
Abouelabbas Ghanaim, Georg Frey

Saarland University, Chair of Automation, D-66123, Saarbrücken, Germany
{abouelabbas.ghanaim, georg.frey}@aut.uni-saarland.de

Abstract: This paper presents a two-step approach for modeling delays in Networked Automation and Control Systems (NAS/NCS). The approach combines the two well-known techniques: formal modeling using Colored Petri Nets (CPN) and mathematical modeling using Markov models to achieve this purpose. The first step is to build and simulate a structure-conserving hierarchical timed model for the whole NAS/NCS using (CPN). The step aims to generate extensive sampled time delay data records for detailed time analysis of delays. In the second step, Markov modeling is used to build compact mathematical models for network induced delays of NCS and response time of NAS. In addition, the second step introduces a new concept of mutual Markov modeling to analyze interaction between the two types of induced delays in NCS, namely, the sensor-to-controller time delay and the controller-to-actuator time delay. The proposed approach is new from two points of view: first using formal discrete event models for simulation in the modeling of NCS delays and their interaction, second using Markov models for modeling response time of NAS instead of estimating probability distributions of delays. In this paper a detailed procedure for the modeling steps will be provided in comparison to the overall concept presented in previous works. Finally, a numerical example is provided to demonstrate the applicability of the proposed mutual Markov modeling approach in the design of NCS state-feedback control.

1. INTRODUCTION

Networked Automation/Control Systems (NAS/NCS) are a type of distributed control systems where sensors, actuators and controllers are interconnected by real-time communication networks. Fig. 1 shows a schematic representation of a typical NAS/NCS with Programmable Logic Controller (PLC) as a system controller. The figure shows a typical time-driven execution platform for a NAS/NCS which is considered in this paper. As shown, the PLC is composed of two modules: a CPU module for processing the control program and an input/output communication module to scan remote input/output units. Each of the two modules has its own scan cycle: \( T_p \) and \( T_c \) respectively, and they exchange data through an internal backplane bus. In such systems, there is a time driven nature of all activities in the system such as sampling process signals, input/output scanning and execution of control program. The figure shows the sampling time \( T_s \) which clarifies the sampled time nature of the system in case of NCS. Several advantages of these systems include: reduced systems wiring, increased system agility and ease of system diagnosis and maintenance.

The term Networked Control Systems (NCS) in recent literature refers to the interdisciplinary research area, combining both network and control theory, in order to guarantee the stability and performance of an NCS (Xiao, 2000; Zhang, 2005). In contrast Networked Automation System (NAS) combines network and formal modeling tools to guarantee certain time performances for time critical automation tasks. NASs perform not only open loop automation tasks but also closed loop control tasks for time critical tasks such as position and motion control. This paper can be considered as a bridge linking the two research areas, which benefits from common modeling tools in the area of NAS to model network induced delays in NCS.

Depending on the devices sharing the network and the volume of information interchanged, the sender waits a variable time until the medium is granted to it. The stochastic nature of the shared resource occupation means a random access time. It will be denoted as \( \tau_{sc} \) the random access time in the sensor-to-controller (SC) link and \( \tau_{ca} \) in controller-to-actuator (CA) one. These random access times mean delays, which are a source of potential instability. In addition, especially in time-driven platforms such as PLC-based NCS, as the controller-plant communication uses a non-exclusive medium; it is difficult for the control device to precisely determine the sampling and actuation instants. The lack of
synchronization between controller and plant causes a significant worsening of the system response (Casanova 2003).

This paper is organized as follows: The next section describes the CPN modeling in more detail. Section 3 presents the Markov modeling approach. Section 4 presents a numerical NCS example. In the concluding section the results are commented and main ideas are summarized.

2. CPN MODELING AND SIMULATION

Colored Petri nets (CPN) are high-level Petri nets with graphical form and well-known semantic which allows for formal analysis and fast simulation. This feature makes CPN suitable to model concurrent and resource sharing systems as discussed in (Jensen, 2007). Applications of CPN to modeling and simulation of NASs and communication channels can be found in (Ghanaim, 2008, 2009; Marsal, 2006; Zaitsev, 2004) for the estimation of response time in open loop schemes.

Response time estimation in NAS can use different modeling approaches like probabilistic model checking (Greifeneder, 2008) or analytic calculus (Addad, 2010). In this paper CPN modeling approach is used to build a structure-conserving component-based model for NAS/NCS, in contrast to previous models built based on object oriented concepts as in (Marsal, 2006). To build the models, CPN tools developed by Jensen at University of Aarhus in Denmark 1982 is used with its existing simulation and monitoring features. The model explores the main features of PLC-based execution platforms which are the interaction between client/server approach for input output scanning and the cyclical execution of control algorithms. The model is simulated using a sampled signal generator model. Sensor (input), controller (PLC), and actuator (output) signals are recorded for delays calculation.

With CPN, a library of control and network components is constructed. The library includes: PLC with communication module, Ethernet switch, remote analog input/output unit with communication interface, and sample process. Each of these components is built in a separate window called a “Page”. In addition, there is a main page for the overall system outline which contains instances of the components. The required control system configuration can be built by “cloning” the required instances, and connecting them in the same structure as the system to be modeled. Hence, for a user of the library there is no need to access component pages.

Fig. 2 shows the main system layout of the CPN model of a typical NAS/NCS system. Separate analog input/outputs units are used for sensor and actuator signals to model the general case of independent sensor/actuator communication nodes. The double line rectangles represent compound transitions with instance name and instance class (small single line rectangles under each transition). Ovals represent places to pass parameters and store data. The PLC component has four parameters: PLC address, CPU scan time Tp, input/output scan time Tc, and remote input/output scan list. The Ethernet switch (SW) has one parameter which is a list assigning each MAC address to a switch port number. The remote analog input/output unit (I/O) has one parameter, which is the address of the unit. The sample process (Process) has one parameter, which is the sample time of the process. The time resolution used here is 1µs, which means that 10ms scan cycle is entered as an integer value of 10000.

Fig. 2. CPN model template with one PLC, one Ethernet switch, two analogue I/O-modules, and a sample process

Fig. 3 shows the PLC component page constructed of two parts to model the interaction between program scan cycle Tp and IO scan cycle Tc of the PLC. The left side represents the communication module IO scan cycle Tc. The scanning is done by sending client request packets from place IO Scan by the firing of the Send_R transition. The packets are sent as Modbus TCP/IP protocol, i.e. each packet has a data field (input/output), command field (read/write) and an acknowledge field. The scan cycle begins with sending data request packets to remote input/output units with addresses located in the place io_no and waits until all remote input/output units reply with acknowledge packets. The new input/output scan will start after receiving all the acknowledgment of the sent packets and the specified scan time that appears in the place IO ST has elapsed. The second part (right side) models the CPU/processor program scan cycle Tp which is divided into three times: read time equals to the “write” time and “execution” time. In the model the control algorithm is modeled as a unit function, i.e. the output is an assignment of the input.

Fig. 3 shows the PLC component page constructed of two parts to model the interaction between program scan cycle Tp and IO scan cycle Tc of the PLC. The left side represents the communication module IO scan cycle Tc. The scanning is done by sending client request packets from place IO Scan by the firing of the Send_R transition. The packets are sent as Modbus TCP/IP protocol, i.e. each packet has a data field (input/output), command field (read/write) and an acknowledge field. The scan cycle begins with sending data request packets to remote input/output units with addresses located in the place io_no and waits until all remote input/output units reply with acknowledge packets. The new input/output scan will start after receiving all the acknowledgment of the sent packets and the specified scan time that appears in the place IO ST has elapsed. The second part (right side) models the CPU/processor program scan cycle Tp which is divided into three times: read time equals to the “write” time and “execution” time. In the model the control algorithm is modeled as a unit function, i.e. the output is an assignment of the input.

Fig. 4 shows a CPN model for a 4-port Ethernet switch with a switch buffer and switching table. The switching table contains a list associating each MAC address to a switch port. Each port has in/out paths to simulate full duplex operation. The in/out paths have a variable called avail to simulate the
CSMA/CD i.e. sensing that path is free before sending or receiving packets.

Fig. 4. CPN model Ethernet switch.

Fig. 5 shows the CPN model for a remote analog input/output unit. The right part is built as input/output buffer through place IO_B which stores the integer pairs of input/output data. Two transitions: IO_Read and IO_Write model the required input/scan cycle time. The second part is modeled for input/output communication interface. Request and acknowledges are received and transmitted through rec and send transitions. The input/output data extracted from packets stored in places rec_b and send_b are interchanged through backplane using transition UP_Io.

The last CPN component model is the model for a sample processes. Fig. 6 shows the process model for sensor samples generator “dummy process” (left side) and a first order sampled system (right side). The dummy process model is used in this paper for sending distinguishable data samples (ramp signal) through the control system to be able to measure delays due to network and system configuration without the effect of process dynamics as used in (Nilsson, 1998) to measure delays in CAN and Ethernet networks. The sampling action is modeled using the time \( t_1 \) in the place Time with the black token “()”. The paper used two sample times for two modeling experiments: \( T_s=1\text{ms} \) with \( T_p=10\text{ms} \), \( T_c=17\text{ms} \) for the case of NAS and \( T_s=40\text{ms} \), \( T_p=10 \), \( T_c=17\text{ms} \) for the case on NCS. It is needless to say that in case of NAS, the sampling time is just used to generate a sequence of events, in contrast to the NCS in which the sampling time is a main requirement of a continuous plant.

The structure conserving CPN models of the NAS/NCS are validated against results of the real data measurements of an equivalent PLC setup in (Greifeneder, 2008). Fig. 7 shows the CPN-models response time in comparison to real measurements. The result shows that at large the model gives equivalent results. The remaining deviation between measured values and simulation results could be reduced by adjusting buffer size and manipulation delays in the Ethernet switch model and the CPU/processor model (data that is not available in detail for the setup). Overall, the NAS/NCS CPN models provide a very powerful flexible simulation platform that can be used all phases of the NCS design procedure. The models can help in study and analysis of many aspects of NCS such as: network induced delays, sampling time deviation, time synchronization, multi rate control, time versus event execution, etc.

3. MARKOV MODELING

Finding the probability distribution for the response time for open loop systems i.e. the round trip time elapsed by the data to reach from input port to the output port was addressed using CPN modeling in (Marsal, 2006) or using probabilistic model checking in (Greifeneder, 2008). In these approaches, the probability distribution is estimated using independent random data measurements without any assumption about the memory property of the system and the sampled data nature of the networked control system. In this paper, a comprehensive approach is proposed for modeling induced network delays separately and to address the correlation between the two types of delays.
Fig. 8 shows CPN models simulation data used in Markov modelling approach. In the case of 1ms sampling (NAS case), missing samples are neglected and delays counted from the point of view of the output unit. In case of 40ms the sample (NCS case) the vacant samples are considered by using the correction rule: if sample \( k \) is missed then \( \tau(k) = \tau(k-1) + T_s \). This correction is naturally done by the controller i.e when the sensor value of sample \( k \) is lost the controller uses the previous sensor value of sample \( k-1 \) again.

![Delays for input sampling of 1ms](image1)

![Delays for input sampling of 40ms](image2)

Fig. 8 Sensor-to-controller, controller-to-actuator delays, and total control delays for \( T_s = 1 \)ms and \( T_s = 40 \)ms input sampling and \( T_p = 10 \)ms and \( T_c = 17 \)ms

3.1 Proposed Markov modelling Procedure

A Markov model is a finite state model that describes a probability distribution over a number of possible sequences. The model states might correspond to network load states that lead to network induced delays. Each state emits observation and the states are connected by state transmission probabilities (Arauz, 2003; Fu-Chun, 2005).

Given a sequence of observations \( O_{seq} \), it’s assumed a hidden sequence of states \( S_{seq} \) corresponding to this observations sequence as shown below:

\[
O_{seq} = o_{(1)}, o_{(2)}, \ldots, o_{(k-1)}, o_{(k)}, o_{(k+1)}, \ldots
\]

\[
S_{seq} = s_{(1)}, s_{(2)}, \ldots, s_{(k-1)}, s_{(k)}, s_{(k+1)}, \ldots
\]

Where \( o_{(k)} \) and \( s_{(k)} \) are the observation and its corresponding state at sample time \( k \), \( o_{(k)} \in O = \{ o_1, o_2, \ldots, o_N \} \) a discrete observation set, and \( s_{(k)} \in S = \{ s_1, s_2, \ldots, s_M \} \) a discrete state set.

The Markov model \( \lambda \) for the system can be written as:

\[
\lambda = (A, B, \pi_0)
\]

Where \( A = [a_{ij}] = P(s_{(k+1)} = s_j \mid s_{(k)} = s_i) \) is the state transition probability matrix, \( B = [b_{ij}] = P(o_{(k)} = o_j \mid s_{(k)} = s_i) \) is the state observations probability matrix, and \( \pi_0 = [\pi_i] = P(s_{(0)} = s_i) \) is the initial state probability vector.

3.2 Markov Modelling Results

The proposed approach is demonstrated using delay sequences shown in Fig. 8 separately. Fig. 9 shows the probability of delays \( T_{sc} \) and \( T_{ca} \) in the case of 1ms input sampling. \( T_{sc} \) delays can be classified into two regions: small delays region \{4, 5, 6, 7, 8, 9\} with high probability and high delays region \{10, 11\} with small probability. Therefore, it is reasonable to model \( T_{sc} \) with a 2-state, 8-observation HMM \( \lambda_{sc} \) model.

![Sensor to controller delay (Tsc) distribution](image3)

![Controller to actuator delay (Tca) distribution](image4)

Fig. 9. Probability density distribution for \( T_{sc} \) and \( T_{ca} \) for 1 ms input sampling.

Using (Maximum likelihood) ML estimate and a Baum-Welch algorithm, the HMM \( \lambda_{sc} \) model parameters for the sensor to controller delay \( T_{sc} \) are given by:

\[
A_{sc} = \begin{bmatrix}
0.83 & 0.17 \\
1 & 0
\end{bmatrix}, \quad B_{sc} = \begin{bmatrix}
0.17 & 0.16 & 0.17 & 0.17 & 0.17 & 0 & 0 & 0.5 & 0.5
\end{bmatrix}
\]

Where \( T_{sc} \) delay equation can be written as (rect [] denotes a uniform distribution on this interval):

\[
T_{sc} = \begin{bmatrix}
\text{rect}[0,4.9] & x_{s1} = s_1 \\
\text{rect}[0,10.11] & x_{s2} = s_2
\end{bmatrix}
\]

In a similar way, a 2-state, 6-observations HMM \( \lambda_{ca} \) can be estimated with parameters:

\[
\pi_{ca} = \begin{bmatrix}
0.14 & 0.86 \\
0.86 & 0.14
\end{bmatrix}, \quad Q_{ca} = \begin{bmatrix}
0.14 & 0.57 & 0.29 & 0 & 0 & 0 \\
0.57 & 0.29 & 0.57 & 0 & 0 & 0
\end{bmatrix}
\]

\[
T_{ca} = \begin{bmatrix}
[33,41,15] & x_{s3} = s_3 \\
[27,29,31] & x_{s4} = s_4
\end{bmatrix}
\]
The important feature of the proposed modeling procedure is the ability to model the correlation between the two delays. The idea which is previously mentioned in (Nilsson, 1998) is used to estimate a mutual model for the two delays. The mutual model -as shown in Fig. 10- is constructed by combining the previously independent models together to yield a single model for the two delays. The number of states of the mutual model will be $M_{m}=M_{ca} \cdot M_{sc}$ and state transitions will be from one model to the other without inter-model state transitions. This model is called 2-state advance by input sample transmission i.e. for each input sample the model evolves two times, one for $\tau_{sc}$ delay and the other for $\tau_{ca}$ delay. The HHM $\lambda_{com}$ parameters are estimated using the same ML and Baum-Welch to:

$$A_{m} = \begin{bmatrix} 0 & 0.58 & 0.42 & 0 & 0 & 0.06 & 0.14 & 0 & 0.85 & 0.15 & 0 \\ 0 & 0.58 & 0.42 & 0 & 0 & 0.06 & 0.14 & 0 & 0.85 & 0.15 & 0 \\ \end{bmatrix} \cdot \begin{bmatrix} 0 & \lambda_{com} & 0 \\ \end{bmatrix}$$

The Mutual model $A_{m}$ can be decomposed into new separate Markov models $\lambda_{ca/sc}$, $\lambda_{sc/ca}$. The new models are interesting especially $\lambda_{ca/sc}$ in estimating $\tau_{ca}$ given $\tau_{sc}$. Since the $\tau_{sc}$ delay is unknown at the time of control value calculation, in contrary to the $\tau_{ca}$ delay, which could be calculated in the case of time stamped sensor signals.

Fig. 10. State diagram of the composite HMM $\lambda_{com}$ for $\tau_{sc}$ and $\tau_{ca}$ delays.

The verification of the proposed Markov models is done according to the type of model used. In case of direct Markov models, the stationary (limiting) distribution of the Markov model versus the probability distribution of the CPN model delays is used. The stationary distribution can be calculated analytically by solving the balance equation $\pi \mathcal{A} = \pi$ for eigenvector corresponding to a unity eigenvalue as shown in (Gilks, 1995). In case of hidden Markov modeling, comparison of the probability distribution of CPN delays with Markov generated delays using Markov model simulation is used as a measure. Fig. 12 shows the two methods for the verification of $\tau_{sc}$ delays for 10ms, and 40ms input sampling. The figure shows a good performance in estimating the probability distribution of induced delays. The next section demonstrates the importance of the mutual Markov modelling for the case of NCS using a numerical example.

Fig. 12. Validation of Markov models using: a) HMM simulation, b) limiting distribution for DMM

### 4 A NUMERICAL EXAMPLE

Consider the cart and inverted pendulum problem which is a fourth order unstable system. The state variables are $[x \; \dot{x} \; \theta \; \dot{\theta}]$, with $\theta(0)=0.1$ rad and all other initial state variables are zero. The parameters are the same as used in (Xiao, 2000): $m_1=1$kg, $m_2=0.5$kg, $L=1$m, the sampling time is $T_s=20$ms. Assume the system is controlled using a PLC with parameters $T_{sc}=T_{ca}=20$ms. The induced random delays resulted from CPN model simulation measured in number of samples are $M(\tau_{sc})=M(\tau_{ca})=[1,2]$ with mutual transition probability matrix:

$$A_{m} = \begin{bmatrix} 0 & 0.95 & 0.051 & 0 & 0 & 0.71 & 0.29 & 0 & 0 \\ 0 & 0.99 & 0.014 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ \end{bmatrix}$$

We design a single mode-dependent control low for the system using the same algorithm used in (Xiao, 2000) using the mutual probability transition matrix $A^{ca/sc}$ for design stabilizing control low for the system modeled as jump linear system (JLS). First we design an LQR using weighting matrix $Q=I_4$ for the state and $Ru=1$ for the control signal. We get:

$$K=[0.9147 \; 2.3520 \; 39.5869 \; 10.9682]$$
After solving the (linear Matrix Inequalities) LMI stability condition in ten steps, results are: \( \min \alpha = 0.9823 \) and 
\[
K_1 = [2.2854 \quad 6.5253 \quad 60.6805 \quad 22.0100], \\
K_2 = [0.2486 \quad 0.0001 \quad 0.0027 \quad 7.5940].
\]
The algorithm used for solving the stability condition using the mutual transition probability shows good results as it can be considered a robust algorithm i.e. it converges shortly to a feasible solution compared to algorithms use independent Markov models (Xiao, 2000; Zhang, 2005).

Fig. 13 shows the response of the initial pendulum position due to a random delay sample according to the required transition probabilities. As seen the blue line (without marks) is the ideal system response with LQR and without delays. The green one (with circle marking) is the system with LQR and with delays. The red one (with + markings) is the response with single mode-dependent control with delays. It is noticed in this probability sample that the proposed control is close to ideal response, also the regular LQR can stabilize the system but there is no guarantee to stabilize all the random sequences because it is not a feasible solution for the LMI stability condition.

5. CONCLUSION

The paper presented a comprehensive procedure for modelling delays in networked automation/control systems. First a structure-conserving CPN model is built concentrating on the effect of client/server input/output scanning and program execution cycles for typical NCS cyclic controllers. By simulating the CPN model, sequences of delays are calculated.

Second, Markov analytical models are obtained for both sensor-to-controller \( \tau_{sc} \) and controller-to-actuator \( \tau_{ca} \) delays independently and in mutual/composite way to benefit from the correlation between the two delays. Based on probability distribution of delays, hidden or direct Markov models are selected. The proposed mutual Markov delay models are used in the third part with a numerical example to demonstrate the procedure. The results show that the proposed modelling and control procedure is efficient in control law design for the case of NCS.

REFERENCES


Proc. of EEE Vehicular Technology Conference, (2), 771 – 775. PA, USA.

Casanova V., Salt, J. (2003). Multirate control implementation for an integrated communication and control system”, 
Control Engineering Practice, Vol. 11, pp. 1335-1348.


Markov Chain Monte Carlo in Practice, Taylor & Francis Inc.

17th IFAC World Congress, 5113-5118. Seoul.


IEEE TASE, (3), pp.287-296. TX, USA.

Proc. ACC. Vol. 3, pp. 2199-2204, Illinios, USA.
