NETWORK-INDUCED DELAY MODEL USING HMM FOR CAN-BASED NETWORKED CONTROL SYSTEMS EVALUATION

Rodrigo Vargas-Rodriguez[1] and Ruben Morales-Menendez[2]

Tecnológico de Monterrey, campus Monterrey
[1]Graduate Program in Automation
[2]Center for Innovation in Design and Technology
Avenida Eugenio Garza Sada #2501 Sur
64,849 Monterrey NL México

{A00790785,rmm}@itesm.mx(Rodrigo Vargas-Rodriguez,Ruben Morales-Menendez)

Abstract

In control systems a representation of the physical process which is to be controlled is needed in order to calculate control signals and keep the system stable. Networked Control Systems (*NCS*) are a special case of control systems where network-induced delays make the system stochastic and hard to predict. Pattern recognition techniques have been extensively used in learning the behavior of processes that present a certain degree of stochastic behavior. The Quality of Control (*QoC*) of each closed-loop system in a Networked Control System is strongly affected by the network-induced delay produced by sensors and control signals. Controller Area Network (*CAN*) is a popular real-time field-bus used for small-scale distributed environments such as automobiles. In *CAN* the delay exhibits a stochastic behavior and varies according to the network load. Since *QoC* is affected by delays, designing and evaluating a controller must take into account the effect of network-induced delays. A continuous Hidden Markov Model (*HMM*) for *CAN* network-induced delays is illustrated. The model plays the role of a classifier and an estimator; based on delay observations, the model can estimate the network load and predict future time delay values. The model was trained/tested using experimental data taken from a real *CAN* system with excellent results.

Keywords: Networked Control Systems, Controller Area Network, Hidden Markov Model.

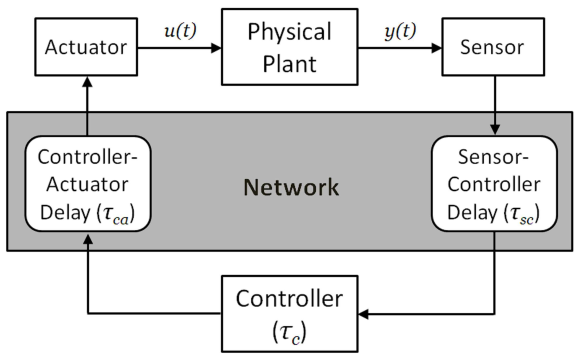
Presenting Author's Biography

Ruben Morales-Menendez is PhD in Artificial Intelligence. He is a full professor in the Center for Innovation in Design and Technology, and a consultant specializing in the analysis and design of automatic control systems. His research interests include artificial intelligence techniques for continuous control processes, manufacturing technology, mechatronics systems and educational systems. Currently, he is member of the Mexican National Researchers System.

1 Introduction

Recently, the use of common-bus architectures in control systems, instead of the traditional point-to-point architectures, has gained much more attention because of its benefits, which include modularization, decentralization of control, integrated diagnostics, quick and easy maintenance and low cost [1]. Common-bus control systems, called Networked Control Systems (*NCS*), exchange information and control signals using a serial communication network. Even though *NCS* offer a series of advantages over traditional control systems, there are certain characteristics related to communication channels (such as bandwidth) that are not considered in control systems design.

The network-induced delay is a dynamic phenomenon introduced by the communication channel used in control systems, Fig. 1. These are communication delays that arise because of sensors, actuators and controllers sharing a common network medium and can vary widely according to the message scheduling and the network overhead. Network-induced delays are inevitable and may cause system performance degradation and reduce the stability of the system.



Actuator
u(t)
Physical Plant
y(t)
Sensor
Controller-Actuator Delay (τ_{ca})
Network
Sensor-Controller Delay (τ_{sc})
Controller (τ_c)

Fig. 1 *NCS* with network-induced delays. τ_{sc} is the sensor to controller delay, τ_{ca} is the controller to actuator delay, and τ_c is the controller computational delay.

Controller Area Network (*CAN*) is a popular real-time and fault-tolerant field-bus used for small-scale distributed environments such as automobiles, aircraft and aerospace electronics, medical equipment, and factory and building automation. In *CAN*, the bus access time is generally non-deterministic, meaning that network-induced delays are random and time-varying [3]. Because of the uncertainty delays introduce in systems, traditional controller design techniques cannot be applied directly when designing a *CAN*-based *NCS*. A model of this phenomenon is needed in order to consider it during the design of the *NCS*.

The *CAN* time delay was modeled with a continuous Hidden Markov Model (*HMM*). The model is built from a series of delay observations taken from a real automobile *CAN* system implementation.

The paper is organized as follows. In section 2 a review of the state of the art of delay modeling is presented. Section 3 describes the experimental setup. Design of experiments is presented in section 4. In section 5 reviews the modelling approach. In section 6 results are discussed. Finally, conclusions and future work are presented in section 7.

2 State of the Art

Most *NCS* modeling research has focused on state space representations that include the time delay features. Little attention has been paid to the characterization and modeling of the network-induced delays in order to obtain an evaluation framework based on a reliable model.

A comparison between three popular control networks, including DeviceNet which is based on the standard *CAN* specification, is shown by [1, 2]. The study identifies the key components of the time delay in *NCS* through an analysis of network protocols and network dynamics. A timing analysis is presented where time delays are characterized by the networks parameters. The research shows the tradeoff between sampling time and network load and the acceptable working range of sampling periods in a *NCS*.

Different processing time models were built based on statistic approaches. DeviceNet models were obtained based on histogram parametrization. The developed models can adopt four different configurations: *zero*, *mean*, *normal* and *uniform*. The *zero* processing model lets the processing time equals zero; the *mean* model uses the mean value as the processing time; the *normal* model assumes a normal distribution of the processing time; and the *uniform* model uniformly assigns the processing time for each message.

Three models for network-induced delays are presented by [3]. The first is the simplest model, the delay is constant for all transfers in the network; even if the system has varying delays. The second model treats delays as random, taking the values from a probabilistic distribution and making them independent of previous ones. The third model is based on a Markov chain, which takes into account varying network loads. With the third model, different probability distributions can be used for τ_{sc} and τ_{ca}, and each probability distribution would belong to a certain network load. The models were experimentally tested.

A methodology of modeling *NCS* using *HMM* is presented by [6]. Delays are governed by a Markov chain for which the state transition probabilities were computed. Three network loads were considered (low, medium and high). The network loads represent the observations of the stochastic process modeled by the *HMM*.

[7] presented a model for network protocols and application performance evaluation. The model is a continuous-time *HMM* and it was built based on a series of end-to-end delays and packet loss observations from an internet application. In order to obtain a good model, data were collected for a wide variety of network settings. Then a *HMM* for each network setting was created. The *HMM*s where then integrated to perform network protocol and application simulations under differ-

ent network settings, representing real network scenarios. The models were experimentally validated using the Transmission Control Protocol (*TCP*) and a streaming video application.

Also, [7] proposed a discrete *HMM*. The *HMM* represented packet losses as large delays. This is because a loss is usually treated as a very large delay.

3 Experimental Setup

The experiments were performed in a multiplexed *CAN* X3 pedagogic scale model from EXXOTest® Fig. 2. The scale model is a training unit with real components of the Peugeot 807 that integrates three different network types: *CAN*, *LIN* (Local Interconnect Network) and *VAN* (Vehicle Area Network).

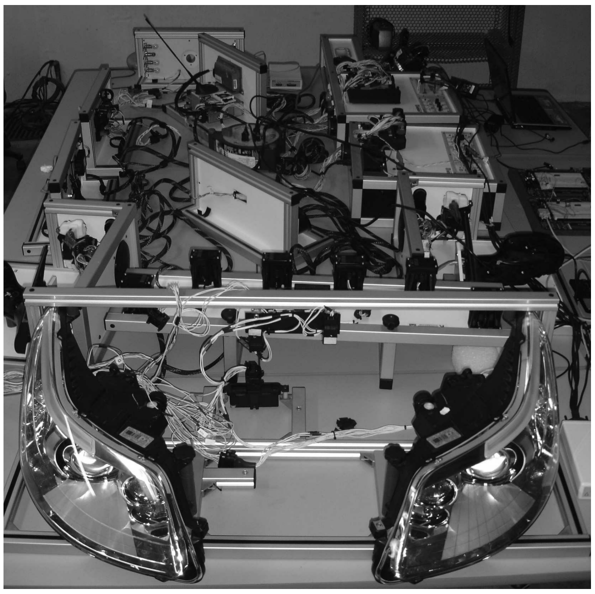


Fig. 2 A multiplexed CAN X3 scale model.

The *CAN* system is composed of twelve modules: Built - In System Interface (BSI), Built - In Supply Module (BSM), air conditioning, passenger door, driver door, front lights, back lights, dashboard, radio, AFIL (Lane Departure Warning System for the abbreviation in French), tow module and alarm. These modules are interconnected in three *CAN* networks: an inter-system network whose baud rate is 500 kbit/s (i.e. high speed *CAN*); a chassis network and a comfort network, both with a baud rate of 125 kbit/s (i.e. low speed *CAN*).

The inter-system network connects the diagnosis module, the motor status module and the steering wheel sensor. On the other hand, the comfort network is composed of the dashboard, the radio system, air conditioning, AFIL, driver and passenger door. Finally, the airbag, the alarm and the lights switchboard system integrate the chassis network.

In order to perform monitoring and load functions, communication with the network is done using the EXXOTest® USB-MUX-4C2L module (which allows interfacing a PC to the *CAN* bus) and the MUXDLL dynamic link library, also provided by EXXOTest®. The MUXDLL was used to interface an application created in *NI LabWindows/CVI* with the high speed *CAN* bus. The application was used to load the network with periodic messages and monitor the network load status.

To gather experimental delay data two network nodes were implemented using the Freescale^{TM} Semiconductor MC9S12C32 microcontroller which integrates a *CAN* controller (Motorola Scalable Controller Area Network module - *MSCAN*). Each MC9S12C32 was interfaced with the network using a Microchip® High-Speed *CAN* transceiver.

One of the network nodes was configured to send periodic messages and the other to receive them. Transmission and reception times were stored in each microcontroller memory.

4 Design of Experiments

A series of experiments were designed to determine network induced delays in the transmission between two nodes connected to the *CAN* system. The experiments were divided in sets in order to measure delays as a function of network load, periodicity and priority. All experiments used embedded timestamps in the messages, which were used to calculate the delay. The delay is obtained by subtracting the reception timestamp to the transmission timestamp.

Several factors can affect message time delays. Network load, message's priority, data packet length, message scheduling and transmission periodicity are the most influential factors.

For all messages the data packet size was held constant at 8 bytes, which is the maximum size for the standard *CAN* message format. The scheduling was also held constant because the EXXOTest® scale model already has a predefined schedule based on a real automobile implementation. Message priority was changed using one low priority and one high priority message id. Transmission periodicity was also changed, using 1 ms, 5 ms and 10 ms transmission periods. Finally, the network load was modified using different periodic messages.

Because of the network nodes that already exchange information when the automobile (EXXOTest® scale model) is turned on the lowest network load was 14%. This condition simulates a car under normal operation. Then periodic messages with low and high priorities were setup in order to load the network. Three network loads were implemented: low (14%), medium (38%) and high (73%).

Experiments were constructed for 18 different network settings. Each setting constituted a total of 1,000 samples divided in 10 sets (runs) of 100 samples. Network settings were adjusted by making combinations of message's priority, transmission periodicity and network load, Fig. 3. For each message, transmission and reception timestamps were recorded in the microcontrollers' memory.

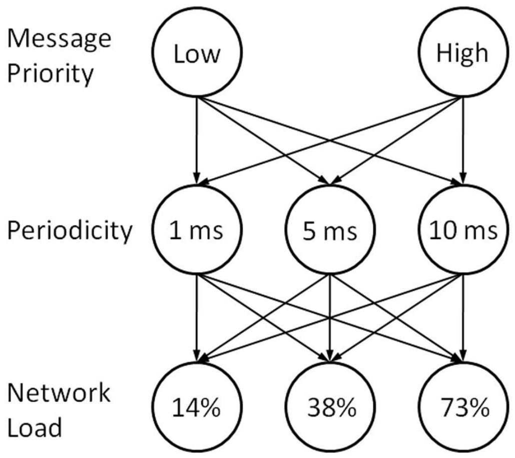


Fig. 3 Design of Experiments.

5 Modeling Approach

Characterizing real-world signals in terms of signal-models is a problem of fundamental interest. When these signals come from a stochastic process, statistical models are useful because they try to characterize only the statistical properties of the signal [8].

For completeness a review of some basic definitions is included. A *HMM* is characterized by [8]:

- *N*, number of states in the model. We denote the states as $\{S_1, \cdots, S_N\}$, and the state at time t a q_t.
- *M*, number of distinct observation symbols per state. We denote the individual symbols as $\{v_1, \cdots, v_M\}$.
- The state transition probability distribution $A = \{a_{ij}\}$

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], \quad 1 \leq i, j \leq N \tag{1}$$

- The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k | q_t = S_j], \quad \begin{array}{l} 1 \leq j \leq N \\ 1 \leq k \leq M \end{array} \tag{2}$$

- The initial state distribution

$$\pi = P[q_1 = S_i], \quad 1 \leq i \leq N \tag{3}$$

Given appropriate values of *N, M, A, B*, and π, the *HMM* can be used as a generator to give an observation sequence $O = O_1, \cdots, O_T$, Fig. 4. Then, a complete specification of an *HMM* requires specification of two model parameters (*N*, *M*), specification of observation symbols, and the specification of the three probability measures $\lambda = (A, B, \pi)$. The parameters N, M and λ

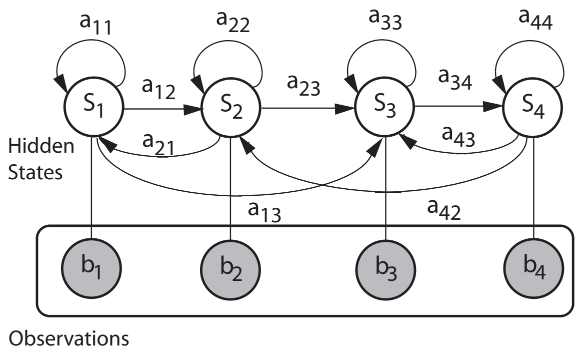


Fig. 4 Continuous *HMM* with four states $\{S_i\}_{i=1}^4$, and output probability density functions $\{b_j\}_{j=1}^4$

are learned from data. Given this model and the observation we can compute $P(O|\lambda)$.

In continuous *HMM*s observations follow a probability distribution. Each state can have one or more probability distributions following a mixture of Gaussians. A continuous *HMM* is specified by the same parameters $\lambda = (A, B, \pi)$, but with the difference that the B vector contains M observation probability distributions.

5.1 Baum-Welch algorithm

The Baum-Welch algorithm [8] is used to compute the model parameters (means, variance, and transitions) given the training data. It is an iterative process for parameter estimation based on a training data set for a given model λ. The goal is to obtain a new model $\bar{\lambda}$ where the auxiliary function $Q(\lambda, \bar{\lambda})$:

$$Q(\lambda, \bar{\lambda}) = \sum_Q P(Q \mid O, \lambda)\, log[P(O, Q \mid \bar{\lambda})] \tag{4}$$

is maximized. For this algorithm it is needed to define two more auxiliary functions:

$$\begin{aligned} \alpha_t(i) &= P(O_1^t, q_t = S_i \mid \lambda) && (5) \\ \beta_t(i) &= P(O_{t+1}^T \mid q_t = S_i, \lambda) && (6) \end{aligned}$$

Based on these two functions, forward and backward variables are defined. One of the variables is the probability for changing from state i at time $t = t$ to state j at time $t = t + 1$ can be defined as

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^M \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{7}$$

where $b_j(O)$ is a continuous output probability density function (*pdf*) for state j and can be described as a weighted mixture of Gaussian functions, as follows

$$\begin{aligned} b_j(O) &= \sum_{k=1}^M c_{jk} N(O, \mu_{jk}, U_{jk}) \\ &= \sum_{k=1}^M c_{jk} b_{jk}(O, \mu_{jk}, U_{jk}) && (8) \end{aligned}$$

where c_{jk} is the weight of the Gaussian k and $\mathcal{N}(O, \mu_{jk}, U_{jk})$ is a single Gaussian of mean value μ_{jk} and a covariance matrix U_{jk}.

The second variable is the *a posteriori* variable, that is the probability of being in state i at time $t = t$ given the observation sequence and the model.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \quad (9)$$

The relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ is given by

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \quad 1 \le i \le N, \; 1 \le t \le M \quad (10)$$

Assuming a starting model $\lambda = (A, B, \pi)$, α's and β's are evaluated using recursions (5) and (6), and then ξ's and γ's using equations (7) and (9). Next step is to update *HMM* parameters according to equations (11) to (13), known as *re-estimation formulas*.

$$\bar{\pi}_i = \gamma_t, \; 1 \le i \le N \quad (11)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \; 1 \le i \le N, 1 \le j \le N \quad (12)$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \; 1 \le j \le N, 1 \le k \le M \quad (13)$$

5.2 Viterbi Algorithm

The Viterbi algorithm [8] is a formal technique used to find the best state sequence, $Q = q_1, q_2, \cdots, q_t$, for the given observation sequence $O = O_1, O_2, \cdots, O_t$. To achieve this the quantity $\delta_i(i)$ has to be defined:

$$\delta_t(i) = \max_{q_1,\cdots,q_{t-1}} P[q_1, \cdots, q_t = i, O_1, \cdots, O_t \mid \lambda] \quad (14)$$

$\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state S_i. By induction $\delta_{t+1}(i)$ can be obtained:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}) \quad (15)$$

To actually retrieve the state sequence the argument which maximized (15) must be tracked, for each t and j. This is accomplished by using the array $\psi_t(j)$, which is initialized with 0 and then recursively updated. The procedure for finding the best state sequence is as follows:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \le i \le N \quad (16a)$$

$$\psi_1(i) = 0 \quad (16b)$$

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), 2 \le t \le T \quad 1 \le j \le N \quad (17a)$$

$$\psi_t(j) = arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], 2 \le t \le T \quad 1 \le j \le N \quad (17b)$$

3. Termination:

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \quad (18a)$$

$$q_T^* = arg \max_{1 \le i \le N} [\delta_T(i)] \quad (18b)$$

4. Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1 \quad (19)$$

6 Results

Fig. 5 shows delay measurements for a network setting of 5 ms of periodicity, low message's priority and the three different network loads. Similar results were obtained for 1 and 10 ms. However, only 5 ms periodicity results will be included because of limitations of space. As observed, delays tend to be near 0 and near 3 ms, such behavior was also noticeable in [3]. In addition, as the network load increase delays tend to disperse more, reaching peaks near 10 ms.

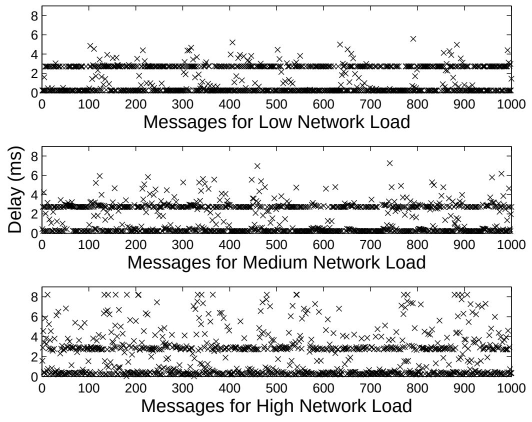


Fig. 5 Delay measurements for 5 ms periodicity.

Fig. 6 shows the procedure used to train and test the *HMM*s. The 70% of the experimental data was chosen randomly for the training stage; the remaining 30% was used for testing the results. This process was repeated 20 times and the results correspond to the average.

In order to evaluate the model performance, the five fundamental time-delay statistics [1] were computed for the experimental and the estimated model observations: minimum, maximum, mean, median and standard deviation. Statistic results for 5 ms periodicity are shown in Table 1.

A continuous *HMM* model was created for each different network load in order to obtain fine results and avoid being too general. The number of hidden states were 4 for low network load and 6 for medium and high network loads. The number of Gaussian mixtures was chosen according to delay data distribution shown in histograms, Figures 7-9.

Figures 7-9 show a comparison between real measurements and data estimated by the continuous *HMM* for

Tab. 1 Time-delay statistics for 5 ms periodicity

	Low Load		Medium Load		High Load	
Statistic	Real	Estimated	Real	Estimated	Real	Estimated
Minimum	0.2245	0.0041	0.2245	0.0143	0.0250	0.0054
Maximum	5.5780	5.0550	7.2480	5.9520	8.2100	8.6480
Mean	1.407	1.5140	1.5950	1.7190	2.0700	2.1640
Median	0.3870	1.1690	0.9613	2.3850	1.8440	2.5010
Std. Dev.	1.2960	1.3000	1.4120	1.1420	1.9800	2.0820

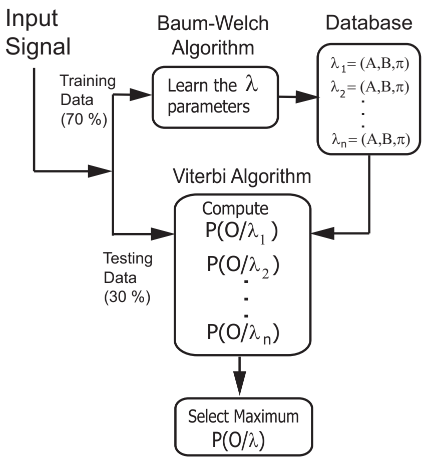


Fig. 6 Modeling procedure.

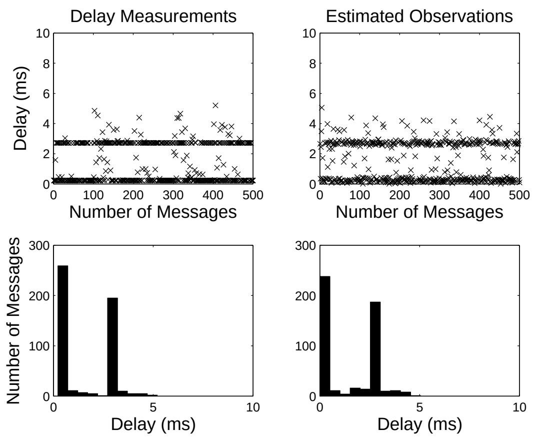


Fig. 7 *HMM* performance - 5 ms periodicity. Low Network Load

5 ms periodicity. The comparison was made by plotting side by side the time series for the experimental and the estimated observations. Histograms in Fig. Figures 7-9 also show the behavior and tendency of delays.

The previous results were obtained from the following initial state distribution matrix and state transition matrix for the low network load.

$$\pi_{low} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$A_{low} = \begin{bmatrix} 0.5473 & 0.4354 & 0.0173 & 0.0000 \\ 0.4982 & 0.4884 & 0.0134 & 0.0000 \\ 0.0000 & 0.0000 & 0.4871 & 0.5129 \\ 0.0000 & 0.0532 & 0.5583 & 0.3885 \end{bmatrix}$$

A graphical representation of these matrices is shown in Fig. 4.

As it can be observed from the obtained results, low sampling rates generally does not impose a real problem because delays are always less than one sample period. On the other hand, using high sampling rates improves performance, as noted by [1], but it also increases the message time delays and it can degrade *QoC*. This is because high sampling rates induce high traffic loads.

For deterministic protocols where message sizes and transmissions rates are fixed, the waiting time value at each source node can be calculated. However, for stochastic protocols such as *CAN* only statistical solutions of the network-induced delay can be derived [2]. Due to this fact, models based on probability distributions constitute a practical solution.

7 Conclusions

A network-induced delay model for *CAN* based *NCS* using continuous *HMM* was shown. The model was computed using measurements taken from a real automobile implementation of the *CAN* protocol. Early results are promising.

Special attention was paid in reproducing the dynamic behavior of delays under different network settings. Performance of a *NCS* at design stage can be tested with the models in order to predict different operational conditions.

A statistic analysis validated the results. It is shown that model behavior for different network loads and periodicities are very similar to the real behavior. Models were built for low priority messages since high priority messages will always have a near-deterministic behavior no matter the network load. Future work includes the adaptation of continuous *HMM* to *NCS* design methodologies and evaluation.

8 References

[1] F L Lian, Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet, *IEEE Control Systems Magazine*, 21(1), 2001, 66-83.

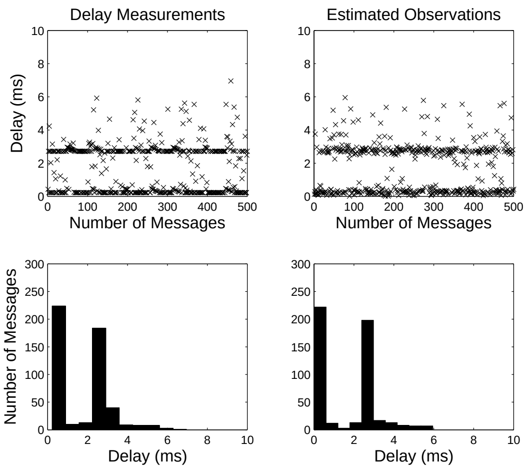


Fig. 8 *HMM* performance - 5 ms periodicity. Medium Network Load

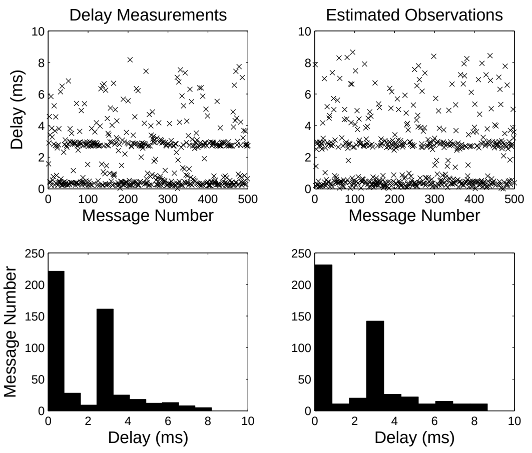


Fig. 9 *HMM* performance - 5 ms periodicity. High Network Load

[2] F L Lian, Time Delay Modeling and Sample Time Selection for Networked Control Systems, *Proc. ASME-DSC 2001 Int. Mechanical Engineering Congress and Exposition*, New York, USA, 2001, 1-8.

[3] J Nilsson, *Real-Time Control Systems with Delays*, PhD dissertation, Dept. Automatic Control, Lund Institute of Technology, Lund, Sweden, January 1998.

[4] Z Huo, Networked Control System: State of the Art, *Proc.* 5^{th} *World Congress on Intelligent Control and Automation*, Hangzhou, P.R. China, 2004, 1319-1322.

[5] Y Li, Control Methodologies of Large Delays in Networked Control Systems, *2005 Int. Conf. on Control and Automation (ICCA2005)*, 2, 2005, 1225-1230.

[6] F C Liu, Modeling and Analysis of Networked Control Systems Using Hidden Markov Models, *Proc.* 4^{th} *Int. Conf. on Machine Learning and Cybernetics*, 2, 2005, 928-931.

[7] W Wei, Continuous-time Hidden Markov Models for Network Performance Evaluation, *Performance Evaluation*, 49(1-4), 2002, 129-146.

[8] L R Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, 77(2), 1989, 257-286.