MODEL BASED PREDICTIVE NETWORKED CONTROL SYSTEMS

By
AHMET TEOMAN NASKALI

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

SABANCI UNIVERSITY
Summer 2006

MODEL BASED PREDICTIVE
NETWORKED CONTROL SYSTEMS

APPROVED BY:

AHMET ONAT (Dissertation Advisor)	------------------------------------
ASIF ŞABANOVİÇ	------------------------------------
AYHAN BOZKURT	------------------------------------
ERKAY SAVAŞ	------------------------------------
GÜLLÜ KIZILTAŞ ŞENDUR	------------------------------------

DATE OF APPROVAL: ------------------------------------

MODEL BASED PREDICTIVE
NETWORKED CONTROL SYSTEMS

Ahmet Teoman NASKALİ

EECS, M.Sc. Thesis, 2006

Thesis Supervisor: Ahmet ONAT

Keywords: Networked Control Systems, predictive control, model based control

ABSTRACT

Advantages of networked control systems (NCS) are very diverse and NCS's address many of the demands of industrial development. As more and more sophisticated problems arise, networked control systems will not only become a convenience or an advantage but they will become an indispensable necessity. However usage of networked control systems introduces different forms of time-delay uncertainty in closed-loop system dynamics. These time delays are caused by the time sharing of the communication medium as well as computation time necessary for control algorithms and digital to analog conversions and have a destabilizing effect on system performance.

Computational power of computers has increased dramatically; networks speed has also increased. Although both the network and computer architectures have tended to improve throughput over time, their real-time characteristics have not evolved to match the requirements from a control point of view. New control methodologies that cope with these factors and even take advantage of them are emerging.

This work first examines some current methods in design and implementation of networked control systems that try to improve existing methods. Then a novel networked control system architecture that runs under non ideal network conditions with packet loss and noise is introduced. The proposed network control system architecture uses a model to predict the plant states into the future and generate corresponding control signals, then an array of the predicted control signals is sent to the actuator node side of the NCS rather than a single control signal like in basic networked control systems. This array of signals can control the plant if they are applied consecutively with sampling time intervals. However this is not the case under ideal conditions, where the network is lossless. Only the first control signal in each array is applied to the plant as a newer packet arrives every sampling period. The remainder of the array of predicted control signals is only used when packet loss occurs. This approach enables the system to be controlled in a pre-simulated manner and stability can be maintained even with high packet loss probabilities. Synchronization of the network elements becomes a major problem in this approach since models are involved. Synchronizing the actuator and controller nodes is done by an algorithm that can identify control signal arrays that have trustable information. Also the controller has a distributed architecture; some parts of the controller are implemented in the sensor node. This is to ensure that sensor to controller synchronization is not broken.

The proposed model based predictive networked control system architecture was tested on a DC motor. The effects of packet loss were examined to reveal that the packet loss does not cause destabilization of the system, when packet loss occurs and the control packet cannot be sent to the actuator node, which prevents the changes in reference from being applied to the plant. The overall effect is the retardation of the response of the plant to the reference. Effects of noise are also examined. Under low packet loss conditions noise does not have an unusual effect on the system but when packet loss increases noise cannot be tolerated because the feedback loop is interrupted due to packet loss.

Finally a method for determining the number of predictions to be made at the controller node (the prediction horizon) is suggested. The systems settling time is examined and the settling time is taken as the basis for the prediction horizon. The transmission of a single array of control signals from the controller node to the actuator node will enable the system to reach the desired reference. However this approach is only valid for open loop stable systems.

MODEL TABANLI ÖNGÖREN AĞ BAĞLANTILI KONTROL SISTEMLERI

Ahmet Teoman NASKALİ

EECS, Yüksek Lisans Tezi, 2006

Tez Danışmanı: Ahmet ONAT

Anahtar Kelimeler: Ağ bağlantılı kontrol sistemleri, öngören kontrol, model tabanlı kontrol

ÖZET

Ağ bağlantılı kontrol sitemlerinin getirileri çeşitlidir ve sanayinin gelişiminin birçok talebine karşılık vermektedir. Günden güne karmaşıklaşan sorunlarla birlikte ağ bağlantılı kontrol sistemleri bir kolaylık olmaktan ziyade vazgeçilmez bir unsur olacaklardır. Ağ üzerinden kontrol yapan sistemlerin getirileri olduğu gibi gecikmelerden ötürü oluşan belirsizlikler birtakım sorunları da beraberinde getirir. Bu gecikmelerin sebebi haberleşme ortamındaki zaman paylaşımı ile kontrol algoritmasını hesaplamak ve dijitalden analoğa çevrim yapmak için gerekli zamanın değişken olmasıdır. Gecikmelerin sistem performansı üzerinde denge bozucu etkisi vardır. Bilgisayarların ve bilgisayar ağlarının süratı yakın zamanda kaydadeğer artış göstermiştir. En çok veriyi aktarmak üzere optimize edilmiştir olan bilgisayar sistemlerinin gercek zamanlı karakterleri kontrol bakış açısıyla özel bir verinin süratli transferi icin optimize edilmemiştir.Bu koşullarda çalışabilen hatta bu koşulları avantaja çevirerek kullanan kontrol sistemleri piyasaya çıkmaktadır.

Bu çalışma öncelikle ağ bağlantılı kontrol sistemleri ile ilgili tasarım ve uygulama alanındaki güncel gelişmelerden bahsetmektedir. Ardından gürültülü ve ideal olmayan network koşullarında çalışabilen yeni bir ağ bağlantılı kontrol sistemi mimarisi tanıtılmaktadır. Önerilen ağ bağlantılı kontrol sistemi mimarisi kontrol edilen sistemin modelinden faydalanarak tesisin gelecekteki halini thamin ederek bu hallere de uygulanması gereken kontrol siynallerini üretmektedir. Bu sinyallere öngörülen sinyaller denmektedirler. Kontrol birimi eyleyici birimine gerçek değerlerden ve öngörülen değerlerden yola çıkılarak hesaplanan bir sinyal paketi yollamaktadır. Bu paketteki sinyaller kontrol edilecek sistemi belirli bir süre kontrol edebilecek özelliktedir fakat ideal koşullar altında bu paketteki sinyallerin çoğu kullanılmamktadır. Kopukluk olması durumunda ise en son yollanan paketteki öngörülen sinyaller kullanılmaya başlanır. Bu sayede kontrol edilecek sistem ağda kopukluk olması durumunda öngörülmüş şekilde kontrol edilebilmektedir. Bunun neticesinde sitem dengesini bozmamaktadır ama kaçınılmaz olumsuzluk olarak referans sinyalinin kontrol edilen sisteme etkimesi gecikir. Bu yaklaşımda model kullanıldığı için ve zaman karşı hassas birçok algoritma olduğu için eşzamanlama çok önemlidir. Kontrol birimi ile eyleyici birim arasında oluşabilecek eşzamanlama sorunların sebebi kullanılan sistem modelinin durumu ile geçek sistemin durumunun biribirinden farklılaşmasıdır. Bu sorun eyleyici tarafında kullanılan ve gelen konrol paketlerinden güvenilmez olanları ayıklayan bir algoritma tarafından çözülmektedir. Algılayıcı birimi ve kontrol birimi arasında oluşabilecek eşzamanlama sorunları da kontrol biriminde uygulanan algoritmanın dağıtık bir yapıda uygulanmasıyla çözülmüştür. Algılayıcı biriminden bir ağ paketi geldiği zaman, kontrol biriminin kullanacağı zamana bağlı değişkenler hesaplanmış olarak gelmektedirler. Bu sayede kontrol birimi algılayıcıdan yeni bir ağ paketi aldığı anda eşzamanlama tamamlanmış olacaktır.

Önerilen ağ bağlantılı kontrol sistemi mimarisi bir DC motor üzerinde simüle edilmiştir. Sitemin ağ üzerinde oluşan verikaybına karşı dayanıklılığı sınanmıştır ve önerilen sistemin veri kaybına karşı dayanıklı olduğu tespit edilmiştir. Veri kaybının kaçınılmaz neticesi olarak referans sinyalinin kontrol edilen sisteme uygulanmasında gecikme görülmüştür. Gürültünün de sistem üzerindeki etkisi incelenmiştir. Ağda oluşan düşük miktarlardaki veri kaybının sistem üzerinde etkisin az olduğu görülmüştür. Ama veri kaybının yüzdesi arttıkça sistemin geribesleme döngüsünde oluşan kopukluk yüzünden kontrol birimi hatadan haberdar olamamakta ve sistemde oluşan hatalara müdahale edememektedir.

Son olarak öngörü penceresinin boyutunu belirlemek için bir metot önerilmektedir. Sistemin maksimum refererans değişikliğinde yatışma süresi

incelenmetedir. Öngörü penceresinin yatışma süresi kadar tutulması yapılacak işlem yükünü en az seviyede tutmaktadır. Bu metot yanlızca açık döngüde kararlı sitemler için geçerlidir.

“To my family”

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor Ahmet Onat for
his valuable advice and guidance of this work. I am grateful to him not
only for the completion of this thesis, but also for his unconditional
support from the beginning.
Special thanks goes to Emrah Deniz Kunt, Ahmet Altınışık and Kazım Çakır who are all great friends who have shared everything they know and who have always found time to discuss my ideas no matter how crazy.
I would also like to thank Gönen Eren for logistical support.
I would like to thank all my friends and colleagues at the Mechatronics lab, specially Ertuğrul Çetinsoy, Eray Doğan and Selim Yannier for their friendship and assistance.
Of course a very special thanks goes to the boss Asıf Šabanoviç.
I would like to thank my family, for their unlimited support and trust that made everything possible for me.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF SYMBOLS

$\tau_{sc}(t_k)$	Sensor node to controller node network delay at time t_k
$\tau_c(t_k)$	Controller node computational delay at time t_k
$\tau_{ca}(t_k)$	Controller node to actuator node delay at time t_k
$\tau_a(t_k)$	Actuator node computation delay at time t_k
$\tau_s(t_k)$	Sensor node computation delay at time t_k
$\tau(t_k)$	Sensor node to actuator node total delay at time t_k
t_k	k'th sampling time of system
$u(t_k,0)$	Control signal calculated from sensor values at time t_k
$\hat{u}(t_k,i)$	Estimated control signal, t = time, i = signal number
$x(t_k)$	Plant sates at time t
$\hat{x}(t_k)$	Estimated plant states at time t_k
$\dot{x}(t_k)$	Derivative of plant states
F	Plants continuous state matrix
G	Plants continuous input matrix
H	Plants output matrix
J	Plants direct transition matrix
Φ	Plants discrete state matrix
Γ	Plants discrete input matrix
P	Plant
$\hat{P}$	Plant model
$Pt(t_k)$	Network packet, generated at sampling period t_k
$r(t_k)$	Reference
$y(t_k)$	Plant output
$KP(t_k)$	Proportional component of control signal
$KD(t_k)$	Derivative component of control signal

K	Proportional gain
T_d	Sampling interval length

TABLE OF ABBREVIATIONS

NCS	Networked Control System
MBPNCS	Model Based Predictive Networked Control System
A/D	Analog to Digital converter
D/A	Digital to Analog converter
QoS	Quality of Service
MPC	Model Predictive Control
ZOH	Zero Order Hold
SF	Sensor Flag
PD	Proportional Derivative (controller)
RMS	Root Mean Square
BNCS	Basic Network Control System

1 INTRODUCTION

In the recent years, digital control systems have gained importance. Use of computers or microcontrollers in control applications is becoming more and more common. Among the many advantages of digital approaches is the ability to program higher capability into the controller increasing the decision making capabilities of the controller rendering the control system more versatile. Another advantage is the fast and easy re-configurability of the systems. One hardware design can be used to control many different applications with a mere software change. This not only reduces design time but also brings down manufacturing costs of such hardware.

A digital control system is made up of the following basic components: Analog to digital converters (A/D) discretize plant outputs and provide them to the controller in a digital form. The conversion is done at sampling times, $t_{k.}$ Digital to analog converters (D/A) provide means of outputting calculated control signals to the plant. Digital to analog converters also function in a discrete way, the generated signal is held constant during the sampling time t_k. Digital control systems contain both continuous-time signals and sampled or discrete time signals.

Since network components have become more affordable, it has become easier to incorporate them in the implementation of digital control systems which has reduced the necessity of a single computer to deal with all aspects of controlling the plant. Components that cooperate over a network simplify the design process by providing a common interface for communication and enable off the shelf parts to be used in many different applications. With the need to rapidly reconfigure systems that can span physically much larger spaces, the importance networked control systems has increased.

A Networked Control System (NCS) is a feedback control system where the control loop is closed over a communication network. The controller and the plant are physically separated. Actuators and sensors which interact with the plant are linked to the controller by a data network. These sensors, actuators and controllers are computer nodes on the network having various computational powers.

Sensor nodes have the task of measuring one or multiple plant outputs by their analog to digital converters. Sensor nodes transmit the values that they measure over the network. Actuator nodes have the task of applying new values to the plant by means of digital to analog converters and suitable actuators. They receive these values over the network. Controller nodes use the plant outputs that they receive form sensor nodes to calculate control outputs by a control algorithm. Then, these control signals are sent to the actuator nodes to be applied to the plant. The data that travels over the network is encapsulated in a packet and, at the receiving side this packet is analyzed and the value is extracted. The physical layout of a NCS is shown in Figure 1-1.

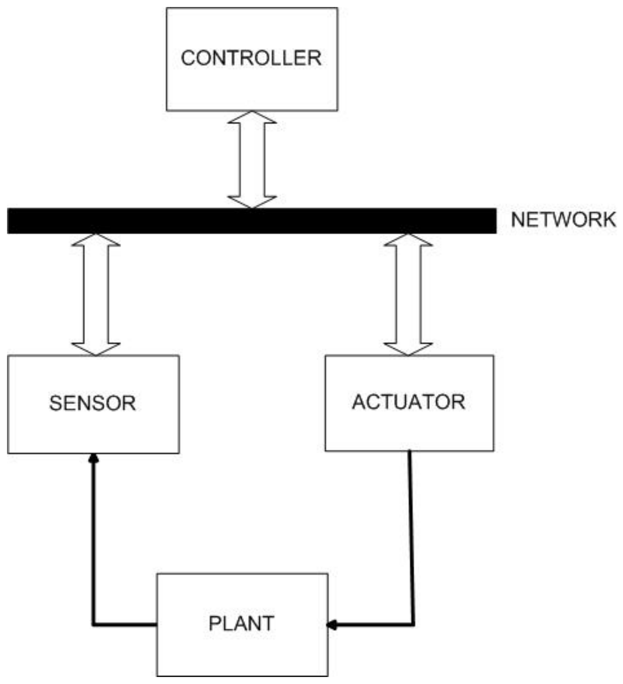


Figure 1-1 - Networked Control System

The primary advantages of a networked control system are:

1. **Increased system agility.**

All the nodes in a NCS are computerized components. And more importantly they all use the same physical network interface. This enables the addition of new components to the system without the need for changes to the existing hardware. Also components can be physically moved to different locations while staying on the same network.

2. **Ease of system diagnosis and maintenance**

Networked control systems are distributed systems. Since all nodes are physically separated, they can easily be replaced. Nodes can have software written inside them that detect problems or a more elaborate system may have special nodes that diagnose the system by only listening to the network, therefore such a node would not create any traffic on the network and would be totally invisible to all other nodes.

3. **Increased reliability due to reduced system wiring**

NCS's communicate over the network connections, in most cases this is in the form of a bus where a single cable runs along the plant and all the network nodes are connected to this. As every connection in a conventional control system is a potential point for breakdown, replacing these connections with a common network significantly reduces the amount of wiring in the system resulting in a simpler and more reliable system.

With the addition of a communication network in the feedback control loop, the complexity of analysis and design for a NCS increases. Bringing the necessity for revision of the control methods with assumptions of non-delayed sensing and actuation. There are two network induced varying time-delays in the control loop; one from the sensor node to the controller node and one from the controller node to the actuator node because it also takes some time to send messages over the network. This delay is dependent on the network scheduling policy and is not constant or bounded. The delay in some implementations may be nearly constant but in many cases it is random. The length of the transfer delay depends on network load, priorities of the ongoing communications, electrical disturbances and various other factors. There are essentially 3 kinds of delays in the systems.

1. Communication delay between the sensor node and the controller node that has occurred during sampling time t_k, $\tau_{sc}(t_k)$
2. Computation delay in the controller node that has occurred during sampling time t_k, $\tau_c(t_k)$
3. Communication delay between the controller node and the actuator node that has occurred during sampling time t_k, $\tau_{ca}(t_k)$

Note that more intelligent sensor nodes and actuator nodes also have some computational load therefore have some delays that can be expressed respectively as

$\tau_s\,(t_k)$ and $\tau_a\,(t_k)$, but these delays can be considered as fixed delays and the sensor node calculation delay can be included in the communication delay between the sensor node and controller node and the computation delay at the actuator node can be included in the communication delay between the controller node and the actuator node.

The total delay from the sensor node to the actuator node is the sum of the previously described delays:

$$\tau\,(\mathrm{t_k}) = \tau_{sc}(\mathrm{t_k}) + \tau_c(\mathrm{t_k}) + \tau_{ca}(\mathrm{t_k}) \tag{1}$$

The random nature of these delays makes the system delay varying and theoretical results for analysis and design for time-invariant systems can not be used directly.

A solution to this problem may be the introduction of buffers. If chosen larger than the maximum delay then the delays can be considered as fixed making analysis of the system deterministic. However this approach can introduce delays that may be unnecessarily large and can degrade the control performance.

Not only are the networked control systems subject to delays but infinite delay or packet loss may also occur. This may be due to physical disturbances on the system as well as collisions in the network caused by network loading. Therefore many systems have certain robustness against limited packet losses, and can assume that packets have a time limit for arrival and consider them lost if this limit is exceeded. Ling & Lemmon [1] have examined the effects of dropout probability on system performance via a comparative simulation so that it can be used in determining the expected performance on the system thus enabling the upper and lower bounds to be set for the system to function properly.

2 BACKGROUND

2.1 Research On Networked Control Systems

2.1.1 Performance Evaluation of Networked Control Systems

Initially work commenced with decentralized control of large scale systems. Important work has been done to synchronize country wide electricity grids with multiple power plants. [2],[3],[4]

Network theory and control theory have developed over the years and are now well established theories, however there has not been much interaction between them. For this reason, NCS has not been a natural result of this development. Much of network theory aims in increasing average throughput with little concern on latency of each packet of data. This implies batch processing where many units of data are first accumulated, then sent over the network at once, increasing efficiency. Lost packets are dealt with methods such as confirmation of reception (acknowledge) which, if not received within a specific amount of time, triggers a retransmission. Another problem is the stochastic nature of media access protocols that fail to guarantee an upper bound on packet transmission time.[5],[6],[7],[8]

In a NCS however, the stability of the controlled system depends on the timely transmission of *each sample* from sensor node to controller node and further to each actuator node. For this reason, network theory and technology developed for general applications is not suitable without modification for networked control systems[9].

Similarly, control theory assumes that implementations will be free of jitter, random delays and information loss along the control loop. No attention is given to the complexity of implementation; a powerful enough computing platform is assumed which is not trivial to implement.

A separation of concerns between the two communities can therefore be mentioned. For implementation of a complex control algorithm, this may be a serious handicap. With the recent trends in applications, it has become clear that a bridge must be formed to solve the above problems. Several methods have been recently developed with various assumptions and shortcomings, some of which will be explained next.

2.1.2 Co-design of control and communication

Considering the control and the network parts of a NCS separately is not an appropriate since they are not decoupled. High amount of data transfer between nodes reduces isolation, but increases traffic, generally also increasing dropped packets and thus communication delay. Therefore a suitable operating point must be found. An optimum combination has to be made with compromises on each side. Designing the network and control parts together aims to minimize a given cost function to ameliorate the performance. Goldsmith has worked with co-design of wireless networks.[5],[10]. Branicky Philips and Wei [11] have adapted rate monotonic scheduling algorithm for networked control systems. The effects of packet loss are examined and cost functions associated. As a result normally un-schedulable NCSs can be scheduled by deliberately over loading the system. Some packets are dropped but effects on system performance are insignificant.

2.1.3 Reduction of Communication

It is a well known fact that jitter and latency which reduce the quality of service (QoS) of a network increase as the network load increases. Therefore using systems that necessitate the least amount of network bandwidth facilitates the linear time invariant approximation of the network, hence facilitating the design process. In addition more systems can utilize the same network.[7].

2.1.3.1 Deadbands

One of the most effective means of reducing communication is to eliminate unnecessary communications. If consecutive packets contain identical or similar data, then only the first is sent and the receiving side assumes that the data is the same if there is no data packet arriving. This method assumes that the network does not corrupt or drop any packets. Otanez, Moyne and Tilbury[12] present a method to determine the size of the deadbands that relies on a performance metric that takes into account system response as well as network traffic. The effectiveness of deadband control with different controllers is studied as well as the effect of disturbances and plan uncertainty.

2.1.3.2 Estimators

Other means of reducing communication load include generating the necessary data on the receiving side of the network by the use of models. Where models of the components exist on each node, data generated from these nodes is used instead of data arriving from the network. The network is only used for synchronization of the models.

Yook, Tilbury, Wong and Soparkar [13]escribe a new framework where estimators are used at each node The nodes use the estimator's values to calculate the control algorithms. When the estimated value deviates from the true value by more than a pre-specified tolerance, the actual value of the state is broadcast to the rest of the system, all of the estimators are then updated to the current value. By computing the estimated values at every node, significant savings in the bandwidth is achieved, allowing large scale distributed control systems to be implemented effectively.

2.1.4 Network Observers

To eliminate the effect of the delay introduced by the network, an alternative approach is to consider this delay as a disturbance. A disturbance observer architecture is used to calculate this disturbance which is then is added to the control signal. This has an effect on the closed loop characteristics of the system similar to the smith predictor[14].

2.1.5 Gain Adaptation

It is possible that the control time delay caused by the network may change slowly depending on the traffic load and because of the prioritization algorithms used in the network, it may not be able to provide the same Quality of Service (QoS). The performance of the network directly affects the performance of the system. Should the QoS of the network change the NCS would have to gracefully degrade its performance and perform the task as well as possible. If the network delay changes, the quality of control deteriorates. Gain adaptive systems monitor network QoS measure the current delay in the network and recalculate the controller gains dynamically to function under the new loop delay conditions imposed by the network[15]

2.1.6 Networked Model Predictive Control

Model predictive control (MPC) has been used for a long time in control applications. To optimize some controlled variables a cost function is chosen. A model of the plant is then iterated with the control algorithm, several time steps into the future, and based on the outcome, the control output which will minimize a given cost function is chosen[16]. From this approach, networked predictive control approach is derived. In the networked predictive control approach there is a networked control predictor and a conventional model predictive controller. The networked control predictor compensates the network communication delay and the predictive controller controls the system. The network control predictor resides on the actuator node of the NCS and has the function of selecting the signal to be applied to the plant. The predictive controller uses the cost function and the model predictive control algorithm to calculate the control signals with the control horizon. Then, all the signals generated for the control horizon are sent to the actuator node. A direct connection between the sensor node and controller node and a-priori knowledge of the reference is assumed.

2.1.7 Co-Simulation of control and communication

The design process of networked control systems can be facilitated with the use of software to simulate these systems before they are implemented so that design citeria can be determined. There are many software that provide aid in such cases. Some of these software are AIDA, Jitterbut, Orcad, Ptolemy II, RTSIM, Syndex, Truetime and XILO details of these software may be found on[22],[6]. Software for co-simulation is introduced by Branicky, Liberatore and Philips in [18] This method is implemented on machining applications [19]

We will examine TrueTime software in detail as it is used in the simulations of this work.

TrueTime is a MATLAB/SIMULINK based tool used to simulate networked real-time systems. The tool gives complete freedom on the level of abstraction that the user wants to create. If desired, very low level instructions can also be used. The execution time of every instruction can also be incorporated, also the ability to assign execution times to code blocks is provided. The network can also be configured as desired. The code and algorithms developed under TrueTime can be directly ported to the actual implementation of a digital control systems. The type of scheduling performed by TrueTime kernels can be selected among the different types such as rate monotonic scheduling algorithm. TrueTime also makes it possible to simulate models of standard network protocols. Thus their influence on networked control loops can be measured.

TrueTime gives the ability to write C++ code, Matlab's 'm code' and use of Simulink blocks to implement the desired algorithms to be performed by network nodes at the kernel. The kernel blocks are event-driven and execute code that models input output tasks, control algorithms, network interfaces and various other tasks. Likewise, network messages are sent and received according to the chosen network model.

The detail of these simulations is also decided by the user. The execution of each piece of code can be simulated and the execution time and jitter can be explicitly expressed to the simulator, this enables the simulation of data dependent or random delays and losses. Events and monitors may be used to synchronize tasks, just like the actual implementation of a real-time computer system.

Digital to analog converters and analog to digital converters exist on the TrueTime kernel block. This enables the interfacing to the Simulink environment.[17]

2.2 What is Currently Lacking in NCS

None of the solutions described above is totally resilient against networks which have unbounded delay or networks that may cause packet loss. Dead bands may reduce communication and improve network conditions however they do not provide any solution to packet loss. Gain adaptation and network observers consider the changes in network to be relatively slow or the network to be symmetric (sensor node to controller node delay is same as controller node to actuator node delay). Some a-priori knowledge of the delay is assumed.

Model predictive controllers either do not take into account the synchronization between the nodes or they are not set up to be networked control systems, because they rely on a direct link between the sensor node and controller node. This means that both controller and sensor tasks reside within the same node of the network or not communicate at all over a network. The reason is that if the sensor node to controller node transmission fails then the basis for predictions is lost. The situation is even worse if in the previous period the controller node to actuator node control packet is lost, in which case the plant states are estimated using the wrong control signal applied to the model of the plant. Also a-priori knowledge of the reference signal is assumed in the model predictive control.

To address these problems, we propose a method that increases the performance of a basic NCS under variable time delays and packet loss. Standard NCS architecture is assumed as explained in Chapter 1 and no direct links or extra nodes are required, therefore the method can be applied to existing NCS's. A-priori knowledge of the reference signal is not assumed as this is not the case with most systems.

3 MODEL BASED PREDICTIVE CONTROL METHOD

Our aim is to render networked control systems resilient and stable. We propose to achieve this by holding a model of the plant within the controller and calculating current and the predicted control output to the plant for several time steps into the future. All of these outputs are then sent to the actuator node which applies the first to the plant. In case of data loss in the controller node to actuator node link, previously sent predictions are applied to the plant at each successive time step, hence the name model based predictive networked control system (MBPNCS).

The packet that arrives to the actuator node contains n+1 signals that are meant to be applied in the next n+1 sampling periods. The first is the control signal for the current sampling period and the rest are predicted control signals from n steps into the future. Should there be packet loss from the controller node to the actuator node, the actuator node uses the signal that is meant to be applied in the respective period. By doing so the actuator node has some meaningful signals that can be applied to the plant in the periods that control packets are not received form the controller node.

The generation of these signals is a computationally intensive task where a model is involved. Please refer to the outline in Figure 3-1. The initial control signal $u(t_k)$ is generated using the plant states $x(t_k)$ from the sensor node and the control algorithm. This signal is stored and also applied as input to the plant model $\hat{P}$. The plant model returns the estimated plant states of the following sampling period $\hat{x}(t_{k+1})$. Then the control algorithm is applied to these predicted plant states. This action generates an estimated control signal $\hat{u}(t_{k+1})$ which is intended to be applied to the plant at period t_{k+1}. This process is repeated n times until $\hat{u}(t_{k+n})$ is obtained. All of these control signals are inserted into a packet and sent over the network to the actuator node.

The proposed control system is composed of 5 parts:

A. Communication network

B. Sensor node

C. Actuator node

D. Controller node

E. Model of the controlled plant

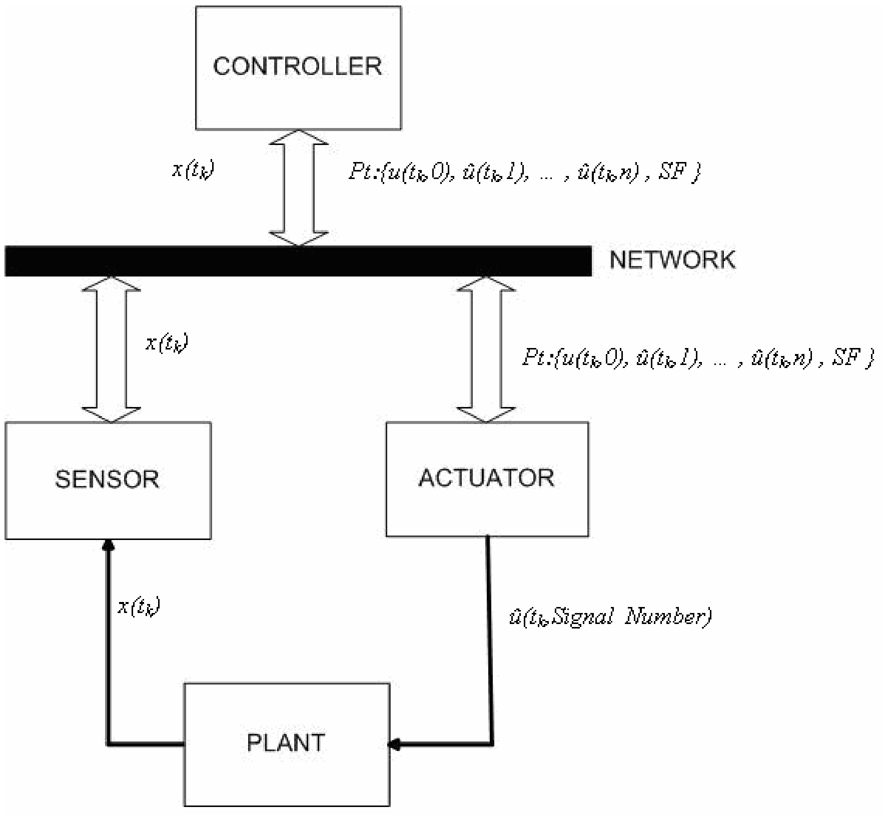

CONTROLLER
x(t_k)
Pt:{u(t_k,0), û(t_k,1), ... , û(t_k,n) , SF }
NETWORK
x(t_k)
Pt:{u(t_k,0), û(t_k,1), ... , û(t_k,n) , SF }
SENSOR
ACTUATOR
x(t_k)
û(t_k,Signal Number)
PLANT

Figure 3-1 - *System Setup*

These components can be located physically far from each other and are all computer systems with various degrees of computational capabilities. These components are equipped with network communication cards and thus each component communicates with the others over a network. Each component appears as a node in the network and shall be referred to as a node when addressed in the context of the network. The model of the controlled plant resides inside the controller node.

3.1 Definitions:

3.1.1 Communication network:

The network provides the communication between the nodes. It introduces a delay to message transmissions which is variable i.e. jittery. The network is also prone to packet losses If not compensated for, these effects will deteriorate stability.

Packet loss means that computed data does not arrive correctly on time. It is not an uncommon strategy in NCS to include late packets in the lost packet category since a late packet does not contain useful information. Transmission of new information is preferable to the retransmission of old information. In case of packet loss, there is no information arriving from the sensor node to the controller node or from the controller node to the actuator node.

In the MBPNCS architecture late packets are considered to be lost.

3.1.2 Sensor node:

The sensor node acquires the plant states as analog inputs at time t_k , $x(t_k)$,. Being an intelligent component of the system, it is configured to run a periodic task to acquire the necessary analog input and transmit it to the controller node at fixed time intervals.

3.1.3 Actuator node:

The actuator node is another intelligent component of the system and is also configured to run a periodic task. The function of the actuator node in the system is to apply the control signal '*u*' or prediction *û*' generated by the controller node to the plant. At the beginning of its period the presence of new control packets is checked. And the control signal to be used is selected from the control packet and applied to the plant.

3.1.4 Controller node:

The controller node, also an intelligent component of the system, is configured to run a periodic task. At the beginning of every period it receives the plant states from the sensor node. A control signal *u(t*$_k$*)* is calculated using the control algorithm and plant states then sent to the actuator node using the network. In this study this basic networked control system approach is extended by the use of a model to be resilient against packet loss, described below.

3.1.5 Model of controlled plant

The state space equations of the plant (2) are used to derive the plant model $\hat{P}$. The model of the plant represents the plant and is used for generating predicted states of the plant. It is reinitialized whenever new plant states arrive from the sensor node. The plant model is obtained by discretizing the plant state space representation with the same sampling time as the networked control system.

$$\begin{aligned} \dot{x}(t) &= Fx(t) + Gu(t) \\ y(t) &= Hx(t) + Ju(t) \end{aligned} \tag{2}$$

The system equation describes a non-homogeneous set of coupled linear differential equations with a solution

$$x(t) = e^{F(t-t_0)}x(t_0) + \int_{t_0}^{o} e^{F(t-\tau)}Gu(\tau)d\tau \tag{3}$$

Note that the first term in the above equation is the homogenous solution and the second term is the particular solution (convolution of the input with the system's impulse response). Therefore we can make t$_0$ = kT and look at the state vector T seconds later,

$$x(kT+T) = e^{FT}x(kT) + \int_{kT}^{kT+T} e^{F(kT+T-\tau)}Gu(\tau)d\tau \tag{4}$$

Since a zero order hold (ZOH) circuit holds the control constant over an entire sample period, we can move the control input *u(τ)* and the input matrix G out of the integration. Then, after changing integration variables and simplifying we can write,

$$x(k+1) = e^{FT}x(k) + \int_0^T e^{F\eta} d\eta \cdot Gu(k) \qquad (5)$$

$$= \Phi x(k) + \Gamma u(k)$$

Where,

$$\Phi = e^{FT} = I + FT + (FT)^2/2! + (FT)^3/3! + \ldots \qquad (6)$$

$$\Gamma = \int_0^T e^{F\eta} d\eta \cdot G$$

If we write,

$$\Phi = I + FT\Psi \qquad (7)$$

Where

$$\Psi = I + FT/2! + (FT)^2/3! + \ldots \qquad (8)$$

then,

$$\Gamma = \int_0^T e^{F\eta} d\eta \cdot G = F^{-1} e^{F\eta}\Big|_0^T G = F^{-1}(\Phi - I)G = T\Psi G \qquad (9)$$

So the discrete equivalent state space model becomes,

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= Hx(k) \end{aligned} \qquad (10)$$

3.2 Control Node Structure:

The controller node is equipped with a model of the plant $\hat{P}$. Plant states *x* from the sensor node are used to reinitialize the model at the beginning of every period whenever they are available. This limits deviation of the sates of the model from those of the actual plant. At time t_k the control signal $u(t_k,0)$ applied to the plant at time t_k is applied to the plant model $\hat{P}$. The output of the model is an estimated state of the plant $\hat{x}(t_{k+1})$ at time t_{k+1}. To calculate the next signal in the packet to be applied at time t_{k+1}, the control algorithm is applied to the estimated states $\hat{x}(t_{k+1})$, providing the control signal $\hat{u}(t_k,1)$ that should be applied to the plant in the sample period after $u(t_k,0)$ is applied. Since this signal is an estimated signal it will be referred to as $\hat{u}(t_k,1)$.

The control signals are noted as:

û(generation time, prediction number)

where a prediction number 0 indicates that the control signal is meant to be applied to the plant when received. $\hat{u}(t_k,2)$ would be the 3rd signal in a control packet generated at time t_k this signal would be applied at time t_{k+2} should there be controller node to actuator node control packet loss at times t_{k+1} and t_{k+2}.

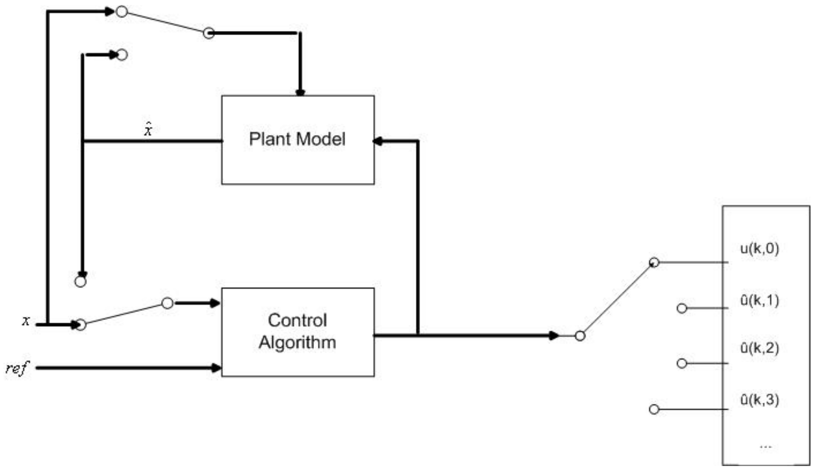


Figure 3-2 Controller Schematic

This process is recursively applied n times until all predicted control signals $u(t_k,0)$, $\hat{u}(t_k,1)$ ... $\hat{u}(t_k,n)$ are obtained at time t_k.

The result of this process is a control sequence containing an array of n+1 signals based on the plant model, current states of the plant and reference input, generated at time t_k, spanning until time t_{k+n}. The packet $Pt(t_k)$ therefore consists of $\{u(t_k,0), \hat{u}(t_k,1), \ldots, \hat{u}(t_k,n)\}$

The number of predictions is chosen based on the following factors:

1) The accuracy of the model should be considered

A model is used to produce the control signals following the first signal in every packet. Since models have inaccuracies, as the horizon broadens the predictions deviate from the actual. Due to this factor it is unreasonable to make predictions beyond a certain limit depending on the accuracy of the model and the disturbances in the system.

2) The packet size should not decrease network quality of service.

More time is needed to transmit more information. If packet size is sufficiently large, the packet will not have enough time to be transmitted over the network in the necessary time interval. Also a larger packet will take more of the network's bandwidth, which gives less bandwidth for the other nodes or other systems on the network.

3) Processing power available

Running a model requires computational power. Making more predictions will take more time, and the packet transmission deadline can be missed.

3.3 Problems with MBPNCS and solutions

Imagine a basic NCS like the one described in Chapter 1. All nodes are configured to run periodic tasks. The sensor node receives the plant states, transmits these states to the controller node which calculates the control signal using the control algorithm and transmits it to the actuator node where it is applied to the plant.

In such a NCS should there be packet loss from the sensor node to the controller node the controller node would not be able to calculate a control signal and there will again be no signal sent to the actuator node. Should there be data loss from the controller node to the actuator node there will be no signal to apply at the actuator node. The usual strategy in these cases is to apply to the plant the last signal received by the actuator node. In both cases correct control is not applied to the plant and stability is jeopardized.

3.3.1 Resolution of the Sensor node to Controller node data loss:

In our method, should such data loss occur, the missing data will be generated at the controller node using the model. The last transmitted signal $u(t_k,0)$ is applied to $\hat{P}$, and the generated states $\hat{x}\,(t_k+1)$ are used instead of the ones from the sensor node.

3.3.2 Resolution of the Controller node to Actuator node data loss:

The controller node uses $\hat{P}$ to generate control signals that span in to the future. If the actuator node does not receive data from the controller node, it uses the signal previously received from the controller node $\hat{u}(t_k,1)$ instead of $u(t_k+1,0)$ which was destined to be applied in this period but does not arrive on time. If further consecutive packets from controller node to actuator node are dropped, signals $\hat{u}(t_k,2)$, $\hat{u}(t_k,3)$... $\hat{u}(t_k,n)$ will be applied at time t_{k+2} , t_{k+3} ... t_{k+n}, as appropriate.

3.3.3 Problems Introduced with the basic model based predictive approach:

In some special cases this solution introduces problems; if the sensor node to controller node data loss and controller node to actuator node data loss occur in a specific order, some synchronization problems appear i.e. the wrong control signal is applied to the plant model to bring it up to date.

Consider the following case as an example:

At time t_k: The sensor node obtains the states of the plant $x(t_k)$ and transmits them to the controller node which calculates an array of control signals $\{u(t_k,0),\ \hat{u}\ (t_k,1)\ ...\}$ sends a packet $Pt(t_k)$ to the actuator node within (t_k- t_{k+1})the packet is transmitted on time to the actuator node which retrieves $u(t_k,0)$ from this packet and applies it to the plant.

At time t_{k+1}: The sensor node obtains the states of the plant $x(t_{k+1})$ and transmits them to the controller node which calculates an array of control signals $\{u(t_{k+1},0),\ \hat{u}$

$(t_{k+1},1)$ ...} and sends the packet $Pt(t_{k+1})$ to the actuator node within (t_{k+1}, t_{k+2})but there is an error during transmission and the packet is lost. The actuator node uses the previous packet and applies the estimated control signal $\hat{u}(t_k,1)$ to the plant.

At time t_{k+2}: The sensor node obtains the states of the plant and sends them to the controller node but the packet is lost. The controller node uses $u(t_{k+1},0)$ and $x(t_{k+1})$ to estimate the states of the plant and generates an array of control signals $\{u(t_{k+2},0), \hat{u}(t_{k+2},1) \ldots\}$and sends them as a packet $Pt(t_{k+2})$ to the actuator node, the packet is transmitted on time to the actuator node which retrieves $u(t_{k+2},0)$ from this packet and applies it to the plant.

However this is not a correct control signal since it was obtained using the control algorithm applied to $\hat{x}(t_{k+1})$ obtained from $\hat{P}$ that was updated using $u(t_{k+1},0)$ whereas the plant was actually subject to $\hat{u}(t_k,1)$ which in general are not the same.

3.3.4 Solution to the Synchronization problem:

Since there is no data sent back from the controller node to the sensor node or the actuator node to the controller node, solution of the information loss problems are left to the receiving nodes: Sensor node to controller node data loss is resolved by the controller node and controller node to actuator node data loss is resolved by the actuator node as explained below:

3.3.5 Sensor node to Controller node data loss:

Should there be sensor node to controller node data loss the controller node solves this problem by calculating $\hat{x}(t_k)$ using $\hat{P}$ and $u(t_k,0)$ from the last packet sent. The controller node assumes that the last packet was successfully received by the actuator node. However as this affects the rest of the synchronization process, the actuator node must be informed of such an event. Therefore the data packet contains a separate sensor flag *SF* that indicates if the control data was generated using actual states from the sensor node or estimated using the model. If the control signals are generated using the states arriving from the sensor node the sensor flag takes the value of 1 which is also expressed as high, else if the signals are generated using estimates SF is set to 0 which is also expressed as low. Another problem is the calculation of the time dependent variables of the plant. Should the controller node run on the model after a few iterations the $x(t_k)$ and $\hat{x}(t_k)$ will differ. Calculating the derivative term using $x(t_k)$ and $x(t_{k+1})$ or $\hat{x}(t_k)$ and $\hat{x}(t_{k+1})$ will give consistent results, however calculating the derivative term using $\hat{x}(t_k)$ and x (t_{k+1}) or vice versa will not give correct results since $\hat{x}(t_k)$ and $x(t_k)$ are different even in the most ideal system running on a network with packet loss.

To resolve this problem the control algorithm was distributed. The task of calculating all the terms is separated from the control algorithm and assigned to the sensor node. The sensor node calculates the derivative, integral and all other terms and transmits these to the controller node. Assuming that when a data packet arrives from the sensor node, the controller states can also be recalculated to reset any synchronization problem between the controller and sensor nodes.

3.3.6 Controller node to Actuator node data loss:

The actuator node has two modes. The first is the “normal mode” where it functions as previously described in section 3.2 if there has not been any controller node to actuator node data loss. The second is an “interrupted mode”. Should there be controller node to actuator node data loss the actuator node enters the interrupted mode. Where it uses only control signal of the last packet that it has received, i.e. only *Signal Number* increases in $\hat{u}(t_k, Signal\ Number)$ with time. The actuator node returns to normal mode only when it receives a data packet that has a high sensor flag SF. Detailed behavior of the actuator node is explained below. Packet is redesigned as $Pt{:}\{u(t_k,0), \hat{u}(t_k,1), \dots, \hat{u}(t_k,n), SF\}$

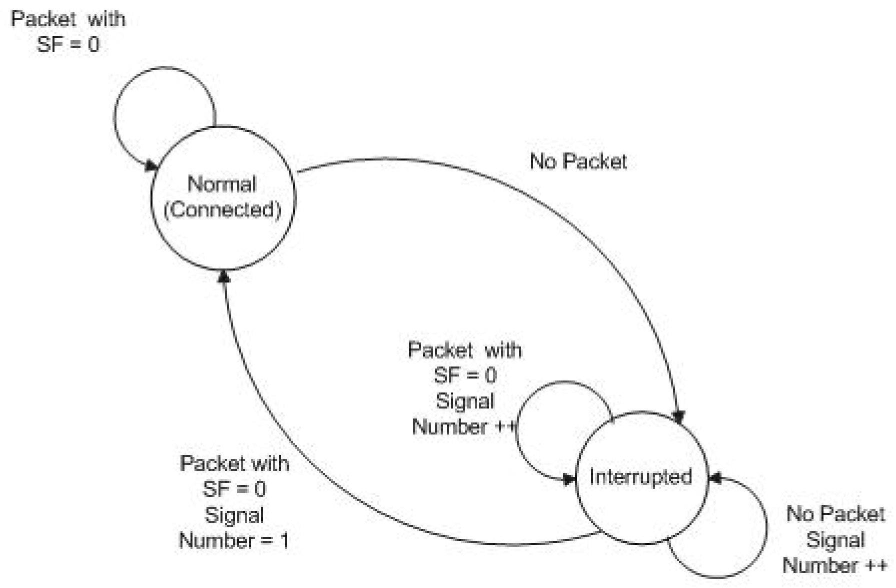

Packet with SF = 0
Normal (Connected)
No Packet
Packet with SF = 0 Signal Number ++
Interrupted
Packet with SF = 0 Signal Number = 1
No Packet Signal Number ++

Figure 3-3 - State diagram of actuator node where "++" means increment

3.3.7 Actuator node State Transition Rules:

1) If no packet is received the actuator node enters an "Interrupted" mode, indicating that there has been an interruption in the controller node synchronization.

2) If the packet from the sensor node is timely and the sensor flag high, the actuator node enters the "Normal" mode, the first signal of the packet *$u(t_k,0)$* is used.

3) While the actuator node is in interrupted mode, all consecutive packets with a low sensor flag are ignored even if they arrive from the controller node on time.

Example :

Period #	Transmission interval	Sensor Flag	Received at actuator node	Signal used at actuator node
0	$t_k - t_{k+1}$	1	OK	$u(t_k,0)$
1	$t_{k+1} - t_{k+2}$	0	OK	$u(t_{k+1},0)$
2	$t_{k+2} - t_{k+3}$	0	OK	$u(t_{k+2},0)$
3	$t_{k+3} - t_{k+4}$	1	OK	$u(t_{k+3},0)$
4	$t_{k+4} - t_{k+5}$	1	NO	$u(t_{k+3},1)$

5	t_{k+5} - t_{k+6}	0	OK	$u(t_{k+3},2)$
6	t_{k+6} - t_{k+7}	1	NO	$u(t_{k+3},3)$
7	t_{k+7} - t_{k+8}	1	OK	$u(t_{k+7},0)$

0. At the first period controller node has received data from the sensor node and it transmits control output package $Pt(t_k)$ to the actuator node which receives the data and retrieves the first control signal $u(t_k,0)$ and this is applied to the plant.

1. In the 1st period there occurs data loss from the sensor node to the controller node, the controller node uses $\hat{x}(t_{k+1})$ to generate the states, calculates the control signals and transmits to the actuator node. There is no data loss and the actuator node retrieves the first signal in the packet and applies this to the plant.

2. At period 2 once again data loss from the sensor node to the controller node occurs, the controller node uses $\hat{x}(t_{k+2})$ to generate the states, calculates the control signals and transmits to the actuator node. There is no data loss and the actuator node retrieves the first signal in the packet and applies is to the plant.

3. At period 3 the sensor node sends the plant states to the controller node and no data loss occurs. $\hat{P}$'s states are updated for the first time in 2 periods. The actual states are used to calculate $u(t_{k+3},0)$ and $\hat{P}$ is used to calculate the control signals up to $u(t_{k+3},n)$ which are transmitted to the actuator node without loss. The actuator node applies $u(t_{k+3},0)$ to the plant.

4. At period 4 sensor data is received at the controller node without loss, the control signals are calculated but there occurs data loss while transmitting to the actuator node. The actuator node enters an interrupted mode and $u(t_{k+3},1)$ is applied to the plant.

5. At period 5 the sensor node sends the plant states to the controller node and no data loss occurs. The controller node calculates the signals assuming that the last packet was received by the actuator node, this causes $\hat{x}(t_{k+5})$ to deviate from $x(t_{k+5})$. The data packet is sent to the actuator node. The actuator node ignores this packet because it is in interrupted mode and the packet has a sensor flag set to low, and $u(t_{k+3},2)$ is applied to the plant.

6. At period 6 the sensor node transmits to the controller node, which in turn calculates the control signals but there occurs data loss while transmitting to the actuator node therefore the actuator node uses $u(t_{k+3},3)$.

7. At period 7 the sensor node transmits the plant states to the controller node without loss. The controller node uses these states to update $\hat{x}(t_{k+7})$. Then the control signals are generated, sensor flag is set to high, and the packet is transmitted to the actuator node. The actuator node receives the packet and reenters normal mode, upon which $u(t_{k+7},0)$ is applied to the plant.

3.3.8 Late, lost and multiple packets:

A late packet is not taken in to account and is discarded. Only the latest, newest of multiple packets is taken in to account.

4 RESULTS

The performance of the proposed MBPNCS was evaluated using a typical plant. In this chapter the results obtained will be shown. First the plant will be introduced, then the properties of a basic networked control system are investigated. Next the performance of the proposed MBPNCS are examined under various conditions of network packet loss and noise. comparisons are made with the basic NCS approach

4.1 Performance of Reference Systems

4.1.1 Continuous Non-Networked Control

The DC-servo described by the following transfer function is used as the system plant.

$$G(s) = \frac{1000}{s(s+1)} \tag{11}$$

A PD controller is implemented according to the following equations;

$$\begin{aligned} KP(k) &= K.(r(k) - y(k)) \\ KD(k) &= a_d KD(k-1) + b_d (y(k-1) - y(k)) \\ a_d &= \frac{T_d}{N_h + T_d} \\ b_d &= \frac{NKT_d}{N_h + T_d} \\ u(k) &= KP(k) + KD(k) \end{aligned} \tag{12}$$

Where $r(t_k)$, $y(t_k)$, $u(t_k)$, $KP(t_k)$ $KD(t_k)$ and K are the reference, plant output, control signal, proportional component of control signal, derivative component of control signal, proportional gain respectively and T_d is the sampling time. N, N_h are constants. The $y(t_k)$ value is obtained by $H*x(t_k)$ where H is the discrete state space matrix of the plant and $x(t_k)$ are the plant states at time t_k.

Initially the plant was controlled under a normal continuous setup. This concluded that the plant could be controlled by the control algorithm. Also rise time and settling times were determined. The setup was implemented under Matlab (Figure 4-1).

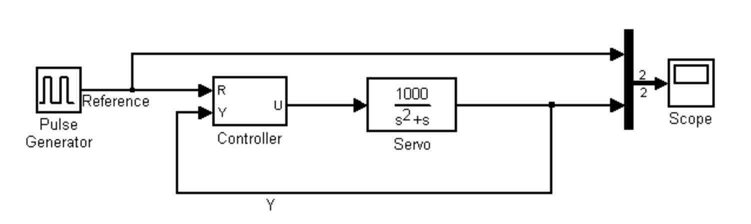


Figure 4-1 – Matlab Simulation Setup

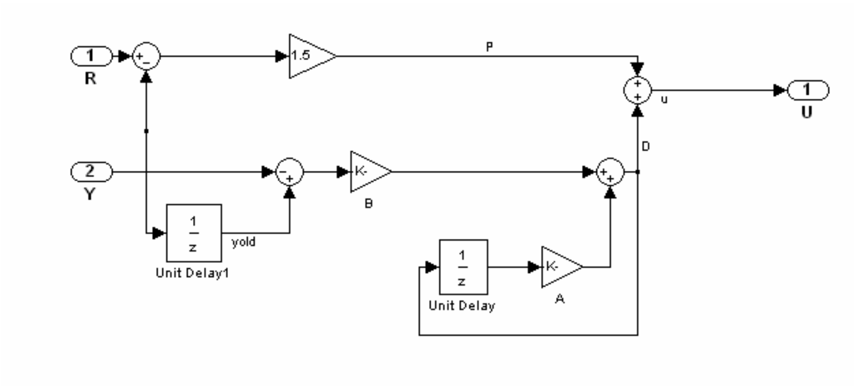


Figure 4-2 Simulink implementation of control algorithm

The reference to the system is a periodic pulse with the following properties:

Amplitude	1
Period (secs)	1
Pulse Width (% Period)	50

Table 4-1 Pulse Reference Characteristics

The simulation was run for 4 seconds, Figure 4-3 shows the response of the system.

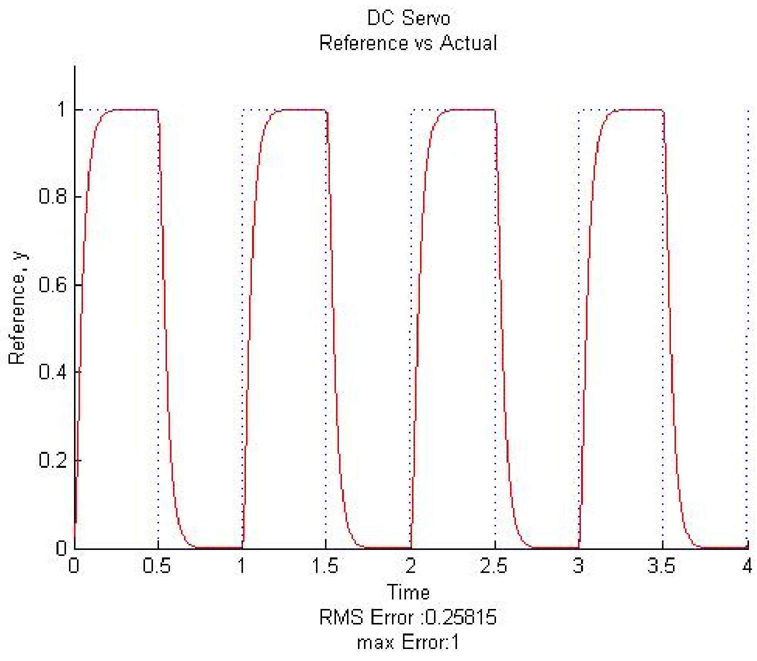


Figure 4-3 – DC Servo Continuous Time Simulation Result

The following information was extracted from this simulation:

Rise time:	0.105 seconds
Settling Time:	0.188 seconds
RMS Error:	0.258
Max Error:	1

Table 4-2 Continuous plant performance

Since NCS introduce random delay, quantitative measurement of system performance becomes difficult. Due to delay and packet loss, the response of the plant may not be a dampened oscillation, which means that standard performance measures such as overshoot cannot be used. For this reason two measures are introduced: RMS error and max error.

RMS error is calculated as the root mean square of the reference minus y i.e. error of the system response. It aims to show the total error created during the execution of the system.

Max error is the maximum value that the error signal reaches during the simulation. It aims to provide insight on the state of the plant when the reference changes or of large errors that can be considered as instability.

One may observe that RMS error is not 0, this is because it takes time for the plant to follow the reference, and during the time it takes for the plant to reach the reference there is an error which reflects to the RMS error value. Maximum error is 1 because 1 is the maximum reference change and at the precise moments of this change the plant has not had time to respond therefore the error is 1 at these moments, since the plant has time to settle, the error is equal to 1 once again when the reference changes. Changes in this value will be observed when the system goes unstable or when the system has not had time to settle before the reference change.

4.1.2 Basic Networked Control System Implementation

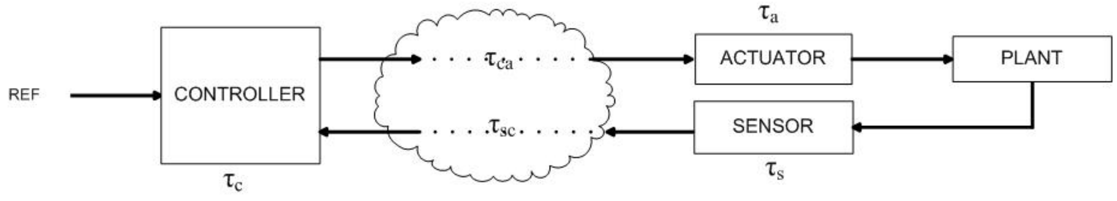
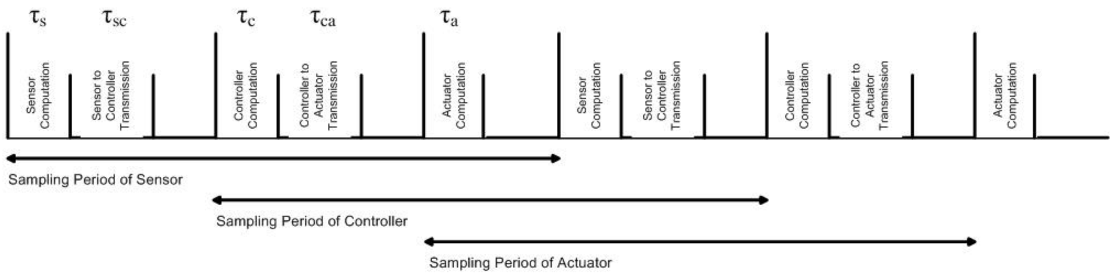


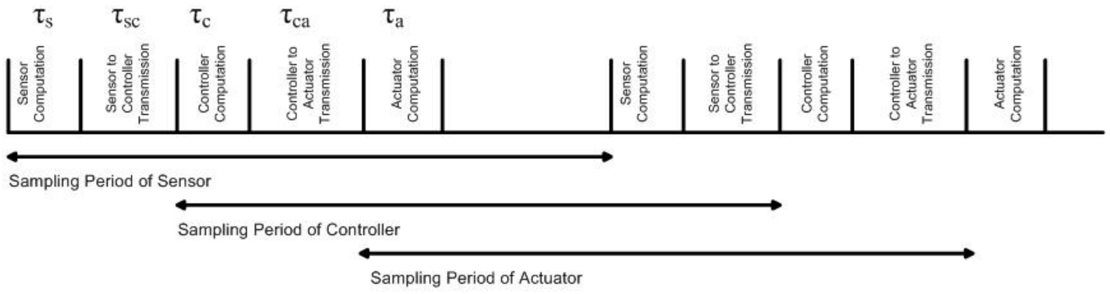
Figure 4-4 NCS Setup

The Control system is implemented as a basic Networked Control System (bNCS) (Figure 4-4) environment. The purpose of this experiment is to demonstrate that the plant can be controlled under an NCS. Architecture wise the NCS is no different than a basic NCS where all tasks are periodic. In the Figure 4-5 the periods are dictated by the internal clocks of the nodes, therefore this setup is always periodic independent of network status, however this does not indicate that the messages are transmitted or delivered in a periodic manner. In the experimental setup however an event driven approach is adopted which is a more common methodology. In the event driven approach only the sensor node runs a periodic task. All other nodes run tasks that are event driven; the controller and actuator node tasks are notified on the event of a new message arriving from the network. For the controller node this is a data packet arriving from the sensor node and for the actuator node this is a control packet arriving from the controller node (Figure 4-6).. The difference between the event driven and time driven approach under ideal conditions is the offset between the nodes is different, with the event driven approach this offset is minimal. Figure 4-5 and Figure 4-6



τ_s
τ_{sc}
τ_c
τ_{ca}
τ_a
Sensor Computation
Sensor to Controller Transmission
Controller Computation
Controller to Actuator Transmission
Actuator Computation
Sampling Period of Sensor
Sampling Period of Controller
Sampling Period of Actuator

Figure 4-5 Timeline of time driven basic NCS Schematic



τ_s
τ_{sc}
τ_c
τ_{ca}
τ_a
Sensor Computation
Sensor to Controller Transmission
Controller Computation
Controller to Actuator Transmission
Actuator Computation
Sampling Period of Sensor
Sampling Period of Controller
Sampling Period of Actuator

Figure 4-6 - Timeline of event driven basic NCS under ideal network

For the initial trials this basic NCS approach was used with an ideal network where there were no packet losses and the delivery times of the packets were fixed. The resulting system can be considered as a system where all the nodes run a periodic task. The sensor node sends the data packet to the controller node at fixed intervals and thanks to the ideal network the packets arrive with a fixed delay, the controller node sends a control packet to the actuator node and again the packet arrives to the actuator node with fixed delay (assuming fixed computational delay). Since the events that trigger the controller and actuator node tasks happen periodically these tasks are equivalent to periodic tasks. Figure 4-6 depicts the timeline of events. Remember that the sensor task running on the sensor node is periodic, since τs τsc τc τca τa are all equal all tasks run at the same period which is equal to the sensor tasks period. In the event driven NCS setup should there be packet loss in the system the receiving node is not woken up. If there is controller node to actuator node packet loss then the actuator node the last control signal received by the actuator node is applied until a new signal arrives. Should there be sensor node to controller node data loss the controller task running in the controller node does not wake up. Therefore a control packet is not sent to the actuator node which continues to apply the last signal it has received. In this work the basic NCS in implemented with an event driven approach.

The sampling time of the sensor node is set to be 0.001 seconds, therefore the controller and actuator nodes can also be considered as periodic tasks with periods of 0.001 seconds.

TrueTime model developed under Matlab and configured for the NCS specifications above is shown in (Figure 4-7).

In the networked control system environment system performance is similar to continuous non-networked control system. (Figure 4-8)

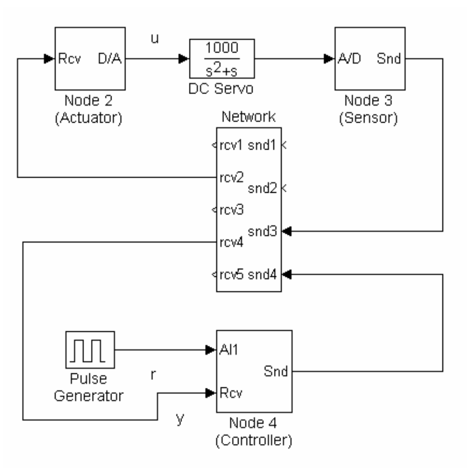


Figure 4-7 Matlab basic NCS Setup

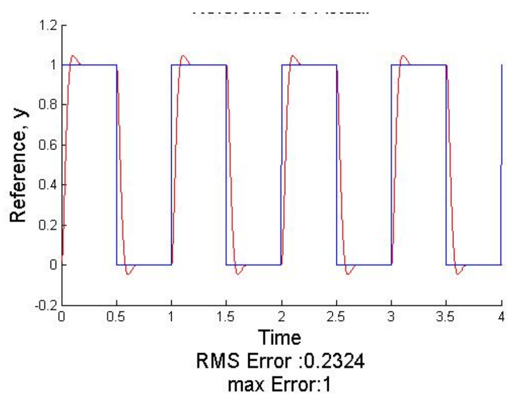


Figure 4-8- NCS Reference. Response of basic NCS under ideal conditions

The following information was extracted from this simulation:

Rise time:	0.105 seconds
Settling Time:	0.188 seconds
RMS Error:	0.2324
Max Error:	1

Table 4-3 Ideal NCS results

As this simulation runs under ideal conditions, we will take these values as the ideal or default values. Systems control quality will be compared to these results in the following work. Note however that these results are for a pulse reference as described in Table 4-1 and a simulation time of 4 seconds. Simulation reference and duration specific ideal values will be provided as the need arises.

4.2 System Performance Comparison Setups

The performance of the proposed model based predictive networked control system setup and the basic networked control system setup were examined under non ideal conditions.

The model based predictive NCS are described below.

4.2.1 Model Based Predictive NCS (MBPNCS) Controller Setup

As the name states, the MBPNCS uses a model. The formation of the model was described in section 3.1.5. The plant (11) is transformed to continuous state space.

$$\dot{x}(t) = Fx(t) + Gu(t)$$
$$y(t) = Hx(t) + Ju(t) \qquad (13)$$

Where

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 32 \\ 0 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 31.25 \end{bmatrix} \quad J = \begin{bmatrix} 0 \end{bmatrix} \qquad (14)$$

This state space representation is discretized with a sampling time of 0.001 seconds giving the following discrete state space representation.

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = Hx(k) \qquad (15)$$

Where:

$$\Phi = \begin{bmatrix} 0.999 & 0 \\ 0.0009995 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 0.03198 \\ 1.599e-5 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 31.25 \end{bmatrix} \qquad (16)$$

The model based predictive NCS controller setup is no different than the basic NCS (Figure 4-4 NCS Setup) setup. The only change is in the control algorithm and the software implementation of the controller and actuator nodes. All the nodes of the system are configured to run periodic tasks at frequencies of 1 kHz. All have offsets of 0.0001 seconds along the logical execution order i.e. 0.0001 seconds between the sensor node and the controller node and 0.0001 seconds between the controller node and the actuator node. This is to ensure that the network has time to deliver the data packets between the nodes. (Figure 4-9)

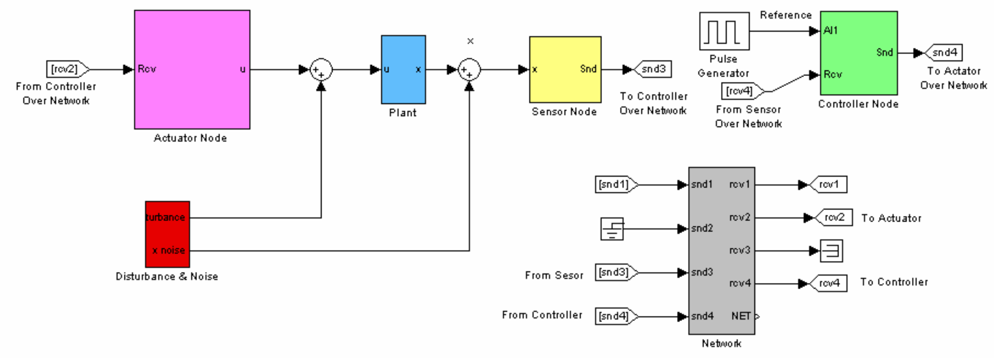


Figure 4-9 - Matlab Model Based Predictive NCS Setup

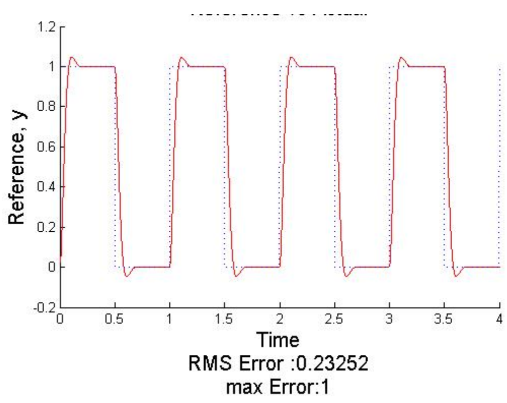


Figure 4-10 - MBPNCS ideal condition output

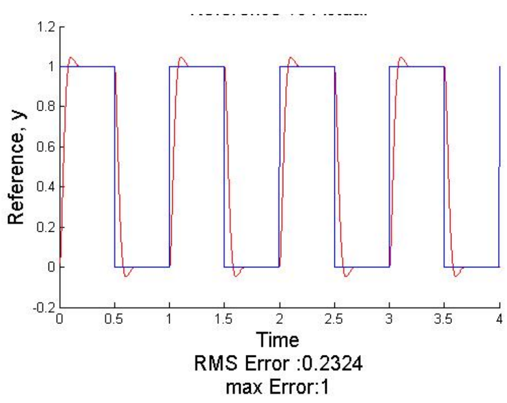


Figure 4-11 – Basic NCS ideal condition output

RMS Error: 0.23252

Max Error: 1

RMS Error: 0.2324

Max Error: 1

The MBPNCS and the basic NCS were both simulated under ideal network conditions. Both systems were run for 4 seconds with a reference as described in Table 4-1 . Both systems behave in the same manner. And have respective RMS error values of 0.23252 and 2324 which an be considered identical. The difference between the RMS error values exists because of the different offset between the network nodes.. The model based predictive networked control system setup can also be considered as a basic NCS under ideal conditions. The only difference between the two systems is the time offset between the nodes. This is 0.0001 in the MBPNCS and a fixed network transmission time in the basic NCS.

4.2.2 Effects of Various Disturbances Acting on the System

The effects of packet loss, noise and the performance of the system under low sampling frequencies is examined.

4.2.3 Packet Loss

Packet loss is a packet getting lost or corrupted in the network. Also in our controller methodology late packets are considered lost and they are discarded. In the experimentation models packet loss is actually packet corruption. By packet corruption it should be understood that a packet is sent from one node to the other and received by the receiving node but then is discarded because its contents cannot be identified. This has two effects on our model: 1. As a missing packet to the receiving node and 2. It occupies the network resource during its transmission.

Figure 4-12 shows the reference, the plants response to the reference. Also the signal count is shown. This is to show for how many sampling periods the actuator node is left to rely on predictions. Also the sensor to controller and controller to actuator packet loss may be observed to understand the controller to actuator synchronization.

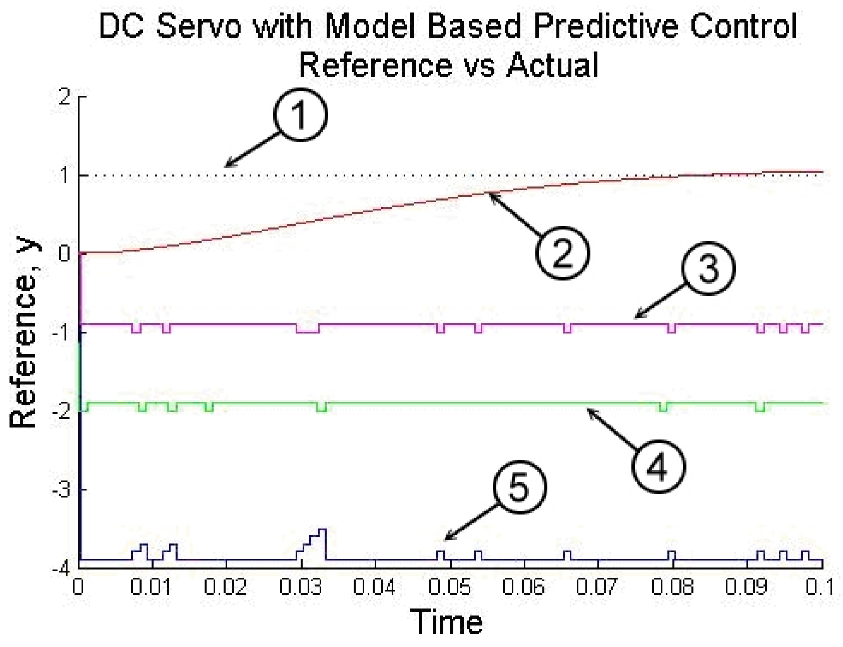


Figure 4-12 - Signal Count

1	**Reference**
2	**Plant output (y)**
3	**Controller node To Actuator node Packet Loss**
4	**Sensor node to actuator node packet loss**
5	**Signal Number in Valid Control Packet**

Table 4-4 – Signals

Reference: Is the reference signal applied to the controller node

Plant output: Is the output of the plant.

Controller node to actuator node packet loss: Indicates weather the packet sent form the controller node to the actuator node is lost. This signal is offset by -1 purely for the purpose of clarity. Its high state and low state should be taken in to account. While the signal is high, the packet has been successfully received, if it is low the packet has been lost.

Sensor node to actuator node Packet Loss: Indicates weather the packet sent form the sensor node to the controller node is lost. This signal is offset by -2 purely for the purpose of clarity. Its high state and low state should be taken in to account. While the signal is high, the packet has been successfully received, if it is low the packet has been lost.

Signal number in valid control packet: Which signal in the last received packet is applied at the actuator node. If the packet is valid then the first signal is applied, should packet loss occur the value of this signal increases. This value has been offset by -4 for the purpose of clarity. The value increases by 0.1 with every signal $u(t_k,\mathit{Signal\ Number})$ used form the packet as *Signal Number* increases. Should a valid packet arrive then the first signal in that control packet is applied as explained in chapter 3.2 this can be observed as a drop to -4 of this signal.

4.2.3.1 Comparative Performance Under Packet Loss

As packet loss increases degradation in performance is observable with the increasing RMS errors. A series of graphs comparing the effects of increasing amounts of packet loss on BNCS and MBPNCS are given in Figure 4-13 through Figure 4-14

Both systems experience some degradation in control quality as packet loss increases however the degradation in the basic NCS is much faster. At %60 packet loss this is very clearly observable Figure 4-25 & Figure 4-26.

At %70 packet loss the basic NCS loses its stability whereas the model based predictive NCS remains controllable with respective RMS errors of 74.4879 and 0.27 respectively Figure 4-28 & Figure 4-27.

RMS error increases as the packet loss probability increases. This increase in RMS is however not entirely related to degradation in control performance. Figure 4-31 clearly shows that when the system is subject to high packet loss then the reference signal is also not transmitted. Without communication between nodes it is impossible for the reference to take effect. Note that our system does not have a-priori knowledge of the reference signal in contrast to other research such as[21]. The retardation effect is amplified even more so by the selection algorithm on the actuator node side, this retardation effect is clearly visible in Figure 4-31 at time:1 seconds. The actuator node does not accept control packets if the synchronization is broken and the control packet does not have a high sensor flag which is a common occurrence at high packet loss rates.

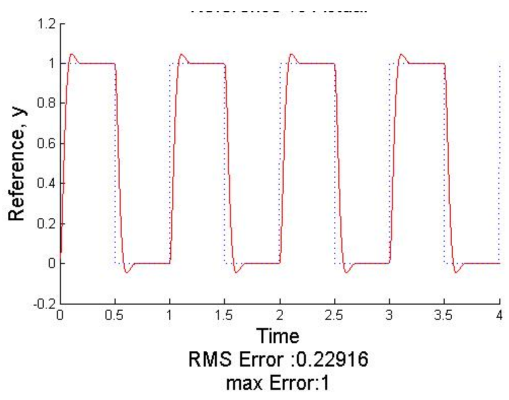


Figure 4-13 - %0 Packet Loss MBPNCS

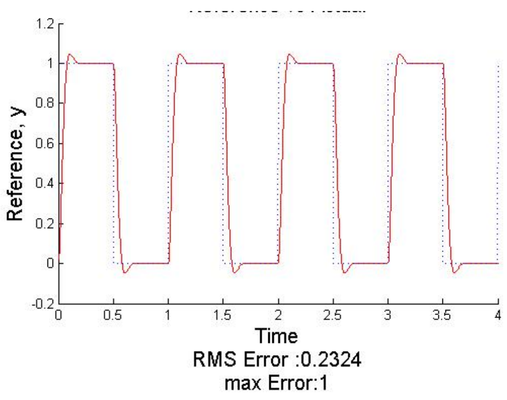


Figure 4-14 - %0 Packet Loss Basic NCS

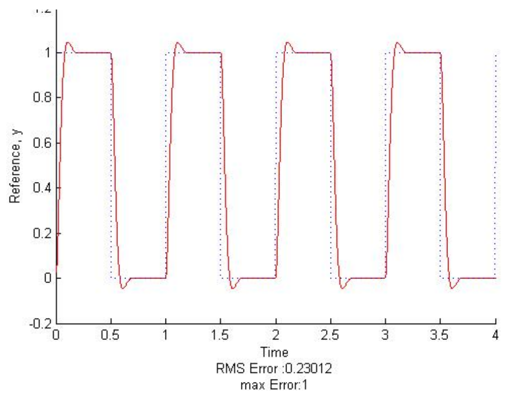


Figure 4-15 %10 Packet Loss MBPNCS

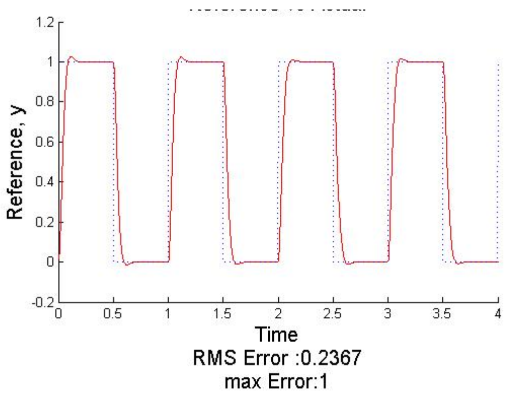


Figure 4-16 - %10 Packet Loss Basic NCS

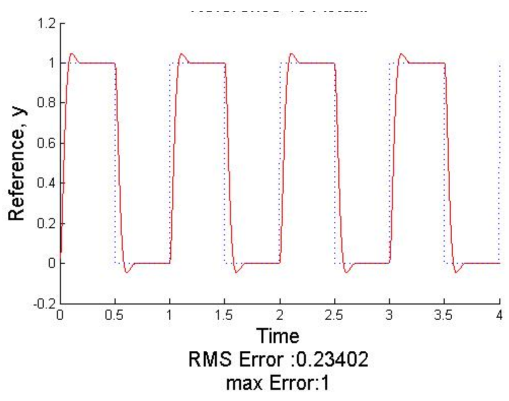


Figure 4-17 - %20 Packet Loss MBPNCS

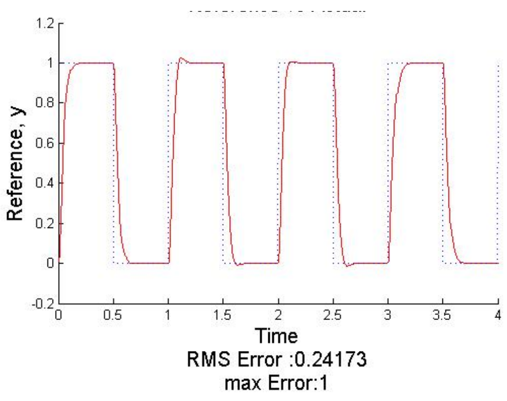


Figure 4-18 - %20 Packet Loss Basic NCS

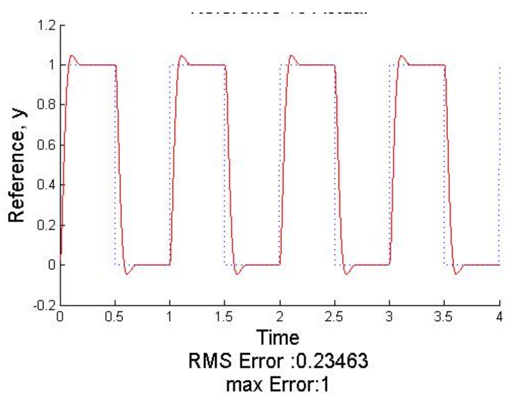


Figure 4-19 %30 Packet Loss MBPNCS

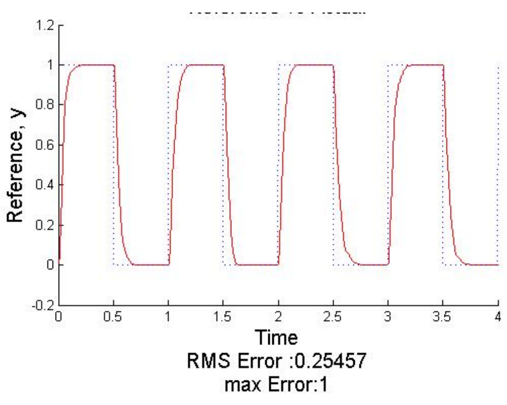


Figure 4-20 - %30 Packet Loss Basic NCS

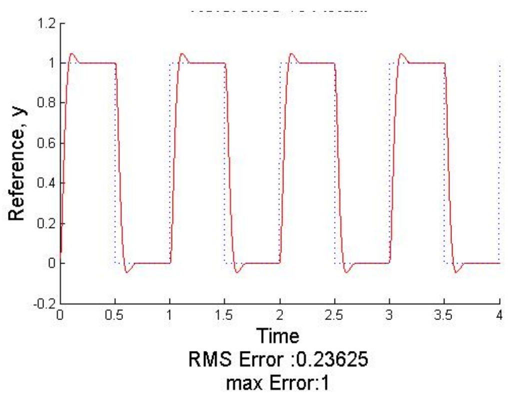


Figure 4-21 %40 Packet Loss MBPNCS

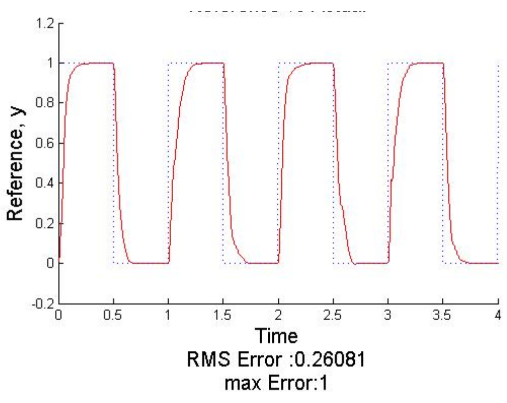


Figure 4-22 - %40 Packet Loss Basic NCS

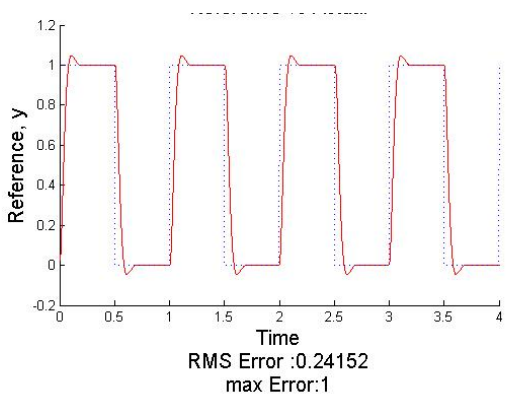


Figure 4-23 %50 Packet Loss MBPNCS

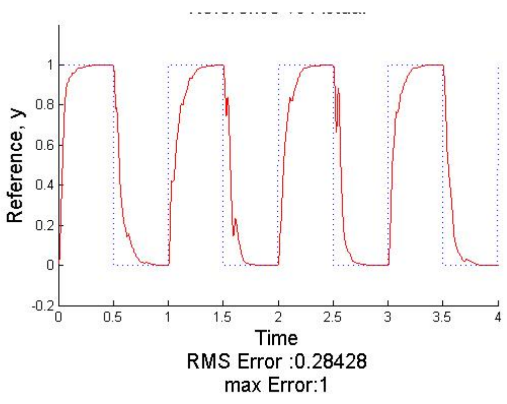


Figure 4-24 - %50 Packet Loss Basic NCS

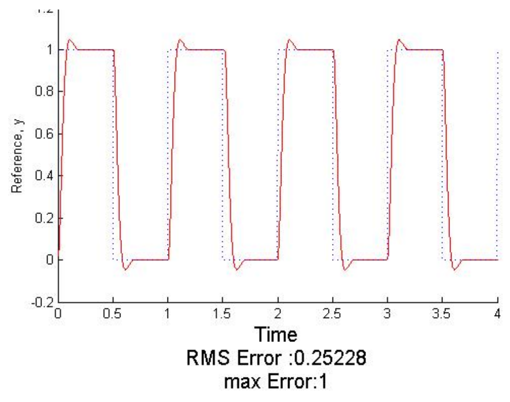


Figure 4-25 %60 Packet Loss MBPNCS

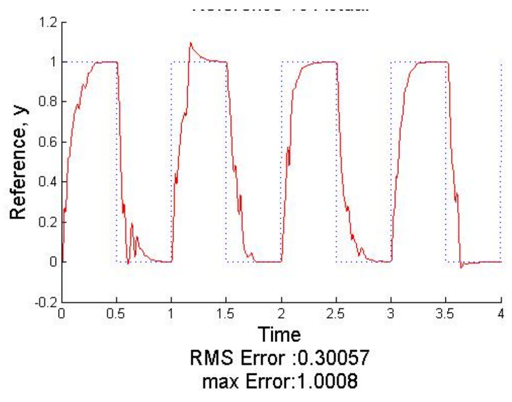


Figure 4-26 - %60 Packet Loss Basic NCS

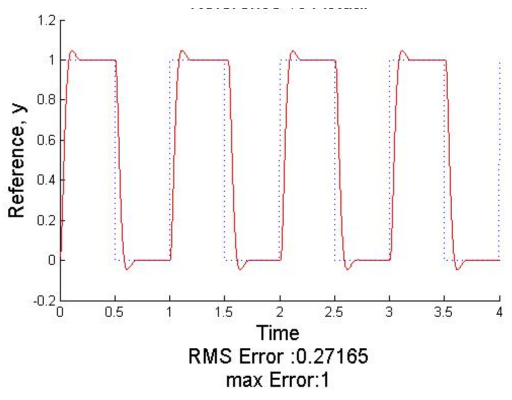


Figure 4-27 %70 Packet Loss MBPNCS

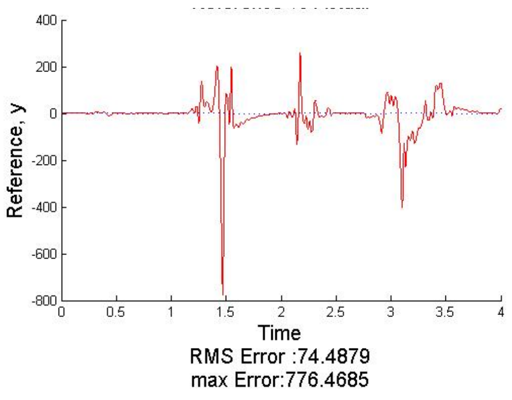


Figure 4-28 - %70 Packet Loss Basic NCS Note change in Y axis scale

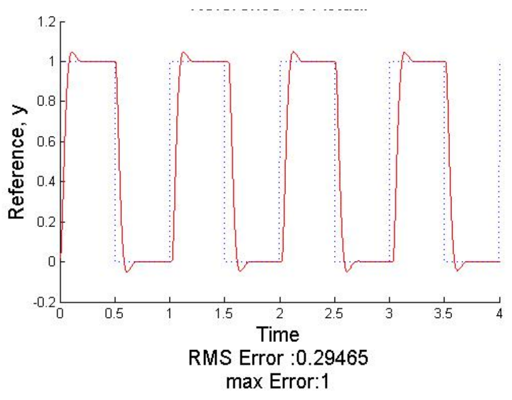


Figure 4-29 %80 Packet Loss MBPNCS

Unstable

Figure 4-30 - %80 Packet Loss Basic NCS

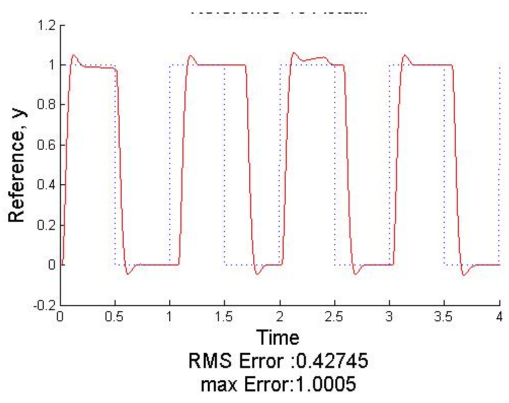


Figure 4-31 %90 Packet Loss MBPNCS

Unstable

Figure 4-32 - %90 Packet Loss Basic NCS

4.2.4 Effects of Noise

There is noise in all systems. It is not possible to create a system which has no noise. Noise can have a diverse number of sources.

If noise has a value changing effect on a packet then this is detected and the packet is considered lost. This translates as packet loss to our network. In our experiments noise is considered only in the analog part of the system. This is measurement noise at the sensors and noise applied to the control signal generated at the actuator node.

Band limited additive white noise is added to the control signal that the actuator node applies to the plant. The power of the noise to be added to the signal is related to the dynamic range of the signal. To have a relation between the noise power and the signal, noise power will be expressed as a fraction of the RMS of the control signal *u* generated under ideal network conditions.

$$u_n(t_k) = u(t_k) + n(t_k) \tag{17}$$

Where $u(t_k)$ is the control signal generated by the control algorithm, $n(t_k)$ is noise and $u_n(t_k)$ is the control signal with noise. $n(t_k) = f(u_{avg} * C_n)$ is a function that generates band limited white noise with noise power as parameter. C_n is a fixed coefficient showing the amount of noise and u_{avg} is the RMS value of u determined statistically. Note that this noise is pseudorandom; however in this paper the same initialization seed value has been given in order for various systems to be compared under the same noise conditions.

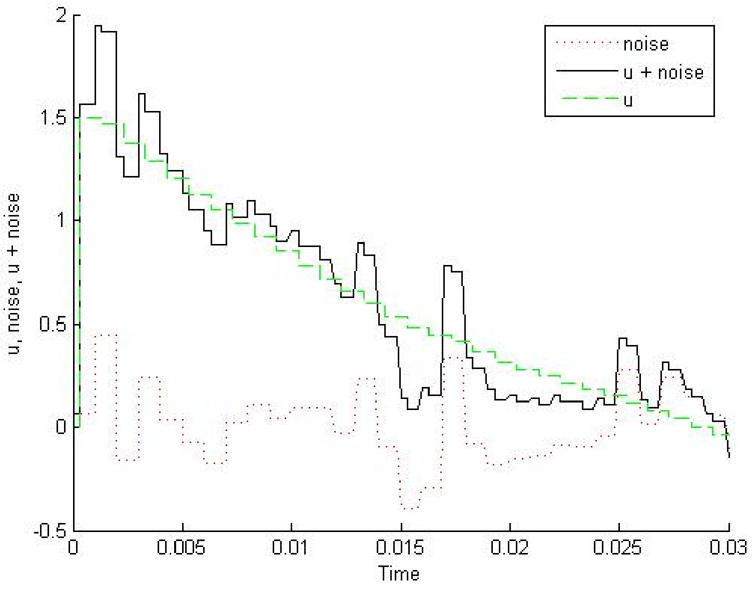


Figure 4-33 - Noise and its effect on u

The effect of noise on the control signal u can be seen in Figure 4-33. The simulation on Figure 4-34 took place on a lossless and fixed time delay network. Packet loss is gradually increased and the controller's robustness against noise can be measured in figures Figure 4-34 through Figure 4-37.

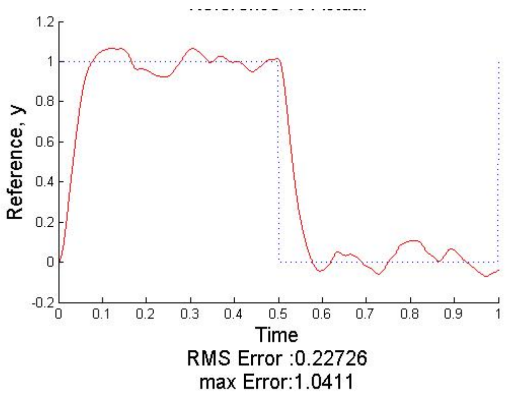


Figure 4-34 – MBPNCS subject to 0.001 Noise, %0 Packet Loss

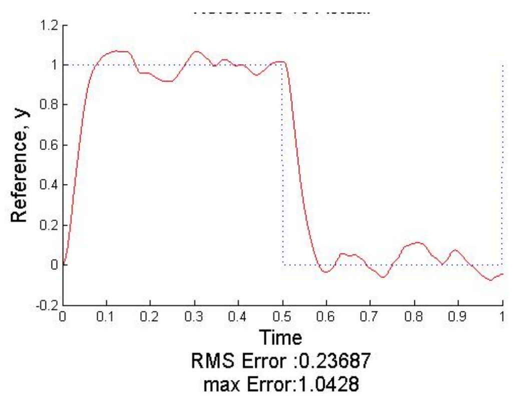


Figure 4-35 MBPNCS subject to 0.001 Noise, %40 Packet Loss

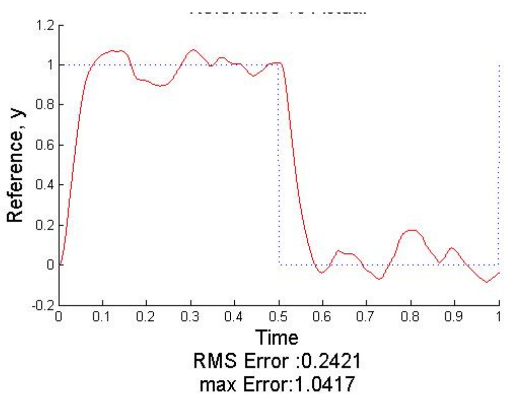


Figure 4-36 MBPNCS subject to 0.001 Noise, %60 Packet Loss

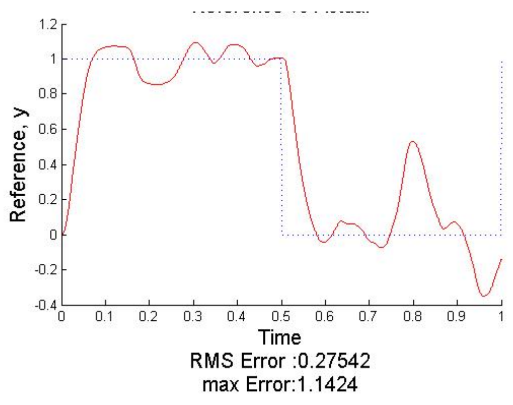


Figure 4-37 MBPNCS subject to 0.001 Noise, %70 Packet Loss

The effect of low noise is not important up to %60 packet loss. But after %70 packet loss the effect of noise becomes significant in control quality. Effects of noise is reduced through the feedback loop. High rate of packet losses mean that the feedback loop cannot be closed until controller node to actuator node synchronization is regained as explained in section 3.2. Therefore losing the advantage of the closed loop system makes the system vulnerable to noise and therefore control quality diminishes, as packet loss increases the systems noise reduction ability also decreases.

Figure 4-38 shows the performance of the MBPNCS subject to 0.001 Noise with %70 packet loss, also the signal number used from the last received packet is depicted. Here it may be noted that where the control quality is at its worst (at 0.8 seconds in to the simulation) controller node to actuator node synchronization is broken for ~ 0.6

seconds, therefore the system can be considered to be running on open loop for 0.6 seconds, this means that the actuator node has not received consecutive 60 acceptable packets from the controller.

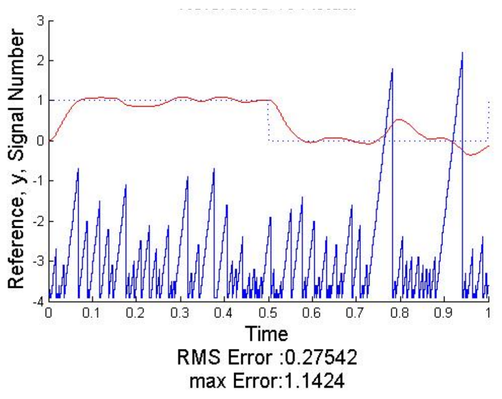


Figure 4-38 MBPNCS subject to 0.001 Noise, %70 Packet Loss with Signal Number

On loaded Ethernet networks it is not uncommon to see %40 packet loss. Therefore the performance of the system is examined at %40 packet loss and various noise levels. Results can be seen in Figure 4-39 to Figure 4-46.

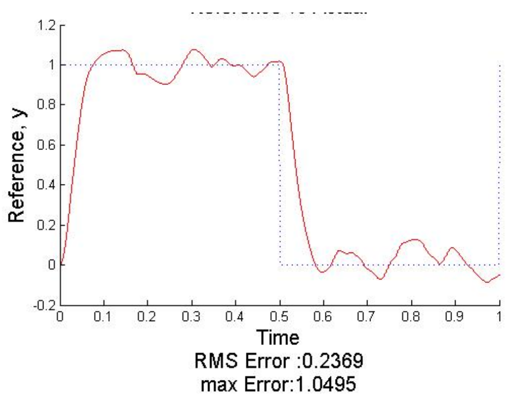


Figure 4-39 MBPNCS subject to 0.002 Noise, %40 Packet Loss

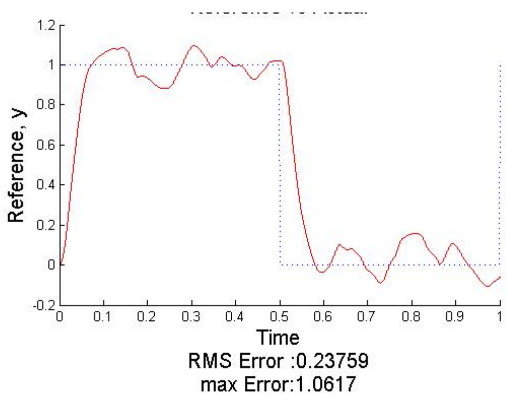


Figure 4-40 MBPNCS subject to 0.003 Noise, %40 Packet Loss

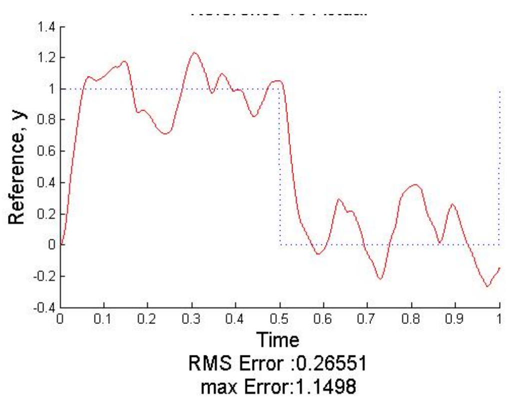


Figure 4-41 MBPNCS subject to 0.01 Noise, %40 Packet Loss

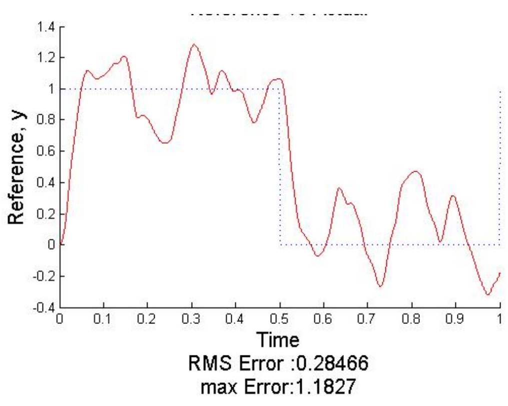


Figure 4-42 - MBPNCS subject to 0.015 Noise, %40 Packet Loss

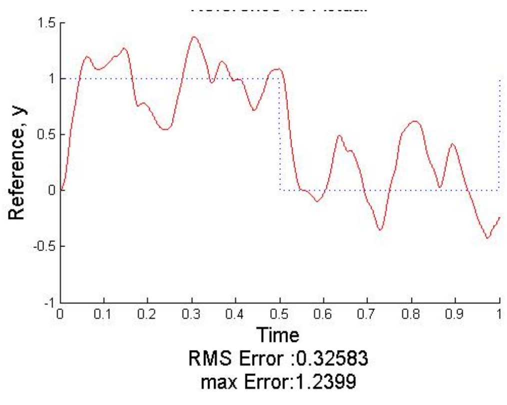


Figure 4-43 - MBPNCS subject to 0.02 Noise, %40 Packet Loss

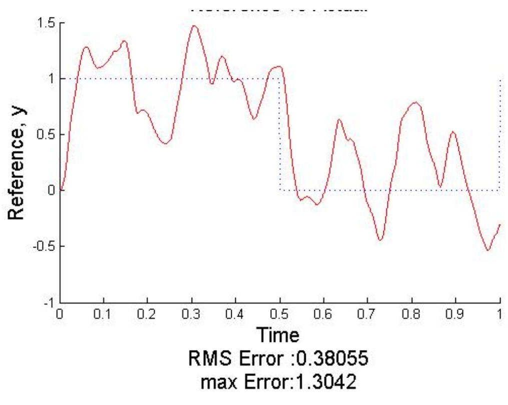


Figure 4-44 - MBPNCS subject to 0.025 Noise, %40 Packet Loss

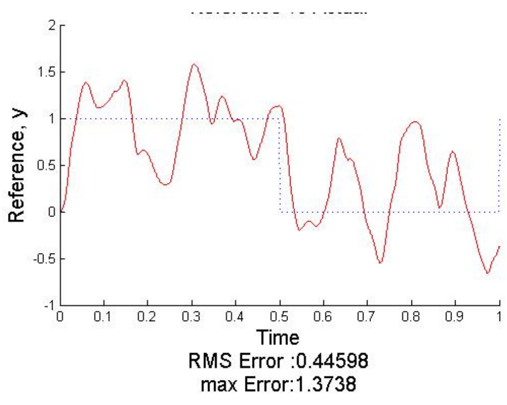


Figure 4-45 - MBPNCS subject to 0.03 Noise, %40 Packet Loss

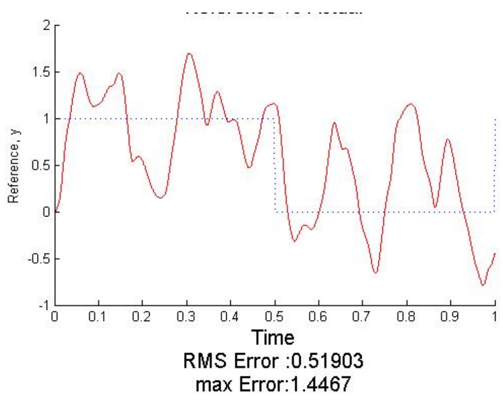


Figure 4-46 - MBPNCS subject to 0.035 Noise, %40 Packet Loss

As the noise increases it may be observed that the RMS Error also increases. When the noise is sufficiently large it dominates the control signal and therefore the system cannot be controlled. At low packet losses the effect of noise is unrelated to packet loss since the system subject to no packet loss also exhibits the similar degradation in performance as can be compared from Figure 4-47.

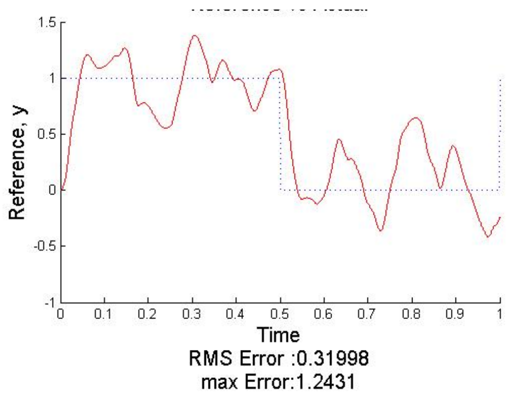


Figure 4-47 MBPNCS subject to 0.035 Noise, %0 Packet Loss

4.2.5 Close to Nyquist Frequency

In this section the effects of running the system at frequencies close to the minimum sampling rate is studied. Control systems are much more vulnerable to noise and packet loss under low frequencies. Since the system is running closer to Nyquist sampling frequency the output of the controller is more critical.

Here we examine the system close to its minimum sampling frequency according to the Nyquist theorem. On a lossless network and a sampling frequency of 100Hz the system performance is identical to the 1kHz version as shown by Figure 4-48.

.

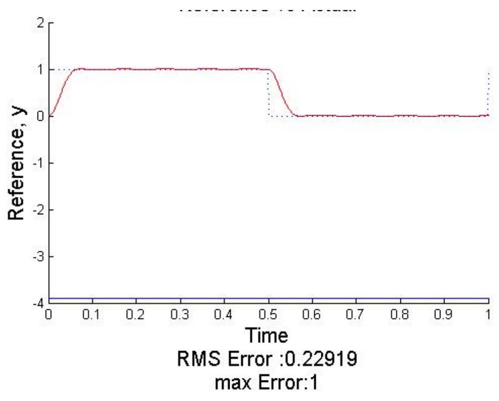


Figure 4-48 - MBPNCS 100Hz, %0 Packet Loss, Noiseless, signal number is also shown

Performance of MBPNCS under noiseless conditions with increasing rate of packet loss was investigated. The results can be seen in Figure 4-49 to Figure 4-52.

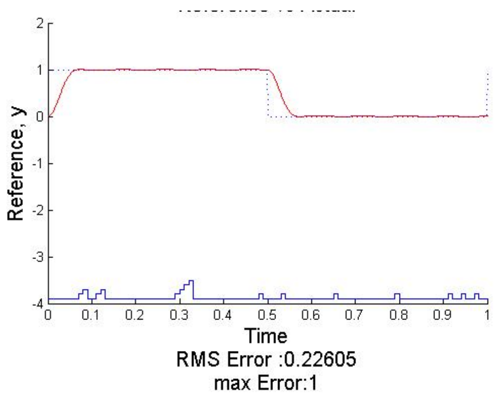


Figure 4-49 MBPNCS 100Hz, %10 Packet Loss, Noiseless, signal number is also shown

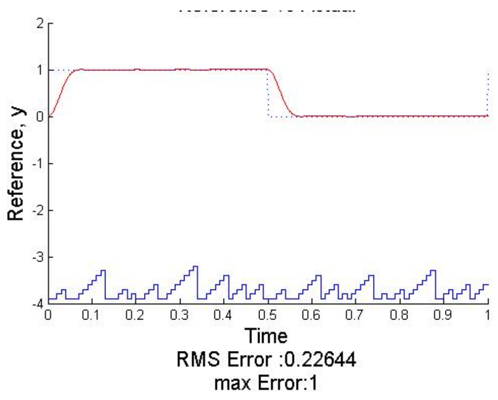


Figure 4-50 - MBPNCS 100Hz, %50 Packet Loss, Noiseless, signal number is also shown

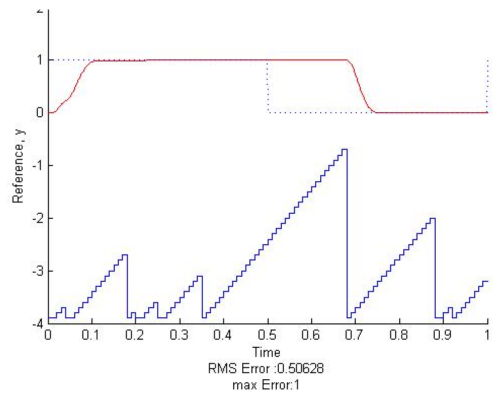


Figure 4-51 - MBPNCS 100Hz, %70 Packet Loss, Noiseless, signal number is also shown

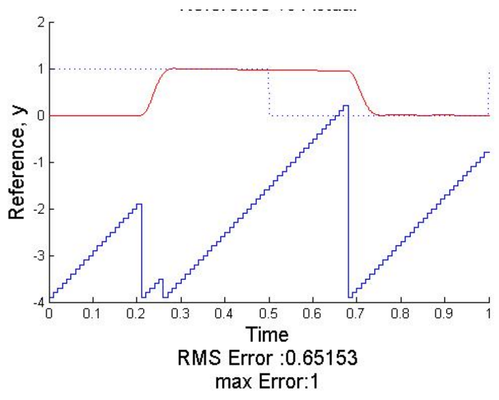


Figure 4-52 - MBPNCS 100Hz, %90 Packet Loss, Noiseless, signal number is also shown

Intervals where the actuator relies on a predicted control signal may be monitored by the signal number indicated on the figures.

The proposed model based predictive networked control system is resilient against packet loss over the network. However there is a delay in the delivery of the reference as explained in section 4.2.3. This is very clearly visible on Figure 4-52.

The basic NCS on the other hand is not capable of tolerating such high rates of packet loss. This is because packet loss has an effect on the sampling rate of the system. For example if every other packet is lost this has the same effect as halving the sampling frequency.

The effect of packet loss rate on performance of basic NCS can be seen below in Figure 4-53 through Figure 4-55.

The basic NCS performance degrades at %20 packet loss. This is due to the fact that as packet loss occurs in a random manner, it tends to change the sampling time of the system, and here the sampling time falls below the Nyquist sampling rate. Therefore the system can not be controlled accurately.

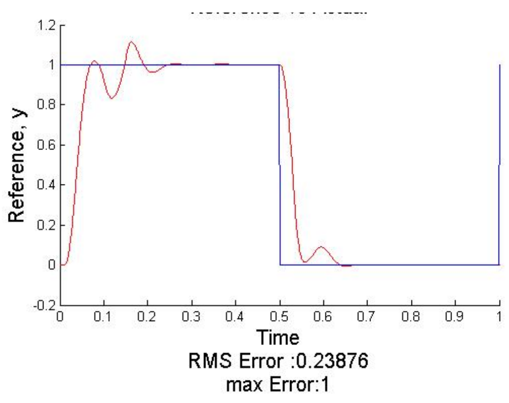


Figure 4-53 – Basic NCS 100Hz %10 Packet Loss

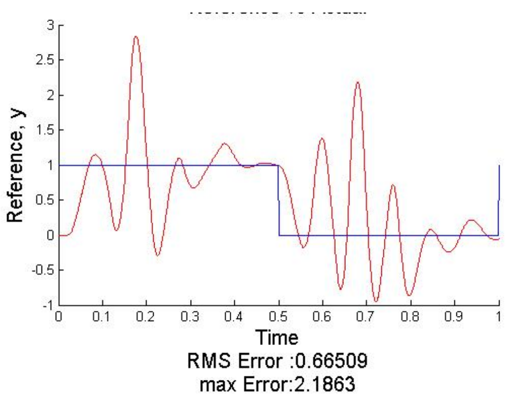


Figure 4-54 Basic NCS 100Hz %20 Packet Loss

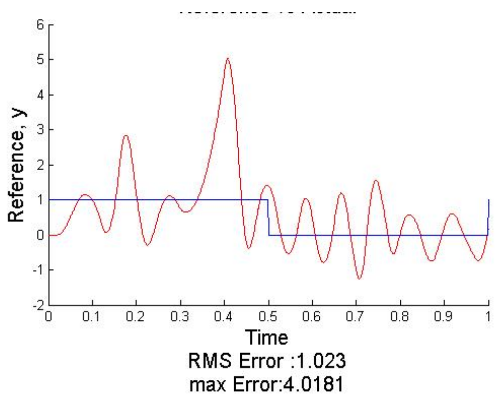


Figure 4-55 - Basic NCS100Hz %30 Packet Loss

Even at %10 packet loss the basic NCS starts to oscillate.

To show that 100 Hz is indeed the lowest sampling rate possible for this plant the experiment is retried with half the sampling frequency, 50Hz (Figure 4-56). Even under ideal conditions (where the basic NCS and the MBPNCS function in an identical manner) the system is uncontrollable as expected .

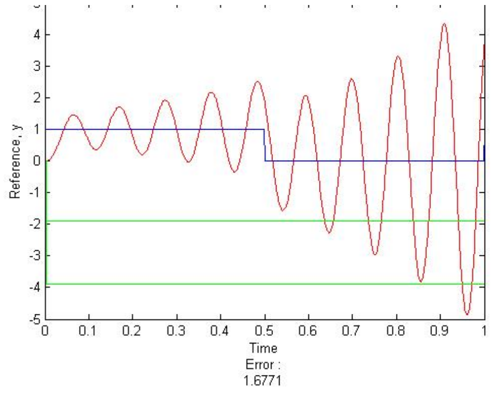


Figure 4-56 – Basic NCS, 50Hz, Lossless, Noiseless

Noise has a much larger effect on the system as the sampling rate of the system is lower, the controller node has fewer instants where it has an effect on the plant.

The MBPNCS maintains control at %60 packet loss as depicted by Figure 4-57. However at %70 packet loss the system's performance is poor Figure 4-58. It should be noted that the actuator node has not received control signals for nearly 30 periods between times 0.35-0.65.

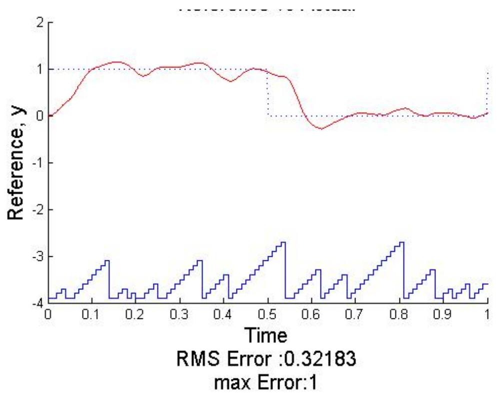


Figure 4-57 - MBPNCS 100Hz, %60 Packet Loss, 0.001 Noise

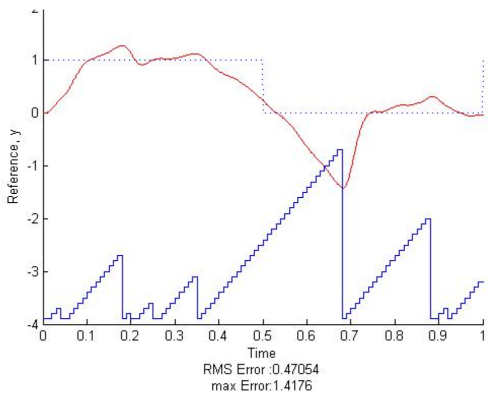


Figure 4-58 - MBPNCS 100Hz, %70 Packet Loss, 0.001 Noise

4.2.6 Prediction Horizon Calculation

To determine the minimum amount of predictions necessary to control the plant, without any degradation in performance, the control signal sent to the plant is examined. Figure 4-59 shows the control signal and Figure 4-60 shows the plant output. The amount of predictions necessary can be related to the plants settling time. The figures show that the control signal rests at 0 after the 0.20th second and the plant has reached the reference. The model will also provide this. After the 0.20th second of predictions the predicted control signal will be 0, therefore making predictions beyond this point would be unnecessary consumption of computational power. But this may not hold for open loop unstable plants. Tilbury has worked on Using Deadbands to Reduce communication in Networked Control Systems [12. This same approach can be used not to reduce communication but to reduce simulation.

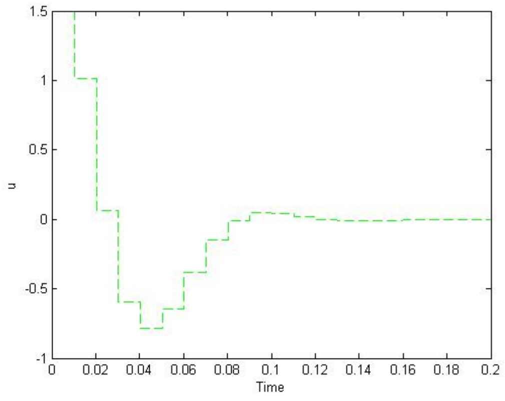


Figure 4-59 - MBPNCS control signal for step input reference on a lossless network at 100Hz

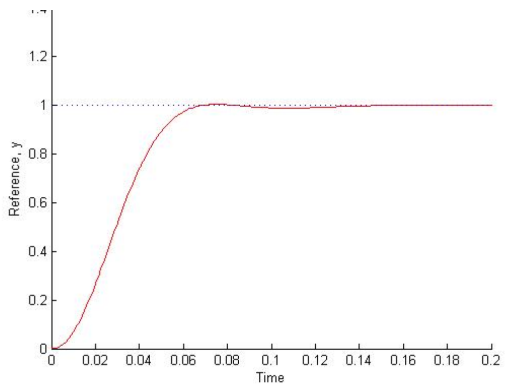


Figure 4-60 MBPNCS output for step input reference on a lossless network at 100Hz

5 CONCLUSION

In this work, a novel networked control system method is presented. The model based predictive network control system method (MBPNCS) takes advantage of the fast processing power of computers and establishes a system that is robust as a networked control system (NCS) using non real-time networks. The architecture is a distributed system, which relies on computational capacities of the sensor and actuator nodes to ensure synchronization to the controller node. Controller states are calculated to some extent at the sensor node to ensure regain of synchronization during recovery from packet loss.

This new NCS method relies on a plant model to predict the effect of control signals to be applied to the plant, and generates an array of control signals that span over a specified prediction horizon. These signals are sent over to the actuator node where a selection algorithm is applied in order to assure the synchronization between the controller and actuator nodes. Then the control signal is applied to the plant. Should communication between the controller node and actuator node fail the actuator node uses the predicted control signals in the lastly received control packet, making the system resilient against packet loss. During operation in this regime, no packets are transmitted over to the actuator node and therefore changes in reference cannot be delivered.

This new architecture is applied to a DC servo motor and various aspects of the MBPNCS have been examined. It has been observed that the architecture is resilient against packet loss. The destabilizing effect of packet loss is reduced to unresponsiveness to the reference command which is an inevitable consequence of communication loss. The effect of noise on the control signal is small in low packet losses, however since the feedback is interrupted, the system is effected by noise when packet loss increases.

Finally means of determining the size of the prediction horizon is provided. This is achieved by measuring the settling time for maximum variation in the reference of the system and setting the prediction horizon to span from the previous state to the new reference. This enables the system to reach the desired reference and stabilize even if the communication is lost after the transmission of a single control packet, however this is only valid for open loop stable systems.

6 FUTURE WORK

Controller node to actuator node packet loss has a degrading performance on the system. The actuator node selection algorithm ignores some packets. A compensation method can be devised to enable all packets to be accepted by the actuator node. This could be achieved by using the plant model, where the controller not only assumes the case that the actuator node receives the last packet but also the case where it does not receive the last packet. Then at the actuator node side of the network, the actuator can look at the last packet received and decide which control signal bunch it should apply to the plant. However this could be computationally intensive.

The system discards late packets. Means of benefiting from the information in these packets should be exploited. This would further increase tolerance to time delays and packet loss.

In this work all nodes are assumed to be totally synchronized time wise. However this may not always be the case. If the clocks are de-synchronized then some packets will be perceived as late and therefore discarded. Methodologies to prevent this effect could be examined. The gradual drift in the packet arrival times could be a hint to this type of de-synchronization.

7 REFERENCES

[1] Q Ling, M. Lemmon "Robust performance of soft real-time networked control systems with data dropouts" *IEEE conference on decision and control*, Las Vegas, Nevada, December 2002.

[2] DD Siljak M. B. Vukcevic "Decentralization, Stabilization, and Estimation of Large-Scale Linear Systems" *IEEE Transactions on Automatic Control*. Vol. AC-29 No.11 November 1984

[3] Arno Linnemann "Decentralized Control of Dynamically Interconnected Systems" *IEEE Transactions on Automatic Control*. Vol. AC-29 No.11 November 1984

[4] M. E. Sezer, D.D. Siljak "On Structural Decomposition and Stabilization of Large-Scale Control Systems" *IEEE Transactions on Automatic Control* Vol AC-26, No. 2, April 1981

[5] X. Liu and A. J. Goldsmith, "Wireless Network Design for Distributed Control" *IEEE Conference on Decision and Control*, 2004.

[6] B. S. Heck, L. M. Wills, and G. J. Vachtsevanos "Software Technology for Implementing Reusable, Distributed Control Systems" *IEEE Control Systems Magazine* February 2003

[7] J. Yook, D. Tilbury; N. Soparkar "Performance Evaluation of Distributed Control Systems With Reduced Communications" *IEEE Control Systems Magazine*, Vol. 21, no. 1, pp. 84-99, 2001.

[8] T. Tsuji, K. Ohnishi "A Controller Design Method of Decentralized Control System" IEEE *Power Electronics Conference* 2005

[9] M. Colnaric "Design of Embedded Control Systems" *ICIT 2003* Maribor, Slovenia 2003

[10] X. Liu and A. J. Goldsmith, "Wireless Medium Access Control in Networked Control Systems," *IEEE American Control Conference*, 2004.

[11] M.S. Branicky, S.M. Phillips, Wei Zhang, "Scheduling and feedback co-design for networked control systems," *Proc. 41st IEEE. Conf. on Decision and Control*, vol.2, no.pp. 1211- 1217 vol.2, 10-13 Dec. 2002

[12] P Otanez; J. Moyne, D. Tilbury "Using Deadbands to Reduce communication in Networked Control Systems" *Proceedings of the 2002 American Control Conference*. 2002

[13] J. K. Yook and D. M. Tilbury and H. S. Wong and N. R. Soparkar "Trading Computation For Bandwidth: State Estimators For Reduced Communication In Distributed Control Systems" *Proceedings of 2000JUSFA 2000 Japan-USA Symposium on Flexible Automation* July 23-26, 200, Ann Arbor, Michigan, USA 2000

[14] K. Natori, K. Onishi "An approach to design of feedback systems with time delay" To be published

[15] C. Mo-Yuen, Y. Tipsuwan "Gain adaptation of networked DC motor controllers based on QoS variations," *IEEE Transactions on Industrial Electronics*, vol.50, no.5pp. 936- 943, Oct. 2003

[16] JB Rawlings, "Tutorial Overview of Model Predictive Control", *IEEE Control Systems Magazine*, Vol. 20, No. 3, June 2000, pages 38-52.

[17] D. Henriksson, A. Cervin, K. Årzén "TrueTime: Real-time Control System Simulation with MATLAB/Simulink" *Proceedings of the Nordic MATLAB Conference,* Copenhagen, Denmark, October 2003.

[18] M. S. Branicky, V. Liberatore, and S. M. Phillips, "Networked Control System Co-Simulation for Co-Design", *American Control Conference,* Denver, USA, vol. 4, pp. 3341--3346, June 2003.

[19] J. Yook, D. Tilbury, K. Chervela and N. Soparkar, "Decentralized, Modular Real-Time Control for Machining Applications" *Proceedings of the American Control Conference*, pp 844-849, 1998.

[20] G. Walsh and H. Ye "Scheduling of Networked Control Systems" *IEEE Control Systems Magazine* February 2001

[21] G P. Liu, J. X. Mu and D. Rees D "Networked Predictive Control of Systems with Random Communication Delay", *UKACC International Conference on Control*, Bath, UK, Sept 2004. ID-015

[22] D. Henriksson, A.Cervin, K. Arzen, "Simulation of Control Loops Under Shared Computer Resources "*Proceedings of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, July 2002.

[23] F. Franklin, J. D. Powell and M. Workman, "Digital Control of Dynamic Systems" (3 rd Ed.), *Addison-Wesley*, 1997.

[24] Y. Fujimoto, T. Yakoh and K. Ohnishi "Dynamic Model of Decentralized Systems With Information Connection" *IEEE Transactions on Industrial Electronic*, Vol. 49, No. 3 June 2002

[25] B. C. Kuo, Digital Control Systems (2 nd Ed.), *Oxford University Press*, 1992

[26] M. Morisawa, K. Ohnishi "Connection Design for Motion Control System" *AMC2002* Maribor, Slovenia 2002

[27] K. Arzen and A. Cervin "Control and Embedded Computing: Survey of Research Directions" *Proceedings IFAC* 2005

[28] K. J. Astrom and B. Wittenmark. "Computer controlled systems: Theory and design" *Prentice-Hall*, ISBN 1990