# Discrete and Hybrid Stochastic State Estimation Algorithms for Networked Control Systems

S. Di Cairano[1], K.H. Johansson[2], A. Bemporad[1], and R.M. Murray[3,⋆]

[1] Dip. Ingegneria dell'Informazione, Università di Siena, Italy
{dicairano,bemporad}@dii.unisi.it
[2] School of Electrical Engineering, Royal Institute of Technology, Sweden
kallej@ee.kth.se
[3] Division of Engineering and Applied Science, California Institute of Technology
murray@caltech.edu

**Abstract.** Networked control systems enable for flexible systems operation and reduce cost of installation and maintenance, potentially at the price of increasing the uncertainty due to information exchange over the network. We focus on the problem of information loss in terms of packet drops, which are modelled as stochastic events that depend on the current state of the network. To design reliable control systems the state of the network must be estimated online, together with the state of the controlled process. This paper proposes various approaches to discrete and hybrid stochastic estimation of network and process states, where the network is modelled as a Markov chain and the packet drop probability depends on the states of the Markov chain. The proposed techniques are evaluated on simulations and experimental data.

## 1 Introduction

Advances in network technology increase the flexibility of modern control systems. Networked control systems[1], in which the controller is remotely located with respect to the plant, are becoming more and more tested and used in industrial applications [2,3]. The advantages of such architectures are in the

⋆ The work by S. Di Cairano and A. Bemporad was supported by the European Commission through the HYCON Network of Excellence (contract FP6-IST-511368), and by the Italian Ministry for University and Research (MIUR) under project "Advanced control methodologies for hybrid dynamical systems" (PRIN'2005). The work by K. H. Johansson was supported by the European Commission through the Network of Excellence HYCON and the Integrated Project SOCRADES, the KTH ACCESS Linnaeus Center, the Swedish Research Council and the Swedish Foundation for Strategic Research. The work by S. Di Cairano and R.M. Murray was partially supported by Boeing through the project Model-Based Design and Qualification of Complex Systems.

M. Egerstedt and B. Mishra (Eds.): HSCC 2008, LNCS 4981, pp. 144–157, 2008.

costs, since a single shared network is less expensive than many point-to-point connections, in the flexibility, since networks are usually capable of automatic reconfiguration, and in the control system maintenance, since the control unit can be deployed far from the plant, that often operates in extreme environmental conditions. Such advantages are further increased when one considers wireless networked control systems, since the absence of wiring further reduces maintenance and deployment costs and increases flexibility. However, when designing a networked control system new issues must be taken into account. The communication network introduces information losses, bandwidth limitations and time-varying delays [4]. These problems are more important in wireless networks [5,6] than in wired networks, as the radio channel performance is affected by many environmental factors and changes rapidly.

When controlling a system over a network, the performance of the network may drastically affect the performance of the system. Several studies have been developed for the particular case in which the network behavior is time invariant, for both, controller and estimator design (see [1,4] for extensive surveys). However, the network characteristics often changes dynamically depending on several factors including network load, number of active users, and environmental conditions [7]. An estimator for a network with piecewise constant statistical properties has been proposed in [8], while a Markov chain model of the network channel has been used for instance in [9,10,11].

When the network is time varying, the overall system state is constituted by the states of the process, of the controller, and of the network. Henceforth, when performing the estimation, it is important to estimate also the current network state. As an example, consider a control system that is composed of a local controller, enforcing stability, and of a remote optimal planner, that communicates via a wireless network, see [12] for a particular example. When the network is reliable, aggressive plans leading to high tracking performances can be safely executed. On the other hand, when the network is unreliable, more cautious plans should be chosen. As another example, consider Alice, the autonomous vehicle of Team Caltech in the 2007 Urban Challenge. Alice's architecture is based on a complex network of sensors, actuators, and computational units. The estimation of the network characteristics can help in revealing whether missing sensor data, such as the localization system data, are not received due to network overload or because the hardware failed, so that the appropriate fault-handling actions can be taken.

This paper analyzes the problem of jointly estimating the network state and the process state under various conditions. Since the physical network models are in general too complicated for control design purposes, we use an abstract model which is simple enough to be analyzable, and still capture the main characteristics of the network phenomena [6]. We focus on networks affected by packet drops, where some of the packets never reach the receiver. The packet drop is an abstract phenomenon representing the loss of information, either physical or logical, in the transmission of a data packet from a sender to a receiver. Many network protocols already provide reliable message delivery through services such

as ARQ (automatic repeat request), but these operations introduce delays, and when used in real-time control systems delayed packets can have major influence on closed-loop performance.

We model a discrete-valued network state. Hence, the joint estimation of the continuous-valued process state and of the network state reduces to the estimation of a particular type of hybrid system. We also aim at understanding when the performance of the process state estimation does not affect the network state estimation. In Section 2 we propose the system model and we formulate the estimation problems. In Section 3 we consider the cases in which the network state estimate can be separated from the process estimate. In Section 4 we consider the joint network state and process state estimation. These problems are first analyzed for a networked control system where the network is connecting the controller to the process, and in Section 5 we extend to the different cases involving an additional network between the sensor and the estimator. Simulations of the estimation algorithms are presented in Section 6 together with some experimental results, and the conclusions are summarized in Section 7.

## 2 Modelling and Problem Formulation

Consider the networked system shown in Figure 1. A discrete-time signal $u(\cdot)$ is transmitted through network $\mathcal{N}_1$ and the received signal $\tilde{u}(\cdot)$ is the input to a dynamical process $\Sigma(x, \tilde{u}, y)$, where $x$ is the process state. The output (or measurement) $y(\cdot)$ is transmitted through network $\mathcal{N}_2$, and the received signal $\tilde{y}(\cdot)$ reaches an estimator $\Theta(u, \tilde{y})$ that knows the signal $u(\cdot)$. The estimator solves a hybrid estimation problem: it provides an estimate $\hat{x}$ of the continuous process state $x$, and, at the same time, it estimates the discrete states of the networks $\mathcal{N}_1$ and $\mathcal{N}_2$, $N_1$ and $N_2$, respectively. Note that the signals $u$, $\tilde{y}$ available to the estimator may be different from the actual system input $\tilde{u}$ and output $y$. We call $\mathcal{N}_1$ the *actuation network* and $\mathcal{N}_2$ the *sensing network*, $\tilde{u}(\cdot)$ and $\tilde{y}(\cdot)$ the network filtered input and measurement, and the packets containing $u$ and $y$ are the command packet and the measurement packet, respectively.

Let $\Sigma(x, \tilde{u}, y)$ be a linear discrete-time process subject to disturbances

$$x(k+1) = Ax(k) + B\tilde{u}(k) + w(k), \tag{1a}$$

$$y(k) = Cx(k) + D\tilde{u}(k) + v(k), \quad k \in \mathbb{Z}_{0+} \tag{1b}$$

where $\mathbb{Z}_{0+}$ is the set of nonnegative integers, $x(k) \in \mathbb{R}^n$ is the process state, $y(k) \in \mathbb{R}^p$ is the process output, $\tilde{u}(k) \in \mathbb{R}^m$ is the commanded input, and $w(k)$ and $v(k)$ are the process noise and the measurement noise, respectively. We assume $w(\cdot)$ and $v(\cdot)$ to be white Gaussian random processes with zero mean and covariance matrices $Q$ and $R$, respectively: for all $k \geq 0$, $w(k) \sim \mathbf{N}(0, Q)$, $v(k) \sim \mathbf{N}(0, R)$. Hence, the state and output dynamics of $\Sigma(x, u, y)$ are Gaussian distributed stochastic processes.

We model the dropping of the packets by a stochastic discrete signal (event) $e(k) \in \{0, 1\}$, where 0 means that the packet at time $k$ has been dropped, while

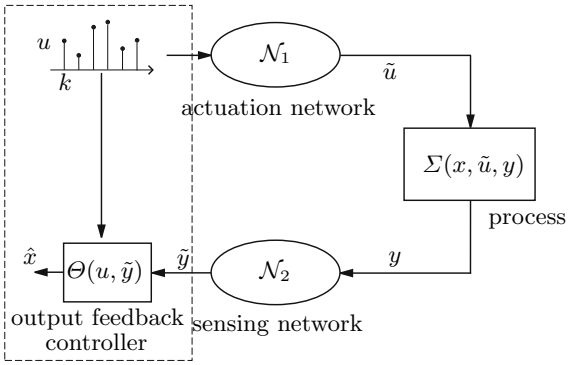



**Fig. 1.** Process and networked control architecture

1 means that the packet has been received. Consider a discrete-time signal $a(\cdot)$ with domain $D_a$. In our context $a(\cdot)$ can be either $u(\cdot)$ or $y(\cdot)$. The relation between $a(\cdot)$ and the corresponding network filtered signal $\tilde{a}(\cdot)$ is given by

$$\tilde{a}(k) = \begin{cases} a(k) \text{ if } e(k) = 1 \\ \varepsilon \quad \text{if } e(k) = 0, \end{cases} \tag{2}$$

where $\varepsilon$ is a special symbol indicating the lack of information, and the domain of $\tilde{a}(\cdot)$ is $D_{\tilde{a}} = D_a \cup \varepsilon$. When the loss occurs in $\mathcal{N}_1$ at time step $k$, we assume the process applies a backup input $\tilde{u}(k) = u_{\text{bu}}(k)$. Common choices for the backup input are $u_{\text{bu}}(k) = 0$, $u_{\text{bu}}(k) = \tilde{u}(k-1)$, and $u_{\text{bu}}(k) = \tilde{u}(k-1) + (\tilde{u}(k-1) - \tilde{u}(k-2))$. The backup input choice depends on the control strategy. In this paper, we consider the strategy as given, and known by the estimator.

We propose a stochastic network model with discrete state $N(k) \in \{0, 1, \ldots, s\}$. The network dynamics are modelled as a Markov chain,

$$\pi(k+1) = M^T \pi(k), \tag{3}$$

where $\pi \in \mathbb{R}^s$ is the vector of state probabilities at time $k$, i.e. $\pi_i(k) = \mathbf{P}[N(k) = i]$, $i \in \{0, 1, \ldots, S\}$, the superscript $^T$ indicates transposition, and $M \in \mathbb{R}^{s \times s}$ is the transition matrix of the Markov chain. The packet drop probability is state dependent, i.e. $p(k) = p(N(k))$, to represent different operating conditions of the network corresponding to different packet reception rates (PRR).

The proposed network model incorporates common models in the literature. The Poisson process model of packet drops with parameter $p$, $\mathbf{P}[e(k) = 0] = p$, $\mathbf{P}[e(k) = 1] = 1 - p$, $\forall k \in \mathbb{Z}_{0+}$, that results in a constant drop probability, is a particular case of model (3), where $i \in \{0\}$. Gilbert model [7] is another particular case of model (3), obtained for $i \in \{0, 1\}$, $p(0) = 0$, and $p(1) = 1$.

Two network states can model appropriately behaviors such as network overload and quality of service degradation [7]. We will focus on a two states Markov chain for notational simplicity; the presented approaches can be straightforwardly extended to the case where the Markov chain has a larger number of

states. The dynamics of $N(k)$ and the packet drop probability are described by

$$\pi(k+1) = M^T \pi(k),\ \mathbf{P}[e(k)=0|N(k)=R] = p_r,\ \mathbf{P}[e(k)=0|N(k)=U] = p_u, \tag{4}$$

where $M = \begin{bmatrix} p_{R,R} & p_{R,U} \\ p_{U,R} & p_{U,U} \end{bmatrix}$, and $\pi(k) = \begin{bmatrix} \pi_R(k) \\ \pi_U(k) \end{bmatrix}$, $\pi_i(k) = \mathbf{P}[N(k)=i]$, $i \in \{R,U\}$ is the vector containing the state probabilities, and we assume $p_u > p_r$. When $N(k) = R$ the network is operating in *normal (Reliable) state*, ensuring a certain performance. In the case $N(k) = U$ the network is in *degraded (Unreliable) state*, and the performance decreases. We assume that the communications are instantaneous, meaning that $u(k)$ sent through $\mathcal{N}_1$ is received at step $k$ by $\Sigma(x, \tilde{u}, y)$, and that $y(k)$ sent through $\mathcal{N}_2$ is received at step $k$ by $\Theta(u, \tilde{y})$. This is reasonable because in general communications are much faster than the process dynamics.

In general the state of the network is not directly observable since it depends on the interferences among network users sharing the same channel. We can only observe its effects, the packet drops, either by direct knowledge or by inference based on the process measurements. We model this estimation problem as a hidden Markov model (HMM) estimation problem [13]. Let $\mathcal{N}_i = I$ indicate the situation in which the $i^{th}$ communication link is a perfect link, $\tilde{a}(k) = a(k)\ , \forall k \geq 0$. Hence, there is no estimation of $\mathcal{N}_i$. Moreover, let $\mathcal{N}_i = \mathcal{N}_j$, $i \neq j$ indicate that two logical communication links correspond to the same physical network, hence $N_i = N_j$. For the system described in Figure 1 we analyze the problems:

$P1$. Estimate $N_1$ and $x$, when $\mathcal{N}_2 = I$ .
$P2$. Estimate $N_2$ and $x$, when $\mathcal{N}_1 = I$ .
$P3$. Estimate $N_1$, $N_2$ and $x$ when $\mathcal{N}_1, \mathcal{N}_2 \neq I$, for $(a)$, $\mathcal{N}_1 \neq \mathcal{N}_2$, $(b)$, $\mathcal{N}_1 = \mathcal{N}_2$.

The case where both links are perfect, $\mathcal{N}_2 = I$ and $\mathcal{N}_1 = I$, reduces to the well known problem of estimating the state of a stochastic linear process, easily addressed by Kalman filtering. We first solve Problem $P1$, where we assume to know $e(k)$ (e.g., by an *acknowledge* signal). Then, we extend the approach to the case where $e(k)$ must be inferred from the process output, so that the estimation of the process state and the estimation of the network sate are treated as a single hybrid estimation problem. We show that Problem $P2$ can be solved by the approach proposed for Problem $P1$, where the packet behavior for $\mathcal{N}_2$ is always known. Finally we combine the approaches to solve Problem $P3$.

## 3 Estimation of Sensing Network State

First, we propose a solution to Problem $P1$, the state estimation of a single network $\mathcal{N}$, assuming to have direct knowledge about the drop events $e(k)$. In this case no process information is used for network state estimation hence process and network state estimation are separated.

Let $\pi_{s|t}(k) = \mathbf{P}[N(k)=s|e(k)=t]$, $s \in \{R,U\}$, $t \in \{0,1\}$. By Bayes' rule,

$$\pi_{s|t}(k) = \frac{\mathbf{P}[e(k)=t|N(k)=s]\,\pi_s(k)}{\mathbf{P}[e(k)=t]},$$

and then by the total probability theorem

$$\pi_{s|t}(k) = \frac{\mathbf{P}[e(k) = t|N(k) = s]\, \pi_s(k)}{\sum_{j\in\{R,U\}} \mathbf{P}[e(k) = t|N(k) = j]\pi_j(k)}, \tag{5}$$

where $\mathbf{P}[e(k) = t|N(k) = j]$, $j \in \{R, U\}$ is defined by (4). If $e(k)$ is measured, the estimation of the network state is independent from the estimation of the process state. Moreover, if $p_r = 1 - \alpha\varepsilon$, $p_u = \beta\varepsilon$, where $\alpha, \beta > 0$, if $\varepsilon \to 0$, then $\pi_{R|1} = 1$, $\pi_{U|0} = 1$. Hence, estimation (5) applied to the Gilbert model in [7] results in a state with probability 1, and the other with probability 0, because the packet behavior is a perfect indicator of the network state.

In (5) we need to compute $\pi(k)$. We present two estimators that differ on the way such a computation is performed: a static estimator that at each step uses only the current measurement, and a dynamic estimator.

**Assumption 1.** *Markov chain* (4) *has reached its steady state.*

The stationary probability distribution of the Markov chain states $\lim_{k\to\infty} \mathbf{P}[N(k) = U] = \pi_U^\infty$, $\lim_{k\to\infty} \mathbf{P}[N(k) = R] = \pi_R^\infty$ can be computed from the equilibrium

$$\begin{bmatrix} \pi_R^\infty \\ \pi_U^\infty \end{bmatrix} = M^T \begin{bmatrix} \pi_R^\infty \\ \pi_U^\infty \end{bmatrix}, \tag{6}$$

that has a unique solution for an irreducible Markov chain. The *static estimator* is obtained by plugging the solution of (6) in (5):

$$\pi_{s|t} = \frac{\mathbf{P}[e(k) = t|N(k) = s]\, \pi_s^\infty}{\sum_{j\in\{R,U\}} \mathbf{P}[e(k) = t|N(k) = j]\pi_j^\infty},\ s \in \{R, U\},\ t \in \{0, 1\}. \tag{7}$$

The maximum likelihood estimate is

$$\hat{N}(k) = \text{argmax}_{\text{j}\in\{\text{R,U}\}}\pi_{j|t} \quad \text{if} \quad e(k) = t, \quad t \in \{0, 1\}. \tag{8}$$

The stationary estimator is simple, since the estimate can be statically computed by evaluating (7) and (8) at every step. In particular, since all the terms in (7) are constant, a lookup table mapping the packet event into the most likely network state can be precomputed and no memory is required. However, such an estimation scheme often leads to poor performance, because only the current measurement is considered, and this information is not used to update the probability of the network state.

We develop a *dynamic estimator* that uses memory to maintain an estimate of the probability of each network state value and uses the measurements on the packet drop to update such an estimate. We can perform estimation by using (3) to compute the predicted state probability to be used in (5), then using the result of (5) as the state probability estimate. This procedure is summarized in Algorithm 3.1, where $\pi_s(k|k) = \mathbf{P}[N(k) = s|e(k)]$, $s \in \{R, U\}$.

The main advantage of Algorithm 3.1 is that the measurements of $e(k)$ are used not only in the decision of the current state, but also to update the state

1. set $k = 0$ and let the initial probability vector be $\pi(k|k-1) = \begin{bmatrix} \pi_R(k) \\ \pi_U(k) \end{bmatrix}$;
2. while (TRUE)
   2.1. given $e(k) \in \{0, 1\}$ perform measurement update $\pi(k|k) = \begin{bmatrix} \pi_R(k|k) \\ \pi_U(k|k) \end{bmatrix}$;
   2.2. if $(\pi_R(k|k) \geq \pi_U(k|k))$ then $\hat{N}(k) = R$ else $\hat{N}(k) = U$;
   2.3. perform prediction by flowing the Markov chain $\pi(k+1|k) = M^T \pi(k|k)$;
   2.4. set $k \leftarrow k+1$ and $\pi(k|k-1) \leftarrow \pi(k+1|k)$;
end

**Algorithm 3.1.** Dynamic network estimation algorithm

probability, which will affect the future estimation steps. From another point of view, the estimate at time $k$ is directly affected by $e(k)$ and indirectly by $e(j)$, $j = 0, \ldots, k-1$, whose effects are stored in the current state probability. Algorithm 3.1 is the discrete-state version of the Kalman estimator. The algorithm can be initialized by setting $\pi(0|-1)$ to the solution of (6).

**Proposition 1.** *The network state estimate $\hat{N}$ obtained by Algorithm 3.1 is asymptotically independent of the algorithm initialization value.*

Proposition 1 states that when $k \to \infty$ the value $\hat{N}$ does not depend on the value of $\pi(0|-1)$. This property follows straightforwardly from the *exponential forgetting* property of the HMM filter, that states that for $k \to \infty$, $\pi(k|k)$ is independent of $\pi(0|-1)$. The exponential forgetting property of HMM filters, which include Algorithm 3.1 was proven in [14], where upper bounds on the exponential forgetting rate are also given.

## 4 Estimation of Sensing Network and Process States

We give now a solution to Problem $P1$, relaxing the hypothesis that $e(k)$ is measured. We modify the estimation algorithm to infer the value of $e(k)$ from the measurement $\tilde{y}(k)$, where $\tilde{y}(k) = y(k)$, since $\mathcal{N}_2 = I$. A simple way to obtain information about $e(k)$ is to add one bit in the measurement packet. The same logic that activates the backup input will also set such a bit to 1 if the command packet has been received, to 0 otherwise. When this strategy is applied, the results of Section 3 hold, and the network state estimation is separated from the process estimation. However, we develop here an estimation approach that is not based on such an additional information, which is useful when the measurement packet cannot be modified, and when $\mathcal{N}_2 \neq I$.

Let $\rho_{t|y}(k) = \mathbf{P}[e(k) = t|y(k)]$, $t \in \{0, 1\}$. Then $\mathbf{P}[N(k) = s|y(k)] = \sum_{t \in \{0,1\}} \mathbf{P}[N(k) = s|y(k), e(k) = t]\rho_{t|y}(k)$, $s \in \{R, U\}$, $t \in \{0, 1\}$. The information given by $y(k)$ about $N(k)$ is entirely contained in the packet event, hence

$\mathbf{P}[N(k) = s|y(k), e(k) = t] = \pi_{s|t}(k)$, and

$$\mathbf{P}[N(k) = s|y(k)] = \sum_{t\in\{0,1\}} \pi_{s|t}(k)\rho_{t|y}(k). \tag{9}$$

In this case the disturbances acting on the process affect also the estimation of the network state. We introduce the joint hybrid-space probability density function $f^{(k)}(e, y)$[1], where $e \in \{0, 1\}$, and $y$ is the process measurement. At a given time $k \in \mathbb{Z}_{0+}$, $f^{(k)}_{y|t}$ is the probability density function of $y(k)$ assuming the event $e(k) = t$. The function $f^{(k)}_{e|y}$ is

$$f^{(k)}_{e|y} = \sum_{t\in\{0,1\}} \frac{f_{y|t}(y(k))}{\sum_{j\in\{0,1\}} f_{y|j}(y(k))}\delta(e(k) - t), \tag{10}$$

where $\delta(\cdot)$ is Dirac's distribution. As a consequence

$$\rho_{t|y}(k) = \frac{f_{y|t}(y(k))}{\sum_{j\in\{0,1\}} f_{y|j}(y(k))}, \tag{11}$$

to be used in (9). Note that the additional uncertainty given by the process noise is contained in the factors $\rho_{t|y}(k)$, $t \in \{0, 1\}$. If the noise vectors in (1) are null and the state is known, there exists $\bar{t} \in \{0, 1\}$ such that $\rho_{\bar{t}|y}(k)$=1, and hence $\rho_{1-\bar{t}|y}(k) = 0$, since the inference on packet behavior is deterministic. Let $\hat{x}(k|k)$ be the estimate of the process state at step $k$, based on measurements available up to step $k$, and $f^{(k)}_x$ be the process state distribution function. Algorithm 3.1 is modified into Algorithm 4.1, where $\hat{x}(k|k, t)$ is the estimate of the process state at time $k$ using measurements until time $k$ and assuming $e(k) = t$, $f^{(k)}_{x|i}$, $i \in \{0, 1\}$ is the process state probability density under the assumption the packet has been dropped or received, respectively, and $f^{(k)}_{y|i}$, $i \in \{0, 1\}$ is the corresponding output probability density. In order to compute $f^{(k)}_{y|i}$ which is used in (10), we need to estimate the probability density function of the process state $f^{(k)}_{x|i}$, $i \in \{0, 1\}$, under the assumption the packet has been dropped or received, respectively. Since we consider linear systems subject to Gaussian noise, the obvious choice for the process state estimation is the Kalman filter. The values $\rho_{i|y}(k)$, $i \in \{0, 1\}$ are computed from the output error densities $f^{(k)}_{\varepsilon_y|i}$, $i \in \{0, 1\}$, that are a shifted version of $f^{(k)}_{y|i}$, $i \in \{0, 1\}$. At each step of the process state estimation: $(i)$, compute $\hat{y}(k|k, i)$, $i \in \{0, 1\}$, the estimated output under the assumption that the packet has been dropped or received, respectively; $(ii)$, compute $\varepsilon_y(k|k, i) = y(k) - \hat{y}(k|k, i)$, $i \in \{0, 1\}$, the output estimation errors in the above cases; $(iii)$, compute the probabilities in (11) by the Kalman Filter output error densities estimation $f^{(k)}_{\varepsilon_y|i}$, $i \in \{0, 1\}$. Since only a deterministic

[1] For simplicity, we will drop the superscript $^{(k)}$ when clear from the context.

1. set $k = 0$ and let the initial probability vector be $\pi(k|k-1) = \begin{bmatrix} \pi_R(k) \\ \pi_U(k) \end{bmatrix}$;
2. while (TRUE)
   2.1. compute $\hat{x}(k|k,t)$, $f^{(k)}_{x|t}$, $f^{(k)}_{y|t}$, $t \in \{0,1\}$ ;
   2.2. compute $\rho_{t|y}(k)$, and $\tau = \text{argmax}_{j\in\{0,1\}}\rho_{j|y}(k)$;
   2.3. compute $\pi_{s|t}(k)$ using (5) for $t \in \{0,1\}$, $s \in \{R,U\}$ ;
   2.4. compute $\pi_s(k|k) = \mathbf{P}[N(k) = s|y(k)]$, $s \in \{R,U\}$, by (9);
   2.5. if $(\pi_R(k|k) \geq \pi_U(k|k))$ then $\hat{N}(k) = R$ else $\hat{N}(k) = U$;
   2.6. perform prediction $\pi(k+1|k) = M^T\pi(k|k)$;
   2.7. update the process state probability density to $f_x(k) = f^{(k)}_{x|\tau}$;
   2.8. set $k \leftarrow k+1$ and $\pi(k|k-1) \leftarrow \pi(k+1|k)$;
end

**Algorithm 4.1.** Network estimation algorithm with packet behavior inference

value in the process state is affected by $e(k)$, $f^{(k)}_{y|0}$ and $f^{(k)}_{y|1}$ have the same shape (same covariance), but they are shifted (different average).

By construction, Algorithm 4.1 is the one-step maximum likelihood estimate of the network and process states. The performance of the continuous state estimate affects the discrete estimation by the terms $\rho_{i|y}$, $i \in \{0,1\}$, hence the process and network estimation performances are related.

*Remark 1.* Algorithm 4.1 assumes that the distribution functions $f^{(k)}_{y|e=1}$ and $f^{(k)}_{y|e=0}$ are different. If this is not the case, $y(k)$ does not give any information about $N(k)$, because the effects of the input on the measurement are delayed. Hence, the estimation must be delayed as well, performing the measurement update to obtain $N(k-\delta|k)$, where $\delta \in \mathbb{Z}_{0+}$ are the steps of delay, followed by $\delta$ steps of prediction. Furthermore, when the current input $u(k)$ and the backup input $u_{\text{bu}}(k)$ are the same, it is not possible to distinguish the packet drop and the packet reception from the measurement, similarly to what discussed in [15]. However, this does not degrade the estimation of the process state, since $u(k) = u_{\text{bu}}(k)$.

## 5 Estimation of Sensing and Actuation Network States

We analyze now Problem $P2$, where packet drops occur in the sensing network, i.e., $\mathcal{N}_2 \neq I$, and all the command packets reach the process, i.e., $\mathcal{N}_1 = I$. The process state can be estimated by applying a Kalman filter modified as in [16], where the authors perform the measurement update only if the measurement packet is received. However, in [16] the network model is static, hence there is no network state estimation. In the case of model (4), the estimation of $N_2(k)$

can be performed by Algorithm 3.1, since the value of $e_2(k)$ is always available to the estimator (it knows whether the measurement packet has been received or not), and the process state estimation does not affect the system state, thus the separation between network state estimation and process state estimation holds. On the other hand, the estimation of the process state is affected by the behavior of the network, since when the measurement packet is dropped, the process state estimate has to be updated in open-loop [16].

Consider now Problem $P3a$, where packet losses occur both in the actuation network and in the sensing network, $\mathcal{N}_i \neq I$, $i = 1, 2$. The uncertainties introduced by $\mathcal{N}_1$ and $\mathcal{N}_2$ are different: a packet loss in $\mathcal{N}_1$ causes the process to evolve in a different way from the commanded one. However, such a drop do not affect the measurements, and the estimation can still be performed by Algorithm 3.1, or by Algorithm 4.1. On the other hand, when packet losses occur in $\mathcal{N}_2$, the measurement is not available, and the only way to update the process state estimate (and the estimate of $\mathcal{N}_1$ if Algorithm 4.1 is used) is by prediction. Thus, losses in $\mathcal{N}_2$ are more critical than losses in $\mathcal{N}_1$ for the estimation problem.

Algorithm 5.1, where $e_i$ is the packet event in $\mathcal{N}_i$, $i = 1, 2$, can be applied for estimation in the case $\mathcal{N}_1, \mathcal{N}_2 \neq I$, $\mathcal{N}_1 \neq \mathcal{N}_2$. We consider the case where $e_1(k)$ is not measured, hence if $e_2(k) = 0$ no measurement is received, and the estimate of $x(k)$ and $N_1(k)$ is updated by prediction. If $e_1(k)$ is measured, the measurement update is always performed also on $\mathcal{N}_1$ (Algorithm 3.1 is used).

---

1. set $k = 0$ and for $\mathcal{N}_i$, $i = 1, 2$, set $\pi^{(i)}(k|k-1) = \begin{bmatrix} \pi_R^{(i)}(k) \\ \pi_U^{(i)}(k) \end{bmatrix}$, respectively;
2. while (TRUE)
   - 2.1. compute $\pi_2(k|k)$ and $\hat{N}_2(k)$ by Algorithm 3.1;
   - 2.2. if $(\pi_R^{(2)}(k|k) \geq \pi_U(k|k))$ then $\hat{N}_2(k) = R$ else $\hat{N}_2(k) = U$;
   - 2.3. if $(e_2(k) = 1)$
     - 2.3.1. compute $\hat{x}(k|k)$;
     - 2.3.2. compute $\pi^{(1)}(k|k)$ and $\hat{N}_1(k)$ by Algorithm 3.1 or by Algorithm 4.1;
   - 2.4. else
     - 2.4.1. set $\hat{x}(k|k) = \hat{x}(k|k-1)$ (open-loop prediction);
     - 2.4.2. set $\pi^{(1)}(k|k) = \pi^{(1)}(k|k-1)$ (open-loop prediction on (4)).
   - 2.5. if $(\pi_R^{(1)}(k|k) \geq \pi_U(k|k))$ then $\hat{N}_2(k) = R$ else $\hat{N}_2(k) = U$;

end

---

**Algorithm 5.1.** Estimation algorithm for the case $\mathcal{N}_1 \neq I$, $\mathcal{N}_2 \neq I$, and $\mathcal{N}_1 \neq \mathcal{N}_2$

We finally consider Problem $P3b$, where the actuation and sensing networks correspond to the same physical network, hence $\mathcal{N}_1 = \mathcal{N}_2$ and $N_1 = N_2 = N$. Since all the packet events refer to the same network, at each step we have up to two measurements that we can use to update the network state estimate.

In (4) the drop probability depends on the state of the network, and not on the behavior of the other packets, hence $\mathbf{P}[e_i(k)|e_j(k), N(k)] = \mathbf{P}[e_i(k)|N(k)]$, $i, j \in \{1, 2\}$, $i \neq j$, and $\mathbf{P}[e_1(k), e_2(k)|N(k)] = \mathbf{P}[e_1(k)|N(k)]\mathbf{P}[e_2(k)|N(k)]$. The measurement update of the network state estimate with two data is

$$\mathbf{P}[N(k)|e_1(k), e_2(k)] = \frac{\mathbf{P}[e_1(k)|N(k)]\mathbf{P}[e_2(k)|N(k)]\mathbf{P}[N(k)]}{\sum_{N(k)=\{U,R\}} \mathbf{P}[e_1(k)|N(k)]\mathbf{P}[e_2(k)|N(k)]}. \tag{12}$$

If the measurement packet is dropped, a single-measurement update (5) is performed, hence packet drops in $\mathcal{N}_2$ results again to be more critical for estimation.

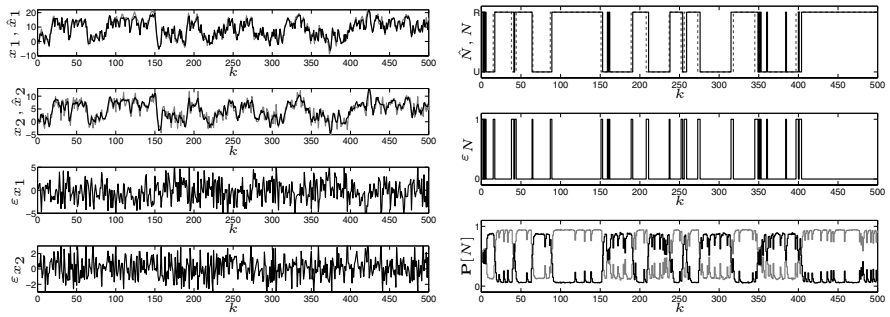
## 6 Simulations and Experiments

We consider $\mathcal{N}_2 = I$, and $\mathcal{N}_1$ defined by $M_1 = \left[\begin{smallmatrix} 0.98 & 0.02 \\ 0.06 & 0.94 \end{smallmatrix}\right]$, $p_r^{(1)} = 0.15$, $p_u^{(1)} = 0.80$. The process is a linear system with transfer function $G(s) = \frac{2.44}{s^2+2.4s+2.44}$, sampled at 2 Hz to obtain a discrete-time representation (1), where $A = \left[\begin{smallmatrix} 0.96 & -0.60 \\ 0.5 & 0 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 0.5 \\ 0 \end{smallmatrix}\right]$, $C = \left[\begin{smallmatrix} 0.40 & 0.54 \end{smallmatrix}\right]$, $D = 0$. We have used as backup strategy $u_{\text{bu}}(k) = 0$, and the noise terms are $v(k) \sim \mathbf{N}(0, 0.6)$, $w(k) \sim \mathbf{N}\left(0, \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]\right)$. All the simulations have been run for 500 steps and the packet drop events $e_1(k)$ and the random signal that generates the transitions of $N_1(k)$ are always the same. An extended discussion on the simulations is available in [17].

**Table 1.** Results of the simulations of the proposed estimation strategies

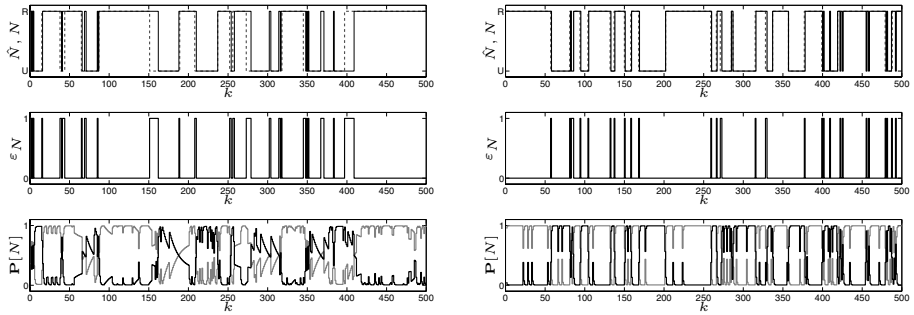
| $N(k)$ Estimator | $e_1(\cdot)$ | $v(k) \neq 0$ | $w(k) \neq 0$ | $x(k)$ Estimator | $\mathcal{N}_2$ | $\mathcal{E}_N$ |
|---|---|---|---|---|---|---|
| Static | meas | - | - | - | I | 79 |
| Dynamic | meas | - | - | - | I | 39 |
| Dynamic | inf | yes | yes | yes | I | 48 |
| Dynamic | emb | no | no | no | $\mathcal{N}_2 \neq \mathcal{N}_1 \neq I$ | $N_1 : 63,\ N_2 : 31$ |
| Dynamic | emb | no | no | no | $\mathcal{N}_2 = \mathcal{N}_1 \neq I$ | 27 |

The simulation results are summarized in Table 1, where $\mathcal{E}_N = \sum_k \varepsilon_N(k)$ is the cumulative network state estimation error, ( $\varepsilon_N(k) = |N(k) - \hat{N}(k)|$). Column $e_1(\cdot)$ refers to the knowledge about $e_1(k)$, where *meas* stands for measured, *inf* for inference, and *emb* means that the information on the reception of the command packet is embedded in the measurement packet. First we have simulated the strategies described in Section 3 based on measured $e(k)$. Table 1 shows that the performance of the dynamic estimator is clearly higher than the one of the static estimator. Next, we have simulated Algorithm 4.1 that is based on inference on the process measurements, and we have used a Kalman filter for hypothesis test, as described in Section 4. Table 1 shows that the network state estimate performance is degraded because of the effects of the process noise. The estimation of the process state is shown in Figure 2(a), where the estimated state components $\hat{x}_i(k)$, $i = 1, 2$, are plotted in black, the real state components



(a) Process state estimation (b) Network state estimation

**Fig. 2.** Joint actuation network and process states estimation with inference on $e(k)$

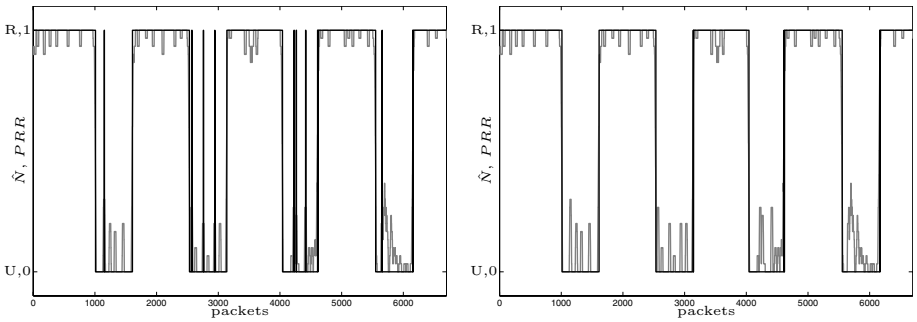


(a) Actuation network ($\mathcal{N}_1$) (b) Sensing network ($\mathcal{N}_2$)

**Fig. 3.** Actuation and sencing network states estimation $\mathcal{N}_1, \mathcal{N}_2 \neq I$, $\mathcal{N}_1 \neq \mathcal{N}_2$. The measurement packet contains information on command packet reception.

$x_i(k)$, $i = 1, 2$, in gray, and $\varepsilon_{x_i}(k)$ is the estimation error on the $i^{th}$ component of the process state vector at time $k$. The estimation of the network state is shown in Figure 2(b), where in the upper plot $N$ is the dashed line, and $\hat{N}$ is the solid line, and in the lower plot $\mathbf{P}[N(k) = R]$ is in gray, $\mathbf{P}[N(k) = U]$ is in black.

Finally, we have introduced another network $\mathcal{N}_2 \neq \mathcal{N}_1$ between the process and the estimator. The model of $\mathcal{N}_2$ is described by $M_2 = \left[\begin{smallmatrix} 0.97 & 0.03 \\ 0.09 & 0.91 \end{smallmatrix}\right]$, $p_r^{(2)} = 0.05$, $p_u^{(2)} = 0.87$. We have simulated Algorithm 5.1, in the case where the information about the command packet reception is embedded in the measurement packet. As a consequence, when this is dropped the estimate of $N_1$ is updated by open-loop prediction. Figure 3 reports the results of the simulation, with the same format as Figure 2(b). Table 1 confirms that the estimation of $\mathcal{N}_2$ is more precise than the estimation of $\mathcal{N}_1$, because it is always known whether the measurement packet has been received or not. The last line of Table 1 refers to a simulation where $\mathcal{N}_2 = \mathcal{N}_1$, which results in the most precise estimation, because at each step at least one data (and at most two) for the network state estimate is available.




(a) Estimated network state for $p_r = 0.10$, $p_u = 0.75$

(b) Estimated network state for $p_r = 0.20$, $p_u = 0.60$

**Fig. 4.** Application of the Algorithm 3.1 to a wireless sensor network with intermittent shadowing

As experimental validation, we have run Algorithm 3.1 on a dataset obtained from a real wireless sensor network composed of Telos T-mote Sky nodes where an object was sometimes shadowing the receiver node, similarly to what discussed in [6]. Part of the dataset was used to estimate the transition probabilities of the Markov chain, and the packet drops probabilities in each state, obtaining $p_{R,U} = 10^{-3}$, $p_{U,U} = 1.5 \cdot 10^{-3}$, $p_r = 0.10$, $p_u = 0.75$. We have run Algorithm 3.1 on the remaining data using the sequential packet ID to identify the occurrence of packet drops. The results are shown in Figure 4(a) where the estimated network state is plotted as a black line while the packet reception rate ($PRR$) obtained by averaging over a symmetric window of 30 packets is plotted as a gray line. The tracking is good, except for some false positives in which nonexistent state switches are detected. These can be removed by tuning the prediction model of the estimator. The behavior obtained by setting $p_r = 0.20$, $p_u = 0.60$ is shown in Figure 4(b), where the false positives are eliminated.

## 7 Conclusions

This paper has proposed different approaches to estimate the state of a networked control system, composed of the process state and the network state. We have shown the approaches in which the network state estimation can be be separated from the process state estimation and we have shown how the information losses in different places of the networked system affect the estimate. As shown in comparative simulations, the performance of the different approaches varies, and the choice of the one to be applied mainly depends on the overall networked system architecture.

The authors want to thank Pan Gun Park for performing the experiments on the wireless sensor network.

## References


1. Zhang, W., Branicky, M.S., Phillips, S.M.: Stability of networked control systems. IEEE Control Systems Magazine 21(1), 84–99 (2001)
2. Årzén, K.E., Bicchi, A., Dini, G., Hailes, S., Johansson, K.H., Lygeros, J., Tzes, A.: A component-based approach to the design of networked control systems. Europ. J. Control 2-3 (2007)
3. Samad, T., McLaughlin, P., Lu, J.: System architecture for process automation: Review and trends. J. Proc. Control 17, 191–201
4. Hespanha, J.P., Naghshtabrizi, P., Xu, Y.: A survey of recent results in networked control systems. Proc. of IEEE, Special Issue on Technology of Networked Control Systems 95(1), 138–162 (2007)
5. Goldsmith, A.: Wireless Communications. Cambridge Univ. Press, Cambridge, U.K (2004)
6. Willig, A., Kubisch, M., Hoene, C., Wolisz, A.: Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. IEEE Trans. Ind. Electr. 49(6), 191–201
7. Yu, X., Modestino, J., Tian, X.: The accuracy of Gilbert models in predicting packet-loss statistics for a single-multiplexer network model. In: INFOCOM. 24th Conf. of IEEE Computer and Communications Societies, pp. 2602–2612 (2005)
8. Jacobsson, K., Hjalmarsson, H., Möller, N., Johansson, K.H.: Some modeling and estimation issues in control of heterogeneous networks. estimation of RTT and bandwidth for congestion control applications in communication networks. In: Proc. 16th Intl. Symposium MTNS (2004)
9. Nilsson, J., Bernhardsson, B.: LQG control over a Markov communication network. In: Proc. 36th IEEE Conf. on Decision and Control., vol. 5, pp. 4586–4591 (1997)
10. Gupta, V., Hassibi, B., Murray, R.M.: On the control of jump linear Markov systems with Markov state estimation. In: American Control Conf., pp. 2893–2898 (2003)
11. Seiler, P., Sengupta, R.: An $h_\infty$ approach to networked control. IEEE Trans. Aut. Control 50(3), 356–364 (2005)
12. Bemporad, A., Di Cairano, S., Henriksson, E., Johansson, K.H.: Hybrid model predictive control based on wireless sensor feedback: An experimental study. In: Proc. 46th IEEE Conf. on Decision and Control., pp. 5062–5067 (2007)
13. Rabiner, L.R., Juang, B.H.: An introduction to hidden Markov models. IEEE Acoustic, Speech, Signal Proc. Mag. 3(1), 4–16 (1986)
14. Shue, L., Anderson, B.D.O., Dey, S.P.: Exponential stability of filters and smoothers for hidden Markov models. IEEE Trans. Signal Proc. 46(8), 2180–2194 (1998)
15. Epstein, M., Shi, L., Murray, R.M.: An estimation algorithm for a class of networked control systems using udp-like communication schemes. In: Proc. 45th IEEE Conf. on Decision and Control., pp. 5597–5603 (2006)
16. Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M.I., Sastry, S.S.: Kalman filtering with intermittent observations. IEEE Trans. Aut. Control 49(9), 1453–1464 (2004)
17. Di Cairano, S., Johansson, K.H., Bemporad, A., Murray, R.M.: Dynamic network state estimation in networked control systems. Tech. Report 2007-5 University of Siena (2007), `www.dii.unisi.it/~dicairano/papers/tr07ntwEstim.pdf`