{\vec{u}(k; \tau_{\max}, \vec{s}_{\tau_{\max}}^{\,(n)}(k))\big\}_n \Big\}, \tag{9}$$

[4]If $\vec{b}(k-1)$ were unknown to the controller, then the networked control strategy to be described could still be applied, recall Remark 1.

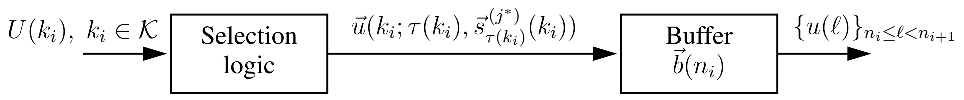
$U(k_i),\ k_i \in \mathcal{K}$
Selection logic
$\vec{u}(k_i;\tau(k_i),\vec{s}^{\,(j^*)}_{\tau(k_i)}(k_i))$
Buffer $\vec{b}(n_i)$
$\{u(\ell)\}_{n_i\le\ell<n_{i+1}}$

Fig. 2. Selection logic at the plant input side.

where $\vec{u}(k;\tau,\vec{s}^{\,(j)}_\tau(k))$ are sequences of length $N \geq 1$:

$$\vec{u}(k;\tau,\vec{s}^{\,(j)}_\tau(k)) = \big\{u(k+\tau;\tau,\vec{s}^{\,(j)}_\tau(k)),\dots \\ \dots,u(k+\tau+N-1;\tau,\vec{s}^{\,(j)}_\tau(k))\big\}. \quad (10)$$

In (10), $u(k+\tau+\ell;\tau,\vec{s}^{\,(j)}_\tau(k))$ corresponds to the plant input, which will be applied at time $k+\tau+\ell$, whenever all three of the following conditions are satisfied:

1) The packet $U(k)$ arrives at time $k+\tau$ (errorless).
2) No packets $U(k+m)$, $m>1$ have been received up to time $k+\tau$.
3) The plant input over the interval $\{k,k+1,\dots,k+\tau-1\}$ equals $\vec{s}^{\,(j)}_\tau(k)$.

In (10), the prediction horizon length N is a design parameter. This allows the designer to trade-off computational complexity and network usage against performance. (More insight into this trade-off will be given in Sections V and VI.)

Each of the elements in (10) results from the underlying policy $\kappa_\ell(\cdot)\colon \mathbb{X}\to\mathbb{U}$, see (3). The key aspect, in the present NCS setting, is that $\kappa_\ell(\cdot)$ uses plant state predictions. To be more precise, for $\tau\geq 1$ and $j\in\{1,2,\dots,|\mathcal{S}_\tau(k)|\}$, the first element in (10) is given by:

$$u(k+\tau;\tau,\vec{s}^{\,(j)}_\tau(k)) = \kappa_{k+\tau}\left(x(k+\tau;x(k),\vec{s}^{\,(j)}_\tau(k))\right), \quad (11)$$

where $x(k+\tau;x(k),\vec{s}^{\,(j)}_\tau(k))$ is a prediction of the plant state at time $k+\tau$. This predictor results from iterating the system equation (1) from initial state $x(k)$ and with input sequence $\vec{s}^{\,(j)}_\tau(k)$.[5]

For the case $\tau=0$, which corresponds to the ideal situation when $U(k)$ arrives with no delay, we simply set

$$x(k;x(k),\{\}) = x(k), \quad u(k;0,\{\}) = \kappa_k(x(k)). \quad (12)$$

For $\tau\geq 0$ and $j\in\{1,2,\dots,|\mathcal{S}_\tau(k)|\}$, the remaining $N-1$ elements of (10) are then calculated recursively via:

$$x(k+\tau+\ell+1;x(k),\vec{s}^{\,(j)}_\tau(k)) \\ = f(x(k+\tau+\ell;x(k),\vec{s}^{\,(j)}_\tau(k))),u(k+\tau+\ell;\tau,\vec{s}^{\,(j)}_\tau(k))$$

and

$$u(k+\tau+\ell+1;\tau,\vec{s}^{\,(j)}_\tau(k))) \\ = \kappa_{k+\tau+\ell+1}\left(x(k+\tau+\ell+1;x(k),\vec{s}^{\,(j)}_\tau(k)\right), \quad (13)$$

where $\ell\in\{0,1,\dots,N-1\}$. The initial values are as in (11) or (12).

Remark 2 (Effective Bit-rates): It is worth mentioning that, in general, the values in (9) which refer to a given time instant for different scenarios are highly correlated. Thus, one can achieve efficient compression of the packets by applying scalar- or vector-quantization and joint entropy coding [34].

[5]If $x(k)$ is unknown at the controller side, then an estimate should be used. As documented in Section VIII-B, good results can often be achieved even in the output feedback case with measurement noise.

D. Resultant Plant Input

To characterize the plant input, we note that, at the buffer side, it will always be known which packets have been received so far. Thus, whenever a fresh packet $U(k_i)\in\mathcal{U}$ arrives, the appropriate scenario from $\mathcal{S}_{\tau(k_i)}(k_i)$ can be identified and the buffer can be updated accordingly. Say this scenario is $\vec{s}^{\,(j^*)}_{\tau(k_i)}(k_i)$, then the buffer content at time $n_i = k_i+\tau(k_i)$ is set to:

$$\vec{b}(n_i) \leftarrow \vec{u}(k_i;\tau(k_i),\vec{s}^{\,(j^*)}_{\tau(k_i)}(k_i)),$$

see (9). Thus, only the N values associated with the actual delay experienced by $U(k_i)$, namely $\tau(k_i)$, are stored in the buffer. These values are kept until time instant $n_{i+1}\in\mathcal{N}$, see (5).

The plant input at subsequent times is then given by the elements of $\vec{b}(n_i)$, i.e.,

$$u(n_i+\ell) \leftarrow u(n_i+\ell;\tau(k_i),\vec{s}^{\,(j^*)}_\tau(k_i)), \\ \forall\ell\in\{0,1,\dots,n_{i+1}-n_i-1\}. \quad (14)$$

until a newer packet arrives. The buffering scheme is illustrated in Fig. 2.

We will call this control strategy *Scenario-Based Networked Control* (SBNC).

Before proceeding, we note that, from (14), it follows that, to avoid running out of data, we require that

$$n_{i+1}-n_i \leq N, \quad \forall n_i,n_{i+1}\in\mathcal{N}. \quad (15)$$

In a practical implementation of the SBNC, whenever the buffer runs out of data, the plant input will be governed by the open-loop control policy $\kappa^f(\ell)$ used when determining the plant input scenarios, see Section III-B.

Remark 3 (Relationship to Receding Horizon Control): SBNC uses control signal predictions and, thus, bears some connections to Receding Horizon Control [35]. It should be noted, however, that SBNC can be formulated based upon *any* state feedback control policy $\kappa_\ell(\cdot)$, see (3). The control sequences sent are obtained by evaluating $\kappa_\ell(\cdot)$ for all plant state predictions (within the model). This explicitly takes into account possible delays and dropouts of all relevant packets. The main feature, which distinguishes SBNC from other approaches, is that all sequences are sent to the plant input buffer, not only those associated with "nominal" or "worst case" scenarios. This stands in stark contrast to standard Receding Horizon Control methods, where only a single control value is used, which is calculated for the current plant state. It also differs to recent extensions of Receding Horizon Control to NCS's, such as [13]–[20], where a single plant input prediction sequence is sent. $\triangle$

IV. A Specific Case

To provide additional insight into SBNC, we will consider a simple case consisting of an (unconstrained) linear plant obeying (1) with $f(x,u)=Ax+Bu$ and where $A\in\mathbb{R}^{\eta\times\eta}$ and $B\in\mathbb{R}^{\eta\times\nu}$. The underlying control policy is simply chosen as the proportional state feedback controller $\kappa_\ell(x(\ell))=Kx(\ell)-Gr(\ell)$, where $\{r(\ell)\}_{\ell\in\mathbb{N}_0}$ is the reference signal, and K and G are given gain matrices. The prediction horizon length was chosen as $N=5$. The maximum (allowed) time delay was set at $\tau_{\max}=2$.

Let us suppose that the packet $U(k-2)$ arrives at time $k-1$ whilst $U(k-1)$ has not been received by time $k-1$. Then, the buffer sequence satisfies

$$\vec{b}(k-1)=\{u_{-1},u_0,u_1,u_2,u_3\},$$

where $\{u_{-1},u_0,u_1,u_2,u_3\}=\vec{u}(k-2;1,\{u(k-2)\})$ and $u(k-2)$ refers to the past plant input, see (1). Following (14), these values are implemented until newer values arrive. In particular, we have $u(k-1)=u_{-1}$.

To compute $U(k)$ at time k, the SBNC examines the possible transmission outcomes for $U(k-1)$. This involves the case where $U(k-1)$ is dropped and the two allowed delay situations, namely, $\tau(k-1)\in\{1,2\}$.

If $U(k-1)$ is lost, then only $\vec{b}(k-1)$ needs to be considered. For $\tau(k-1)\in\{1,2\}$, we denote the elements of the corresponding sequences via:

$$\begin{aligned}\vec{u}(k-1;1,\{u_{-1}\})&=\{\bar{u}_0,\bar{u}_1,\dots,\bar{u}_4\}\\ \vec{u}(k-1;2,\{u_{-1},u_0\})&=\{\underline{u}_1,\underline{u}_2,\dots,\underline{u}_5\},\end{aligned}$$

where u_{-1} and u_0 stem from $\vec{b}(k-1)$.

The sets of scenarios used for calculating $U(k)$ are then:

$$\begin{aligned}\mathcal{S}_1(k)&=\left\{\vec{s}_1^{(1)}(k),\vec{s}_1^{(2)}(k)\right\},\\ \mathcal{S}_2(k)&=\left\{\vec{s}_2^{(1)}(k),\vec{s}_2^{(2)}(k),\vec{s}_2^{(3)}(k)\right\},\end{aligned}$$

where:

$$\begin{aligned}&\vec{s}_1^{(1)}(k)=\{u_0\},\ \vec{s}_1^{(2)}(k)=\{\bar{u}_0\},\ \vec{s}_2^{(1)}(k)=\{u_0,u_1\},\\ &\qquad\vec{s}_2^{(2)}(k)=\{u_0,\underline{u}_1\},\ \vec{s}_2^{(3)}(k)=\{\bar{u}_0,\bar{u}_1\}.\end{aligned}\tag{16}$$

This gives $|\mathcal{S}_0(k)|=1$, $|\mathcal{S}_1(k)|=2$, $|\mathcal{S}_2(k)|=3$ and the total number of scenarios to be considered at time k is equal to 6.

If we denote the plant state at time k as $x(k)=x_0$, then the corresponding plant state predictions at times $k+\tau$, $\tau\in\{0,1,2\}$ are:

$$\begin{aligned}x(k;x_0,\{\})&=x_0,\\ x(k+1;x_0,\vec{s}_1^{(1)}(k))&=f(x_0,u_0),\\ x(k+1;x_0,\vec{s}_1^{(2)}(k))&=f(x_0,\bar{u}_0),\\ x(k+2;x_0,\vec{s}_2^{(1)}(k))&=f(x(k+1;x_0,\vec{s}_1^{(1)}(k)),u_1)\\ &=f(f(x_0,u_0),u_1),\\ x(k+2;x_0,\vec{s}_2^{(2)}(k))&=f(x(k+1;x_0,\vec{s}_1^{(1)}(k)),\underline{u}_1)\\ &=f(f(x_0,u_0),\underline{u}_1),\\ x(k+2;x_0,\vec{s}_2^{(3)}(k))&=f(x(k+1;x_0,\vec{s}_1^{(2)}(k)),\bar{u}_1)\\ &=f(f(x_0,\bar{u}_0),\bar{u}_1).\end{aligned}\tag{17}$$

These values can be calculated recursively. The situation is depicted in Fig. 3.

$$\begin{tikzcd}
x_0 \arrow[r, "u_0"] \arrow[rdd, "\bar{u}_0"'] & x(k+1;x_0,\{u_0\}) \arrow[r, "u_1"] \arrow[rd, "\underline{u}_1"'] & x(k+2;x_0,\{u_0,u_1\}) \\
 & & x(k+2;x_0,\{u_0,\underline{u}_1\}) \\
 & x(k+1;x_0,\{\bar{u}_0\}) \arrow[r, "\bar{u}_1"] & x(k+2;x_0,\{\bar{u}_0,\bar{u}_1\})
\end{tikzcd}$$

Fig. 3. Construction of plant state predictions for times $k+\tau$, $\tau\in\{0,1,2\}$. The first row corresponds to the situation where $U(k-1)$ is dropped, the second row to $\tau(k-1)=2$, the third to $\tau(k-1)=1$.

The state predictions in (17) are used to calculate the elements of $U(k)$. More precisely, the current state x_0 is used to calculate $\vec{u}(k;0,\{\})$; the state predictions in the second column (see Fig. 3) are used to calculate $\vec{u}(k;1,\vec{s}_1^{(j)}(k))$, $j\in\{1,2\}$; those in the third column are used to calculate $\vec{u}(k;2,\vec{s}_2^{(j)}(k))$, $j\in\{1,2,3\}$. The resultant control packet is then given by

$$\begin{aligned}U(k)=\Bigg\{&\left\{\vec{u}(k,0,\vec{s}_0^{(1)}(k))\right\},\\ &\left\{\vec{u}(k,1,\vec{s}_1^{(1)}(k)),\vec{u}(k,1,\vec{s}_1^{(2)}(k))\right\},\\ &\left\{\vec{u}(k,2,\vec{s}_2^{(1)}(k)),\vec{u}(k,2,\vec{s}_2^{(2)}(k)),\vec{u}(k,2,\vec{s}_2^{(3)}(k))\right\}\Bigg\},\end{aligned}$$

where, for every $\vec{s}_\tau^{(i)}$ in (16),

$$\begin{aligned}\vec{u}(k,\tau,\vec{s}_\tau^{(i)}(k))=\Big\{&u(k+\tau,\tau,\vec{s}_\tau^{(i)}(k)),\cdots\\ &\dots,u(k+\tau+4,\tau,\vec{s}_\tau^{(i)}(k))\Big\},\end{aligned}$$

with[6]

$$u(k+\ell,\tau,\vec{s}_\tau^{(i)}(k))=K(x_\ell-r(k+\ell)),\ \ell\in\{\tau,\cdots,\tau+4\}.\tag{18}$$

In (18), plant state predictions are formed as:

$$x_{\ell+1}=\begin{cases}Ax_\ell+B[\vec{s}_\tau^{(i)}(k)]_{\ell+1}, & 0\le\ell\le\tau-1\\ Ax_\ell+BKx_\ell-BGr(k+\ell), & \tau\le\ell\le\tau+3,\end{cases}$$

with $\vec{s}_\tau^{(i)}(k)=\left\{[\vec{s}_\tau^{(i)}(k)]_1,\cdots,[\vec{s}_\tau^{(i)}(k)]_\tau\right\}$.

Note that, in the case considered so far, the only uncertainty that needs to be taken into account to calculate $U(k)$ is that related to the transmission of $U(k-1)$. However, if neither $U(k-1)$ nor $U(k-2)$ were received at time $k-1$, then more scenarios would need to be considered. Indeed, if we denote:

$$\vec{u}(k-2;2,\{u_{-2},u_{-1}\})=\{\hat{u}_0,\hat{u}_1,\dots,\hat{u}_4\},$$

then, in this most complex case (remember that $\tau_{\max}=2$), and with the same notation as above, we obtain

$$\begin{aligned}&\mathcal{S}_1(k)=\big\{\{u_0\},\{\hat{u}_0\},\{\bar{u}_0\}\big\},\\ &\mathcal{S}_2(k)=\big\{\{u_0,u_1\},\{u_0,\underline{u}_1\},\{\hat{u}_0,\hat{u}_1\},\{\hat{u}_0,\underline{u}_1\},\{\bar{u}_0,\bar{u}_1\}\big\},\end{aligned}$$

[6]Here we assume that the reference signal is known, at least, $\tau_{\max}+N-1$ steps in advance.

which amounts to $1+3+5=9$ scenarios. In Section VI we will give bounds on the number of scenarios to be considered in the control calculations.

V. Performance of SBNC

In this section we will characterize the performance of SBNC. The main aspect here is that SBNC acts as a safeguard against network effects and, thus, improves the network reliability as seen by the plant.

A. Characterization of SBNC trajectories

The power of SBNC as presented in Section III derives from the fact that knowledge of all possible scenarios is used. The approach adopted is entirely deterministic and, thus, network delay and dropout probability distributions are not needed in the calculations. Nevertheless, the NCS which results from combining SBNC with the plant (1) and an unreliable network will be stochastic. Performance will, thus, depend upon the interplay between the SBNC parameters and the underlying network behavior. This interplay is captured by means of Theorem 1 below. We first define:[7]

$$\begin{aligned} p_{eq}(\ell) &\triangleq 1-\mathcal{P}\left\{\tau(\ell)=0\right\} \\ &- \sum_{\substack{j=0 \\ (i,j)\neq(0,0),\, i+j\leq\ell}}^{N-1} \sum_{i=0}^{\tau_{\max}} \mathcal{P}\Big\{\tau(\ell-i-j)=i\wedge\tau(\ell)>0 \\ &\wedge\tau(\ell-1)>1\wedge\cdots\wedge\tau(\ell-j-i+1)>j+i-1\Big\}. \end{aligned} \tag{19}$$

Theorem 1 (SBNC and erasure channels): Suppose that the plant obeys (1), then the plant state trajectories when controlled by the SBNC over the corresponding unreliable network are given by

$$\begin{aligned} x(\ell+1) &= f(x(\ell),u_r(\ell)) \\ u_r(\ell) &= d_r(\ell)\kappa_\ell(x(\ell))+(1-d_r(\ell))\kappa^f(\ell), \quad \forall\ell\in\mathbb{N}_0. \end{aligned} \tag{20}$$

In (20),

$$d_r(\ell)=\begin{cases}0, & \text{if } 0\leq\ell\leq n_1-1,\\ 1, & \text{if } n_1\leq\ell\leq n_1+N-1,\end{cases} \tag{21}$$

where n_1 is the first "valid" arrival instant, see Section III-A.

For $\ell\geq n_1+N$, $d_r(\ell)$ is a Bernoulli random variable such that:

$$\begin{aligned} \mathcal{P}\left\{d_r(\ell)=0\right\} &= p_{eq}(\ell), \\ \mathcal{P}\left\{d_r(\ell)=1\right\} &= 1-p_{eq}(\ell). \end{aligned} \tag{22}$$

Proof: The proof is included in Appendix A. ■

Theorem 1 states that controlling a nonlinear constrained unstable plant model over an unreliable channel by means of SBNC amounts to controlling the same plant over an erasure channel with *equivalent data dropout probability* $p_{eq}(\ell)$. This important result, implies that, in order to analyze or design SBNC loops, it suffices to consider a setting wherein the network is modeled via an erasure channel with a given dropout probability (see, e.g., [5], [7]–[10], [36] and the many references therein).

It should be emphasized here that if $\{\tau(\ell)\}_{\ell\in\mathbb{N}_0}$ is a sequence of independent random variables, then

$$p_{eq}(\ell)=p_{eq}, \quad \forall\ell\geq\max\{n_1+N, N-1+\tau_{max}\}, \tag{23}$$

i.e., the equivalent probability, is a constant. However, $\{d_r(\ell)\}$ is, in general not a sequence of independent random variables, even if the underlying network delay and dropout distributions are.

The equivalent dropout probability characterizes closed loop control performance. It can be used as a guideline for choosing the horizon length N and the timeout value $\tau_{\max}$. Indeed, $p_{eq}(\ell)$ can be made arbitrarily small (and, thus, the performance will become indistinguishable from that achieved in the non-networked case). This is achieved by choosing sufficiently large values for N and $\tau_{\max}$, albeit at an increase in computational complexity. (This aspect will be further explored in Section VI.) Fortunately, in practice, choosing moderate values for N and $\tau_{\max}$ is often sufficient to achieve good performance as illustrated by the following example:

Example 1 (Effect of N and $\tau_{\max}$ on $p_{eq}(\ell)$): Consider a situation where $\{\tau(\ell)\}_{\ell\in\mathbb{N}_0}$ is a sequence of i.i.d.[8] geometric random variables with parameter λ, i.e.,

$$\mathcal{P}\{\tau(\ell)=i\}=(1-\lambda)^i\lambda, \quad \forall\ell\in\mathbb{N}_0, i\in\mathbb{N}_0. \tag{24}$$

This model (approximately) describes typical end-to-end delay profiles over the Internet, as discussed in [33]. Note that $\lambda\rightarrow1$ corresponds to a more reliable network and $\lambda\rightarrow0$ to a less reliable one. As a specific illustration, Fig. 4 shows the delay distribution for the case $\lambda=0.2$.

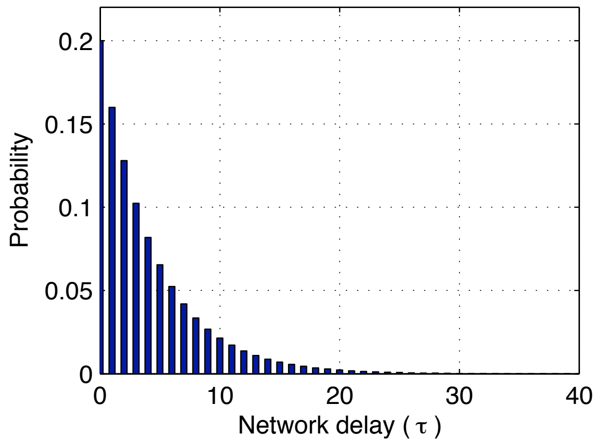

Fig. 4. Delay distribution (24) for $\lambda=0.2$.

For $\mathcal{P}\{\tau(\ell)=i\}$ as in (24), the equivalent dropout probability achieved by SBNC is given by:

$$p_{eq}=1-\lambda-\lambda\sum_{j=0}^{N-1}\sum_{i=0}^{\tau_{\max}}(1-\lambda)^i\prod_{k=0}^{j+i-1}(1-\lambda)^{k+1},$$

see (23) and (19).

[7]Here and in the sequel $\mathcal{P}\{\Omega\}$ stands for probability of Ω.

[8]i.i.d. stands for independent and identically distributed

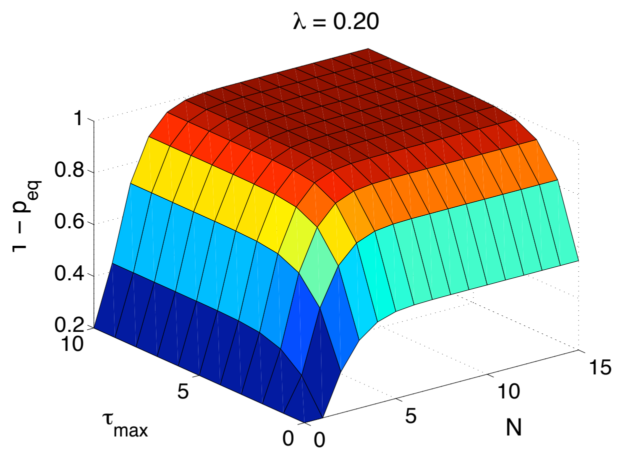

Fig. 5. $1 - p_{eq}$ as a function of N and $\tau_{\max}$ for $\lambda = 0.2$.

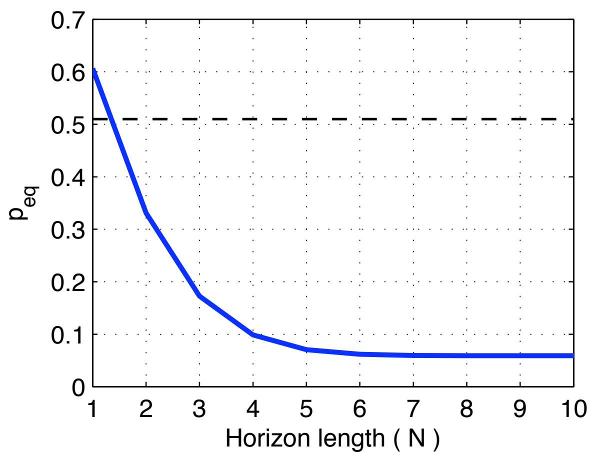

Fig. 6. The equivalent probability p_{eq} as a function of N, for $\lambda = 0.2$ and timeout value $\tau_{\max} = 2$. The dashed line corresponds to the probability of packets deemed lost.

For the illustrative case $\lambda = 0.2$, Fig. 5 shows a graph of p_{eq} as a function of N and $\tau_{\max}$. If the timeout value is set to $\tau_{\max} = 2$, then the probability of packets being considered as lost is equal to the tail probability $\mathcal{P}\{\tau(\ell) > 2\} = 0.51$, see (24).

It can be seen that for a fixed N (resp. fixed $\tau_{\max}$), p_{eq} is monotonic in $\tau_{\max}$ (resp. in N). Also, for moderate values of either N or $\tau_{\max}$, one can make $1 - p_{eq}$ arbitrarily close to 1. Indeed, even for this quite unreliable network, choosing $(N, \tau_{\max}) = (4, 2)$ yields $p_{eq} < 0.1$. For $\tau_{\max} = 2$, horizons $N \geq 7$ give $p_{eq} < 0.06$, which is much smaller than the probability of packets being considered as lost, see Fig. 6.

B. Deterministic Results

As shown in Theorem 1, the use of SBNC leads to a non-deterministic control system, where the stochastic features arise from the fact that the buffer may run out of data with non-zero probability. In some situations, however, it is possible to give deterministic results as follows:

Corollary 1 (SBNC reduces to non-networked control): Suppose that the plant obeys (1) and that (15) holds. Then,

$$x(\ell + 1) = f(x(\ell), \kappa_\ell(x(\ell))), \quad \forall \ell \geq n_1. \tag{25}$$

Proof: If (15) holds, then the buffer never runs out of data and $p_{eq}(\ell) = 0$ for every $\ell \geq n_1$. Expression (25) then follows from Theorem 1. ■

Corollary 1 states the intuitively clear result that, if the buffer never runs out of data, then the *trajectories of the SBNC are exactly the same as in the non-networked case*. This is an interesting result, but may be difficult to achieve in practice. A non-trivial case where this may be achieved arises when the network has no dropouts and the maximum delay value is bounded by, say $\hat{\tau} \in \mathbb{N}$, see, e.g., [14], [37], [38]). In this case, choosing $\tau_{\max} \geq \hat{\tau}$ guarantees that (15) holds and SBNC makes the network transparent (within the model).

If the conditions of Corollary 1 are met, then we can also derive deterministic stability conditions. To that end, we begin by examining the start-up trajectory, namely

$$\vec{x}_0 \triangleq \{x(0), x(1), \ldots, x(n_1)\}.$$

Since no packet arrives before time n_1, the trajectory $\vec{x}_0$ is completely determined by the initial state, say $x(0) = x_0$ and the initial buffer, say $\vec{b}(0) = \{b_0, b_1, \ldots, b_{N-1}\}$.[9] Indeed, $\vec{x}_0$ can be calculated directly from the system:

$$\begin{aligned} x(\ell + 1) &= f(x(\ell), b_\ell), \quad \forall \ell \in \{0, 1, \ldots, n_1 - 1\} \\ x(0) &= x_0 \end{aligned} \tag{26}$$

Equation (26) allows us to conclude the following:

Corollary 2 (Stability): Suppose that $\kappa_\ell(\cdot)$ used in the SBNC is chosen such that the *non-networked* system (1)–(3) is stable (in any deterministic sense) when starting at time $\ell = n_1$ with state $x(n_1)$ given by (26). Then, the same system, having initial state $x(0) = x_0$, when controlled with the SBNC over an unreliable network, is also stable (in the same sense) provided (15) holds.

Proof: This is an immediate consequence of Corollary 1. ■

Corollaries 1 and 2 allow one to establish deterministic performance and stability guarantees for the NCS directly in the design procedure. These guarantees hold for general constrained non-linear stable and unstable plants controlled via an unreliable packet-network with data-dropouts and unknown time-varying delays satisfying (15). This should be contrasted with the results concerning other packet-based NCS approaches, such as, [13]–[18], [20]. In the latter, stability guarantees are restricted to a smaller class of plants or networks and often require careful case-by-case analysis.

Remark 4: We note that (15) is a quite natural demand on the network quality of service, since between successful transmission instants the plant is necessarily operated in open-loop. However, we stress that, as shown in Theorem 1, we do not need this condition to hold for the proper operation of SBNC. Expression (15) is merely a sufficient condition for SBNC to give performance identical to the non-networked

[9] In accordance with (15), we will assume $n_1 \leq N$.

case. If (15) is not always satisfied, then performance is characterized by the results of Section V-A.

Requirements similar to (15) also appear in other packet-based NCS strategies. For instance, for erasure channels (i.e., where $\tau_{\max} = 0$), (15) reduces to N being larger than a bound on the consecutive packet dropouts. In this case, closed loop stability guarantees can also be established by using open-loop predictions [20]. $\triangle$

Remark 5 (Robustness): Our analytical results concern the ideal setting, where the plant is exactly described by (1). Whilst this assumption is commonly made in the literature which deals with stability of NCS's, it would certainly be informative if stability and performance were addressed for more general situations, such as when the plant is affected by unmodeled disturbances, measurement noise or when there is plant-model mismatch. Note that, given the results in [39], [40], there appears to be little hope that deterministic stability and performance guarantees could be obtained for the general framework we are considering in the present work, which, inter-alia, includes unstable constrained nonlinear plants controlled via networks having finite bit-rates and affected by data-loss, see also [4]. Despite the above, and as will become apparent from the simulation results included in Section VIII, the SBNC can be expected to give good performance, even when (1) is only an approximate description of the plant behaviour. $\triangle$

VI. Complexity Analysis

As shown in the previous section, good performance can be achieved using SBNC in the presence of an unreliable network, especially if the horizon N and timeout value $\tau_{\max}$ are chosen large enough. This attractive feature comes at a price, namely, the computational cost involved in calculating the control packets $U(k)$ for large N and $\tau_{\max}$. In this section we quantify complexity of SBNC as a function of the design parameters and the network characteristics.

A. Basic Aspects

It follows directly from (9) that implementing SBNC involves calculating

$$\mathcal{C}(k) \triangleq N \sum_{\tau=0}^{\tau_{\max}} |\mathcal{S}_\tau(k)| \tag{27}$$

control values at each time instant k. Thus, the complexity of SBNC is proportional to the horizon length N and to the total number of scenarios considered at each time. As already foreshadowed in the example given in Section IV, $\sum_{\tau=0}^{\tau_{\max}} |\mathcal{S}_\tau(k)|$ depends upon how many valid packets are still in the network, and, therefore, upon the time when the buffer contents $\vec{b}(k-1)$ were generated.

For example, in the case where $U(k-1)$ is received undelayed, only the buffer contents at time $k-1$ are needed to determine $U(k)$. In more general situations, where $\tau(k-1) > 0$ so that $\vec{b}(k-1)$ stems from $U(k-n)$, $n \geq 2$, the sets taken into account to calculate $U(k)$ are:

$$\begin{aligned}\big\{ &(U(k-n+1), \bar{\tau}(k-n+1)), \\ &\quad (U(k-n+2), \bar{\tau}(k-n+2)), \dots \\ &\quad \dots, (U(k-1), \bar{\tau}(k-1)) \big\}_{\tau \in \{1,2,\dots,\tau_{\max}\}}.\end{aligned} \tag{28}$$

In (28), for all $\tau \in \{1, 2, \dots, \tau_{\max}\}$ and for all $\ell \in \{0, 1, \dots, n-1\}$, the values $\bar{\tau}(k-n+\ell) \in \{1, 2, \dots, \tau_{\max}\}$ are all possible combinations such that, for all $\bar{\tau}(k-n+\ell) \in \{1, 2, \dots, \tau_{\max}\}$, it holds that:

$$k - n + \ell + \bar{\tau}(k-n+\ell) \in \{k, k+1, \dots, k+\tau-1\} \tag{29}$$

$$\bar{\tau}(k-n+\ell-j) \leq \bar{\tau}(k-n+\ell), \quad \forall j \geq 1. \tag{30}$$

It is worth noting that (30) results from the precedence rule imposed by the buffer selection logic, see Section III-A and from the fact that the number of pairs in (28) depends upon the timeout value $\tau_{\max}$.

B. Bounds on Complexity

To give further insight into the complexity of SBNC for a given timeout value $\tau_{\max}$, we will denote the total number of scenarios to be considered at each time $k \in \mathbb{N}_0$, when the buffer contents $\vec{b}(k-1)$ were generated at time $k - \eta(k)$, via $\mathcal{I}_{\tau_{\max}}(\eta(k))$, i.e.

$$\mathcal{I}_{\tau_{\max}}(\eta(k)) \triangleq \sum_{\tau=0}^{\tau_{\max}} |\mathcal{S}_\tau(k)|, \quad \eta(k) \in \{1, 2, \cdots, \tau_{\max}+1\}. \tag{31}$$

Clearly, $\mathcal{I}_{\tau_{\max}}(\eta(k))$ is monotonically increasing in $\eta(k)$. Thus, the minimum and maximum complexity of SBNC satisfy:

$$\begin{aligned}\mathcal{C}_{\min} &= \min_{\eta(k) \in \{1, \cdots, \tau_{\max}+1\}} N\mathcal{I}_{\tau_{\max}}(\eta(k)) \qquad (32)\\ &= N\mathcal{I}_{\tau_{\max}}(1), \\ \mathcal{C}_{\max} &= \max_{\eta(k) \in \{1, \cdots, \tau_{\max}+1\}} N\mathcal{I}_{\tau_{\max}}(\eta(k)) \qquad (33)\\ &= N\mathcal{I}_{\tau_{\max}}(\tau_{\max}+1),\end{aligned}$$

where N is the horizon length.

It can be shown (by a somewhat nontrivial inductive argument) that $\mathcal{I}_{\tau_{\max}}(\eta(k))$ can be calculated recursively from:

$$\begin{aligned}\mathcal{I}_{\tau_{\max}}(1) &= \tau_{\max} + 1 \\ \mathcal{I}_{\tau_{\max}}(\tau_{\max}+1) &= \mathcal{I}_{\tau_{\max}}(\tau_{\max}) + \mathcal{I}_{\tau_{\max}-1}(\tau_{\max}) \\ \mathcal{I}_{\tau_{\max}}(\eta(k)) &= \mathcal{I}_{\tau_{\max}}(\eta(k)-1) + \mathcal{I}_{\tau_{\max}-1}(\eta(k)), \\ &\qquad \eta(k) \in \{2, 3, \cdots, \tau_{\max}\}.\end{aligned} \tag{34}$$

Table I gives some numerical values for $\mathcal{I}_{\tau_{\max}}(\eta(k))$.

The following result establishes upper bounds for $\mathcal{I}_{\tau_{\max}}(\eta(k))$ and for the worst case complexity $\mathcal{C}_{\max}$:

Lemma 1 (Upper bounds): The total number of scenarios $\mathcal{I}_{\tau_{\max}}(\eta(k))$ and the maximal complexity of SBNC satisfy the

TABLE I
NUMBER OF SCENARIOS $\mathcal{I}_{\tau_{\max}}(\eta(k))$, SEE (31), FOR SOME VALUES OF $\eta(k)$ AND $\tau_{\max}$.

	$\eta(k)$								
$\tau_{\max}$	1	2	3	4	5	6	7	8	9
0	1	-	-	-	-	-	-	-	-
1	2	3	-	-	-	-	-	-	-
2	3	6	9	-	-	-	-	-	-
3	4	10	19	28	-	-	-	-	-
4	5	15	34	62	90	-	-	-	-
5	6	21	55	117	207	297	-	-	-
6	7	28	83	200	407	704	1001	-	-
7	8	36	119	319	726	1430	2431	3432	-
8	9	45	164	483	1209	2639	5070	8502	11934

following upper bounds:

$$\mathcal{I}_{\tau_{\max}}(\eta(k)) \leq \sum_{\ell=0}^{\eta(k)-1} \binom{\tau_{\max}+1}{\tau_{\max}-\ell}\binom{\eta(k)-1}{\ell} \tag{35}$$

$$\mathcal{C}_{\max} \leq N \sum_{\ell=0}^{\tau_{\max}} \binom{\tau_{\max}+1}{\tau_{\max}-\ell}\binom{\tau_{\max}}{\ell} = N\binom{2\tau_{\max}+1}{\tau_{\max}}. \tag{36}$$

Proof: The proof is included in Appendix B. ■

C. Average Complexity

To characterize the average complexity of SBNC for a given timeout value $\tau_{\max}$, we denote the probability that $\vec{b}(k-1)$ was generated at time $k-\eta$, $\eta \in \{1, 2, \cdots, \tau_{\max}+1\}$ by $\mathbb{P}(\eta)$. Clearly, for $n = 1$ we have $\mathbb{P}(1) = \mathcal{P}\{\tau(k-1) = 0\}$. For larger values of n it holds that

$$\mathbb{P}(2) = \mathcal{P}\big\{\tau(k-2) \in \{0,1\} \cap \tau(k-1) \geq 1\big\},$$
$$\mathbb{P}(3) = \mathcal{P}\big\{\tau(k-3) \in \{0,1,2\} \cap \tau(k-2) \geq 2 \cap \tau(k-1) \geq 1\big\}$$

and, more generally,

$$\mathbb{P}(\eta) \triangleq \mathcal{P}\Big\{\tau(k-\eta) \in \{0, 1, \cdots, \eta-1\} \bigwedge_{\ell=1}^{\eta-1} \tau(k-\ell) \geq \ell\Big\}. \tag{37}$$

Given the above, the average (or expected) complexity of SBNC is given by

$$\mathcal{C}_{\text{avg}}(k) = N \sum_{\eta=1}^{\tau_{\max}+1} \mathbb{P}(\eta)\mathcal{I}_{\tau_{\max}}(\eta). \tag{38}$$

The following upper bound for $\mathcal{C}_{\text{avg}}(k)$ then follows directly from Lemma 1:

$$\mathcal{C}_{\text{avg}}(k) \leq N \sum_{\eta=1}^{\tau_{\max}+1}\sum_{\ell=0}^{\eta-1} \binom{\tau_{\max}+1}{\tau_{\max}-\ell}\binom{\eta-1}{\ell}\mathbb{P}(\eta).$$

Note that, if $\{\tau(k)\}_{k\in\mathbb{N}_0}$ is a sequence of i.i.d. random variables, each one having cumulative probability function $F(\cdot)$, then the probabilities $\mathbb{P}(\eta)$ in (37) are simply given by

$$\mathbb{P}(\eta) = F(\eta-1)\prod_{\ell=1}^{\eta-1}(1-F(\ell-1)).$$

VII. ALTERNATIVE STRATEGIES

SBNC examines all possible plant input scenarios in (7). This feature, in conjunction with the use of appropriate closed loop policies $\kappa_\ell(\cdot)$, allows one to derive the performance results presented in Section V. As seen in Section VI, the complexity of SBNC will be significant for large values of timeout, $\tau_{\max}$, and horizon length, N. Although often moderate values of $\tau_{\max}$ and N will give good results, in some applications, simpler strategies may be preferable. We will next briefly outline two such alternatives.

A. Central SBNC

An extremely simple variant of SBNC can be formed by calculating $U(k)$ based only on the input scenario resulting from the (known) buffer state $\vec{b}(k-1)$, without accounting for any packets (other than $U(k)$) which could arrive at times $k, k+1, \ldots, k+\tau(k)-1$.

More precisely, at every time instant k, given the state $x(k) = x_0$, the buffer sequence

$$\vec{b}(k-1) = \{u_{-1}, u_0, u_1, \ldots, u_{N-2}\},$$

and with $N \geq \tau_{\max}+2$, the control strategy sends only:

$$U(k) = \Big\{\vec{u}(k;0,\{\}), \vec{u}(k;1,\{u_0\}), \vec{u}(k;1,\{u_0,u_2\}), \ldots \\ \ldots, \vec{u}(k;\tau_{\max},\{u_0,\ldots,u_{\tau_{\max}}\})\Big\}, \tag{39}$$

where we have used the notation of Section III-C.

If the prediction horizon is chosen such that $N < \tau_{\max}+2$, then one can simply set:

$$u_{N-2+j} = u_{N-2}, \quad \forall j \in \{0, 1, \ldots, \tau_{\max}-N+2\}.$$

Note that in (39), there are only $\tau_{\max}+1$ sequences of length N to be calculated.

We will call this strategy *Central SBNC*. It requires only very limited computational effort. Unfortunately, the performance and stability results for SBNC obtained in Section V do not, in general apply to Central SBNC. Nevertheless, it can be expected that, for fairly reliable networks and small $\tau_{\max}$, Central SBNC will give good results. In particular, for erasure channels (where $\tau_{\max} = 0$), SBNC reduces to Central SBNC.

B. Handshaking SBNC

If reliable acknowledgment procedures are available, then one can modify the transmission policy of Central SBNC described above such that the controller only sends packets $U(k)$, once it knows whether the last packet sent has been received or has been dropped. This reduces network usage further and impedes the possibility that more than one packet is on the network at the same time. Consequently, any uncertainty on future plant inputs is eliminated. We call the associated strategy *Handshaking SBNC*.

It is easy to see that, with this transmission procedure and provided $U(k)$ is designed as in (39), performance and stability results similar to those of SBNC presented in Section V can be derived. Note that, in general, larger horizon lengths N will need to be used.

Nevertheless, it is worth emphasizing that, between successful transmission instants, the plant operates in open loop. Thus, in the presence of disturbances, etc., this modification of SBNC may give poor performance, due to the reduced network access.

VIII. Simulation Studies

To illustrate the properties of SBNC and its reduced complexity variants, we will next document simulation studies for non-idealized cases, i.e., where (1) does not hold due to the presence of disturbances and measurement noise. For future reference, we present in Fig. 7 the two specific i.i.d. network delay distributions that will be considered. In the sequel, we will refer to these two situations as *Network A* and *Network B*, respectively. The prediction horizon N and timeout value $\tau_{\max}$ are used as SBNC design parameters. Note that in the case of Network A, if one sets $\tau_{\max} \leq 4$, then most packets will be effectively lost. For Network B, if $\tau_{\max}$ is chosen to be larger than 3, then almost no packets will be considered as lost.

For simplicity, we take κ^f such that $\kappa^f(\ell) = u(\ell - 1)$, i.e., empty buffers are dealt with by keeping plant inputs at their current value.

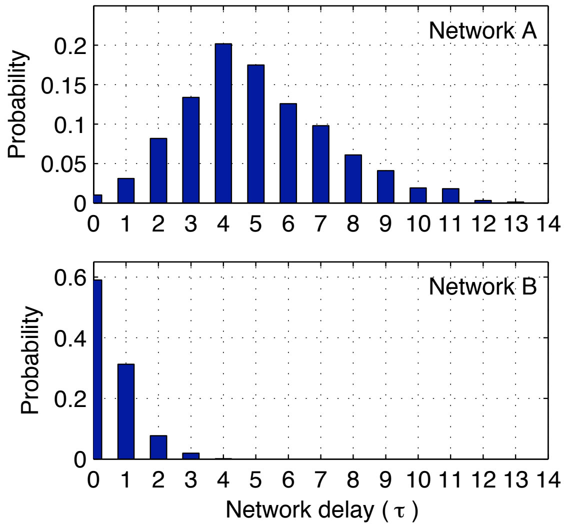

Fig. 7. Delay distributions of the networks examined.

A. Nonlinear Plant Model

Consider the following stable nonlinear plant having scalar input (adapted from Example 19.3 in [26])

$$\begin{aligned} x_1(\ell+1) &= x_1(\ell) + 0.01\big(x_2(\ell) + x_2(\ell)^3\big) \\ x_2(\ell+1) &= x_2(\ell) + 0.01\big(-2x_1(\ell) - 3x_2(\ell) \\ &\qquad\qquad + u(\ell)\big(1 + 0.1x_1(\ell)^2\big)\big) \\ y(\ell) &= x_1(\ell) + d(\ell), \end{aligned}$$

where $x(\ell) = [x_1(\ell)\ x_2(\ell)]^T$ is the plant state at time ℓ, $\{u(\ell)\}$ is the plant input, $\{y(\ell)\}$ is the plant output, and $\{d(\ell)\}$ is a piecewise constant output disturbance having infrequent steps at random times. Both the plant state and disturbance measurements are affected by Gaussian white noise of variance $\sigma_d^2 = 0.01$ and $\sigma_x^2 = 0.04 I_2$, respectively.

The control objective is to achieve reference tracking for the plant output. Whilst future reference values are known to the controller, future disturbances are not. For simplicity, the latter are assumed constant for control calculations.

The underlying controller is given by (see [26]):

$$\kappa_\ell(x(\ell)) = \frac{r_y(\ell) - d(\ell) - B(x(\ell))}{A(x(\ell))},$$

where $\{r_y(\ell)\}$ is the reference for $y(\ell)$,

$$\begin{aligned} A(x) &= (1 + 3x_2^2)(1 + 0.1x_1^2)/9, \\ B(x) &= -(2x_1 + 6x_2^2 x_1 + 9x_2^3 - 4x_2 - 4x_2^3 - 9x_1)/9. \end{aligned}$$

To investigate the effect on performance of the SBNC design parameters $\tau_{\max}$ and N, we consider Network A, see Fig. 7.[10] Fig. 8 shows typical plant output trajectories for step-like reference and disturbances given by:

$$r_y(\ell) = \mu(\ell) + \mu(\ell - 1800), \quad d(\ell) = \mu(\ell - 3600),$$

where $\mu(k)$ denotes a unit step at time k. For each pair $(N, \tau_{\max})$ we also calculated the equivalent dropout probability $p_{eq}(\ell)$ (see Theorem 1) and the average complexity C_a (see (38)).

As expected, closed loop performance improves when N and τ_{max} are increased. However, it is also appreciated that there are cases where increasing N has no obvious effect on SBNC performance. This is easily explained if one notes that, as suggested by the results of Theorem 1, the equivalent dropout probability p_{eq} is the key parameter determining SBNC performance. Indeed, cases with very disparate values of N or τ_{max} can exhibit similar performance if the corresponding equivalent dropout probabilities are similar.[11]

It is also informative to compare the probability of a packet being effectively lost due to the choice of $\tau_{\max}$, and the equivalent dropout probability p_{eq} (which depends on the interplay between N and $\tau_{\max}$). It follows from Fig. 7 that if $\tau_{\max} \in \{1, 2, 7\}$, then $\mathcal{P}\{\tau(k) > \tau_{\max}\}$ equals $0.959, 0.877$ and 0.143, respectively. Our results show that by a proper choice of N one can decrease this probability dramatically. It must be noted, however, that if one chooses N too small (as compared with $\tau_{\max}$), then p_{eq} could be larger than $\mathcal{P}\{\tau(k) > \tau_{\max}\}$ (see the case $(N, \tau_{\max}) = (2, 7)$). This is due to the fact that, in those cases, the buffer will run out of data quite often.

It is also worth mentioning that, for this problem, using $\tau_{\max} = 0$, does not stabilize the loop for *any* value of N. In addition, the simple policy that regards all delayed data as lost and only sends the current plant input, i.e., SBNC with $N = 1$ and $\tau_{\max} = 0$, gives unstable behaviour also. This strongly supports the use of the more complex SBNC scheme advocated here.

[10]As mentioned in Section V-A, SBNC does not use the delay profiles.

[11]Consider, for example, the cases $(N, \tau_{\max}) = (15, 1)$ and $(N, \tau_{\max}) = (2, 2)$ or, more dramatically, $(N, \tau_{\max}) = (6, 7)$ and $(N, \tau_{\max}) = (15, 7)$.

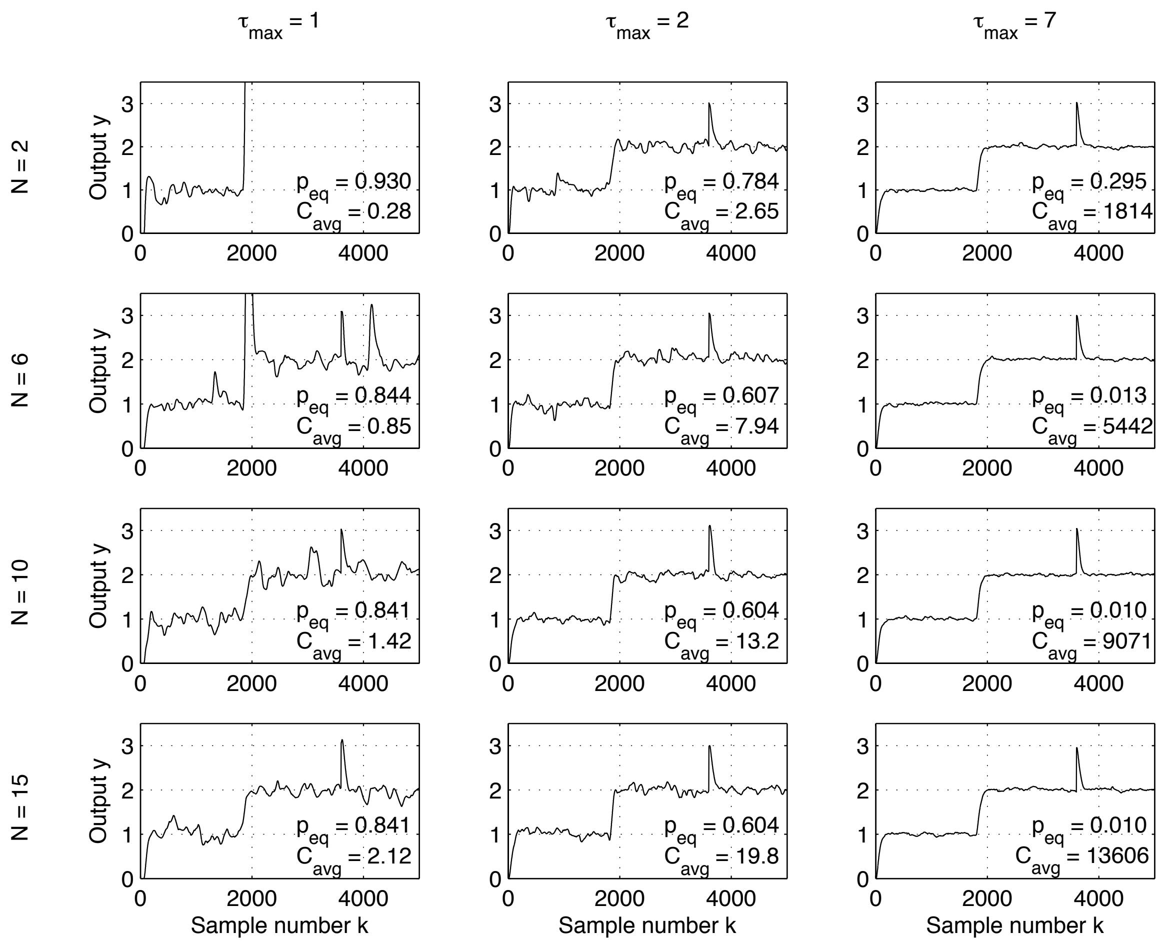

Fig. 8. SBNC Closed loop performance as a function of N and $\tau_{\max}$.

B. Output Feedback

To examine the alternative strategies described in Section VII, we next consider an unstable linear plant given by[12]

$$x(\ell+1) = \begin{bmatrix} 1.184 & 0.5362 \\ -0.01543 & -0.7164 \end{bmatrix} x(\ell) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} (u(\ell)+d(\ell))$$
$$y(\ell) = \begin{bmatrix} 1 & 1 \end{bmatrix} x(\ell).$$

As in the previous situation, we focus on reference tracking and disturbance rejection at the plant output and use

$$r(\ell) = \mu(\ell) - 2\mu(\ell-20) + 3\mu(\ell-40),$$
$$d(\ell) = -0.5\mu(\ell) + \mu(\ell-60).$$

The controller does not have access to the plant state or disturbance measurements. Only noisy plant output measurements are available. The measurement noise is taken as white and Gaussian of variance 10^{-3}. We design SBNC based upon using a standard full order observer for an augmented plant model that includes a disturbance description, and static estimated state feedback combined with a reference feedforward gain; see, e.g., [26].[13]

We compare SBNC, Central SBNC and Handshaking SBNC. Fig. 9 illustrates our results for Network B. In each of the cases shown, we have chosen parameters N and $\tau_{\max}$ consistent with keeping computational burden low, whilst maintaining good performance. It can be clearly seen that, for the fairly reliable network considered here, and for these particular tuning parameters, all three methods give good performance. Under these conditions, the alternative strategies proposed in Section VII are certainly appealing.

Interestingly, in the case of Central SBNC, increasing $\tau_{\max}$ can have a negative impact on loop performance. This is due to the fact that the Central SBNC uses only the current buffer content to predict plant states at future possible arrival time instants. Due to disturbance changes and arrival of other packets, the actual plant state may differ substantially from the predicted one. Since the plant is unstable, this mismatch often has negative consequences. This effect does not occur in the SBNC and the handshaking SBNC, since more precise predictions are used.

We next consider Network A. Here, one should bear in mind that, due to network effects, the (open loop unstable) plant is unavoidably operated in open-loop over many successive time instants. Fig. 10 shows typical output trajectories. In this case, the SBNC performs well. However, the Central SBNC and the Handshaking SBNC are incapable of providing acceptable performance.

It is interesting to note that, in accord with observations

[12]The dynamics matrix was chosen randomly.

[13]We assume no data loss in the estimation-link.

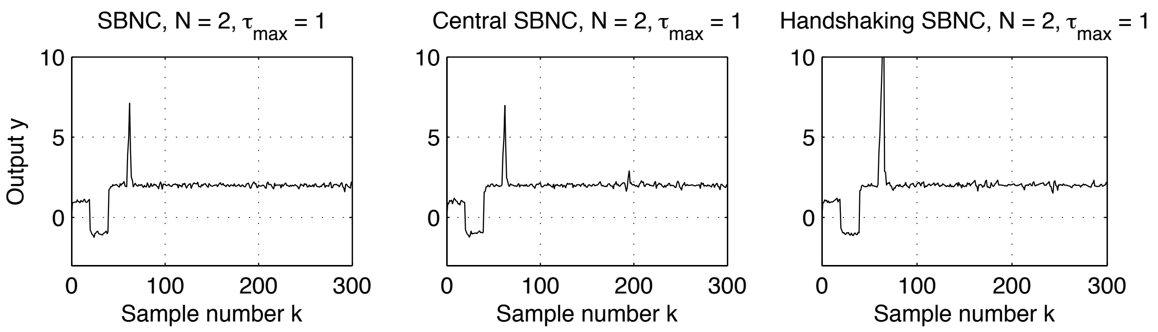

Fig. 9. Comparison between different control strategies for the Network B.

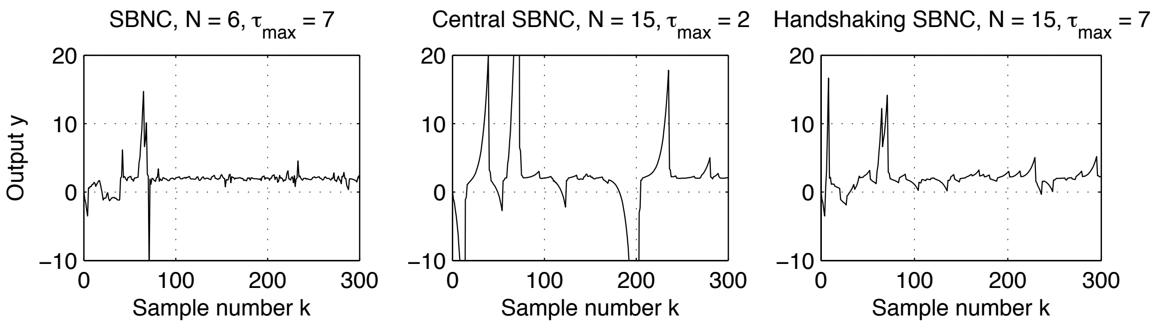

Fig. 10. Comparison between different control strategies for Network A.

made above, increasing the value of $\tau_{\max}$ in the Central SBNC case leads to unacceptable performance and even instability. This can be explained as before. Also, higher values for N and $\tau_{\max}$ lead, as expected, to better performance in the SBNC and Handshaking SBNC cases.

As in the case considered in Section VIII-A, if $\tau_{\max} = 0$, then the closed loop becomes unstable for the Network A and any prediction horizon N.

IX. Conclusions

In the present work we have presented a strategy for closed loop control of non-linear plants over unreliable networks having data-dropouts and variable time-delays. Our proposal uses transmission of packets containing various plant input sequences, which are calculated by considering the various scenarios associated with possible transmission outcomes. The method can be used to embellish any state-feedback control law.

We have shown that, under nominal conditions, the resultant NCS is equivalent to an NCS which operates over a network without time-delays and with a small data-dropout probability. In some situations, closed loop stability and performance, in the networked case, are directly inherited from that which would be achieved if the network were transparent.

Design parameters, namely prediction horizon N and time-out value $\tau_{\max}$, allow one to trade-off control, computation and communication aspects. In some cases, simplified strategies lead to acceptable results. Future work could include a further study of the simplified strategies described in Section VII and the treatment of stochastic disturbances and modeling errors.

Appendix

A. Proof of Theorem 1

Since (1) holds, we have that

$$x(n_i + \ell; x(k_i), \vec{s}_\tau^{(j^*)}(k_i)) = x(n_i + \ell),$$
$$\forall \ell \in \{0, 1, \ldots, \min\{N-1, n_{i+1} - n_i - 1\}\},\ \forall i \in \mathbb{N}.$$

It then follows from (14), (11) and (13) that $\forall i \in \mathbb{N}$ and for all $\ell \in \{0, 1, \ldots, \min\{N-1, n_{i+1} - n_i - 1\}\}$, we have:

$$u(n_i + \ell) = \kappa_{n_i+\ell}\left(x(n_i + \ell)\right).$$

Therefore, provided valid received data is in the buffer at time ℓ, we have $u(\ell) = \kappa_\ell(x(\ell))$. On the other hand, if no valid data is present at time ℓ, i.e., if the buffer runs out of data or no data has ever been received, then, by definition of SBNC, $u(\ell) = \kappa^f(\ell)$. Expression (21) then follows on noting that, for $0 \leq \ell \leq n_1 - 1$, no data arrives, and that the first time instant at which the buffer may run out of data is $\ell = n_1 + N$.

To prove (22), we first observe that $1 - p_{eq}(\ell) = \sum_{j=0}^{N-1} P_j$, where

$$P_j \triangleq \mathcal{P}\left\{\text{data which is still valid at instant } \ell \text{ arrived at } k-j\right\}$$

We have

$$P_0 = \mathcal{P}\big\{\tau(\ell)=0\big\} + \mathcal{P}\big\{\tau(\ell)>0 \wedge \tau(\ell-1)=1\big\}$$
$$+ \mathcal{P}\big\{\tau(\ell)>0 \wedge \tau(\ell-1)>1 \wedge \cdots$$
$$\cdots \wedge \tau(\ell-\tau_{\max}+1) > \tau_{\max}-1 \wedge \tau(k-\tau_{\max}) = \tau_{\max}\big\},$$
$$P_1 = \mathcal{P}\big\{\tau(\ell)>0 \wedge \tau(\ell-1)=0\big\}$$
$$+ \mathcal{P}\big\{\tau(\ell)>0 \wedge \tau(\ell-1)>1 \wedge \tau(\ell-2)=1\big\}$$
$$+ \mathcal{P}\big\{\tau(\ell)>0 \wedge \tau(\ell-1)>1 \wedge \cdots$$
$$\cdots \wedge \tau(k-\tau_{\max}) > \tau_{\max} \wedge \tau(k-\tau_{\max}-1) = \tau_{\max}\big\},$$

and, more generally, $P_j = \sum_{i=0}^{\tau_{\max}} P_{ji}$, where:

$$P_{ji} \triangleq \mathcal{P}\big\{\tau(\ell-i-j)=i \wedge \tau(\ell)>0 \wedge \tau(\ell-1)>1 \wedge \cdots$$
$$\cdots \wedge \tau(\ell-(j+i-1)) > j+i-1\big\}.$$

The result follows.

B. *Proof of Lemma 1*

To compute the bound (35) we will allow *all* packets $\{U(k-\tau_{\max}), \cdots U(k-1)\}$ to be considered valid if they arrive at any of the instants $\{k, k+1, \ldots, k+\tau_{\max}\}$.[14]

Pick any n, i.e., assume that the last valid packet was $U(k-n)$. Define ℓ as the number of packets, besides $U(k)$, that could arrive at the instants $\{k, k+1, \ldots, k+\tau_{\max}\}$. Clearly, $\ell \in \{0, 1, \cdots, n-1\}$ and, for every ℓ, the number of different combinations that can be made out of the $n-1$ packets that may arrive equals $\begin{pmatrix} n-1 \\ \ell \end{pmatrix}$.

On the other hand, there exist $\tau_{\max}+1$ instants at which data may arrive. Noting that the number of ways in which a given set of ℓ packets (besides $U(k)$) may arrive in $\{k, k+1, \ldots, k+\tau_{\max}\}$ equals the number of ways in which $\tau_{\max}-\ell$ non-arrivals can be chosen out of the $\tau_{max}+1$ possibilities, we conclude that, for every ℓ, the number of scenarios is upper bounded by $\begin{pmatrix} \tau_{\max}+1 \\ \tau_{\max}-\ell \end{pmatrix}\begin{pmatrix} n-1 \\ \ell \end{pmatrix}$. Summing for all values of ℓ gives (35). Equation (36) follows directly from (33) and Vandermonde's convolution formula [41].

Acknowledgment

The authors would like to thank Jan Østergaard and the anonymous reviewers for valuable suggestions on this work.

References

[1] J. Baillieul and P. Antsaklis, "Control and communication challenges in networked real-time systems," *Proc. IEEE*, vol. 95, pp. 9–27, Jan. 2007.

[2] J. R. Moyne and D. M. Tilbury, "The emergence of industrial control networks for manufacturing control, diagnostics, and safety data," *Proc. IEEE*, vol. 95, pp. 29–47, Jan. 2007.

[3] D. Hristu-Varsakelis and W. S. Levine, eds., *Handbook of Networked and Embedded Systems*. Boston, MA: Birkhäuser, 2005.

[4] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans, "Feedback control under data rate constraints: An overview," *Proc. IEEE*, vol. 95, pp. 108–137, Jan. 2007.

[5] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. IEEE*, vol. 1, pp. 138–162, Jan. 2007.

[14] In reality, when using SBNC, $U(k-n)$ is considered valid only if it arrives before or at instant $k-n+\tau_{\max}$.

[6] G. C. Goodwin, H. Haimovich, D. E. Quevedo, and J. S. Welsh, "A moving horizon approach to networked control system design," *IEEE Trans. Automat. Contr.*, vol. 49, pp. 1427–1445, Sept. 2004.

[7] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proc. IEEE*, vol. 95, pp. 163–187, Jan. 2007.

[8] P. Seiler and R. Sengupta, "An $\mathcal{H}_\infty$ approach to networked control," *IEEE Trans. Automat. Contr.*, vol. 50, pp. 356–364, Mar. 2005.

[9] V. Gupta, B. Hassibi, and R. M. Murray, "Optimal LQR control across packet-dropping links," *Syst. & Contr. Lett.*, vol. 56, pp. 439–446, 2007.

[10] O. C. Imer, S. Yüksel, and T. Başar, "Optimal control of LTI systems over unreliable communication links," *Automatica*, vol. 42, pp. 1429–1439, Sept. 2006.

[11] P. V. Zhivoglyadov and R. H. Middleton, "Networked control design for linear systems," *Automatica*, vol. 39, pp. 743–750, 2003.

[12] G. C. Goodwin, D. E. Quevedo, and E. I. Silva, "Architectures and coder design for networked control systems," *Automatica*, vol. 44, pp. 248–257, Jan. 2008.

[13] A. Bemporad, "Predictive control of teleoperated constrained systems with unbounded communication delays," in *Proc. IEEE Conf. Decis. Contr.*, (Tampa, Florida), pp. 2133–2138, 1998.

[14] G. P. Liu, J. X. Mu, D. Rees, and S. C. Chai, "Design and stability analysis of networked control systems with random communication time delay using the modified MPC," *Int. J. Contr.*, vol. 79, pp. 288–297, Apr. 2006.

[15] A. Casavola, E. Mosca, and M. Papini, "Predictive teleoperation of constrained dynamic systems via internet-like channels," *IEEE Trans. Contr. Syst. Technol.*, vol. 14, pp. 681–694, July 2006.

[16] P. L. Tang and C. W. de Silva, "Compensation for transmission delays in an Ethernet-based control network using variable-horizon predictive control," *IEEE Trans. Contr. Syst. Technol.*, vol. 14, pp. 707–718, July 2006.

[17] W.-J. Kim, K. Ji, and A. Ambike, "Real-time operating environment for networked control systems," *IEEE Trans. Automat. Sc. Eng.*, vol. 3, pp. 287–296, July 2006.

[18] G. P. 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," *IEEE Trans. Ind. Electron.*, vol. 54, pp. 1282–1297, June 2007.

[19] V. Gupta, B. Sinopoli, S. Adlakha, and A. Goldsmith, "Receding horizon networked control," in *Proc. Allerton Conf. Communications, Control, and Computing*, (Monticello, IL), 2006.

[20] D. E. Quevedo, E. I. Silva, and G. C. Goodwin, "Packetized predictive control over erasure channels," in *Proc. Amer. Contr. Conf.*, (New York, N.Y.), 2007.

[21] Y. Bar-Shalom and E. Tse, "Dual effect, certainty equivalence, and separation in stochastic control," *IEEE Transactions on Automatic Control*, vol. 19, pp. 494– 500, Oct. 1974.

[22] F. V. Louveaux and M. H. van der Vlerk, "Stochastic programming with simple integer recourse," *Journal Mathematical Programming*, vol. 61, pp. 301–325, Aug. 1993.

[23] J. Higle and S. Sen, "Statistical approximations for recourse constrained stochastic programs," *Journal Annals of Operations Research*, vol. 56, pp. 157–175, Dec. 1995.

[24] J. Dupacova, G. Consigli, and S. W. Wallace, "Scenarios for multistage stochastic programs," *Annals of Operations Research*, vol. 100, 2000.

[25] G. C. Pflug, *Optimization of Stochastic Models: The Interface Between Simulation and Optimization*. Boston: Kluwer Academic Publishers, 1996.

[26] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control System Design*. Prentice-Hall, 2001.

[27] K. Zhou, J. Doyle, and K. Glover, *Robust and optimal control*. Upper Saddle River, New Jersey: Prentice-Hall, 1996.

[28] A. Isidori, *Nonlinear Control Systems*. London, U.K.: Springer-Verlag, 1995.

[29] E. F. Camacho and C. Bordons, *Model Predictive Control*. New York, N.Y.: Springer-Verlag, 1999.

[30] G. C. Goodwin, M. M. Serón, and J. A. De Doná, *Constrained Control & Estimation – An Optimization Perspective*. London: Springer-Verlag, 2005.

[31] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná, "Finite constraint set receding horizon quadratic control," *Int. J. Robust Nonlin. Contr.*, vol. 14, pp. 355–377, Mar. 2004.

[32] V. K. Goyal, "High-rate transform coding: How high is high, and Does it matter?," in *Proc. IEEE Int. Symp. Inform. Theory*, (Sorrento, Italy), p. 207, 2000.

[33] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uiterwaal, and P. Van Mieghem, “Analysis of end-to-end delay measurements in the internet,” in *Proc. Passive and Active Measurement Workshop-PAM*, (Fort Collins, Colorado), 2002.

[34] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic, 1992.

[35] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, “Constrained model predictive control: Optimality and stability,” *Automatica*, vol. 36, pp. 789–814, June 2000.

[36] Q. Ling and M. D. Lemmon, “Power spectral analysis of networked control systems with data dropouts,” *IEEE Trans. Automat. Contr.*, vol. 49, pp. 955–960, June 2004.

[37] J. Nilsson, B. Bernhardsson, and B. Wittenmark, “Stochastic analysis and control of real-time systems with random time delays,” *Automatica*, vol. 34, no. 1, pp. 57–64, 1998.

[38] H. Shousong and Z. Qixin, “Stochastic optimal control and analysis of stability of networked control systems with long delay,” *Automatica*, vol. 39, pp. 1877–1884, 2003.

[39] A. S. Matveev and A. V. Savkin, “Comments on ”control over noisy channels” and relevant negative results,” *IEEE Trans. Automat. Contr.*, vol. 50, pp. 2105–2110, Dec. 2005.

[40] A. Sahai, *Anytime Information Theory*. PhD thesis, MIT, Cambridge, MA, 2001.

[41] S. Roman, *The Umbral Calculus*. New York, N.Y.: Academic Press, 1984.

PLACE
PHOTO
HERE

Daniel E. Quevedo received Ingeniero Civil Electrónico and Magister en Ingeniería Electrónica degrees from the Universidad Técnica Federico Santa María, Valparaíso, Chile in 2000. In 2005, he received the Ph.D. degree from The University of Newcastle, Australia, where he currently holds a research academic position.

He has lectured at the Universidad Técnica Federico Santa María and The University of Newcastle. He also has working experience at the VEW Energie AG, Dortmund, Germany and at the Cerro Tololo Inter-American Observatory, La Serena, Chile. His research interests cover several areas of automatic control, signal processing, and communications.

Daniel was supported by a full scholarship from the alumni association during his time at the Universidad Técnica Federico Santa María and received several university-wide prizes upon graduating. He received the IEEE Conference on Decision and Control Best Student Paper Award in 2003 and was also a finalist in 2002.

PLACE
PHOTO
HERE

Eduardo I. Silva was born in Valdivia, Chile, in 1979. He received the Ingeniero Civil Electrónico and Magister en Ingeniería Electrónica degrees from the Universidad Técnica Federico Santa María (UTFSM), Valparaso, Chile in 2004.

He worked as a Research Assistant at the UTFSM during 2005. He is currently working toward the Ph.D at the School of Electrical Engineering and Computer Science, The University of Newcastle, Australia. His research interests include multivariate control systems, performance limitations, decentralized control, networked control and signal processing.

PLACE
PHOTO
HERE

Graham C. Goodwin obtained a B.Sc (Physics), B.E (Electrical Engineering), and Ph.D from the University of New South Wales. He is currently Professor of Electrical Engineering at the University of Newcastle, Australia and holds Honorary Doctorates from Lund Institute of Technology, Sweden and the Technion, Israel.

He is the co-author of eight books, four edited volumes, and many technical papers. Graham is the recipient of Control Systems Society 1999 Hendrik Bode Lecture Prize, a Best Paper award by IEEE Trans. Automatic Control, a Best Paper award by Asian Journal of Control, and 2 Best Engineering Text Book awards from the International Federation of Automatic Control. He is a Fellow of the IEEE; an Honorary Fellow of Institute of Engineers, Australia; a Fellow of the Australian Academy of Science; a Fellow of the Australian Academy of Technology, Science and Engineering; a Member of the International Statistical Institute; a Fellow of the Royal Society, London and a Foreign Member of the Royal Swedish Academy of Sciences.