STABILITY AND IMPLEMENTATION OF MODEL BASED PREDICTIVE NETWORKED CONTROL SYSTEM

By

OZAN MUTLUER

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 2009

STABILITY AND IMPLEMENTATION OF MODEL BASED PREDICTIVE NETWORKED CONTROL SYSTEM

APPROVED BY:

AHMET ONAT (Dissertation Advisor)	....
GÜLLÜ KIZILTAŞ	....
ÖZGÜR GÜRBÜZ	....
AYHAN BOZKURT	....
ALİ KOŞAR	....

DATE OF APPROVAL: ....

ABSTRACT

Digital control systems that have computer nodes which communicate over a data loss and random delay prone common network are called Networked Control System (NCS). In a typical NCS, the sensor, controller and the actuator nodes reside in different computers and communicate with each other over a network. Random delays and data loss of the communication network can endanger the stability of the NCS and retransmission of data is not feasible in control applications since it adds delay to the system.

The aim of this thesis is to verify that the distributed NCS method called Model Based Predictive Networked Control System (MBPNCS) can be implemented using an observer and that it can control an open loop unstable plant. MBPNCS compensates for missed and late data by implementing an intelligent predictive control scheme based on a model of the plant. MBPNCS does not use retransmission and does not guarantee timely delivery of data packets to each computer node since this solution is not feasible on every control application and every communication medium. Instead, MBPNCS offers a control solution that can work under random network delay and data loss by the use of a predictive architecture that predicts plant state estimates and respective control signals from actual plant states.

In this thesis, MBPNCS is described along with an introduction to a theoretical stability criterion. This is followed by an implementation of MBPNCS with two different plants. First, MBPNCS is implemented with an observer based DC motor plant to demonstrate the system's efficiency with an observer. Next, MBPNCS is implemented with an inverted pendulum to demonstrate the system's efficiency with an open loop unstable plant. Finally, two separate MBPNCS's are implemented over a common network to demonstrate the systems efficiency and feasibility in industrial applications. The results show that considerable improvement over performance is achieved with respect to an event based networked control system.

ÖZET

Veri kaybı ve rastgele gecikmelerin bulunduğu bir haberleşme ağı üzerinden, dağıtık bir sistem ile kontrol uygulayan dijital kontrol sistemlerine ağ bağlantılı kontrol sistemleri denir. Tipik bir ağ bağlantılı kontrol sistemi farklı bilgisayarlara yerleştirilmiş ve ağ üzerinden haberleşen algılayıcı, kontrol ve eyleyici düğümünden oluşur. Tipik bir ağ bağlantılı kontrol sistemi, veri kaybı ve rastgele gecikmeleri verileri tekrar gönderme yaparak düzeltmeye çalışır. Ama tekrar gönderim, sistemdeki gecikmeyi artırır ve bu sistemin kararlığını tehlikeye attığı için kontrol uygulamaları için elverişli değildir.

Bu araştırmada Modele Dayalı Öngörülü Ağ Baglantılı Kontrol Sistemi (MODOAKOS) adı altında bir ağ bağlantılı kontrol sisteminin bir gözleyici ile çalışabileceği ve açık döngüde kararsız bir sistemde kararlı olduğu gösterilmiştir. MODOAKOS sistemdeki veri kaybı ve rastgele gecikmeleri, tesisin modelini kullanarak hesapladığı tahmini tesis durumları sayesinde telafi eder. MODOAKOS veri tekrar gönderimi yapmaz ve verinin zamanında düğümlere ulaşmasını beklemez çünkü bu çözüm endüstride kullanılan her haberleşme ağı için elverişli olmaz. Bunun yerine MODOAKOS veri kaybı ve rastgele gecikmenin olabileceği her haberleşme ağı üzerinden çalışabilen bir çözüm sunar ve bunu akıllı öngörü algoritması sayesinde başarır.

Bu tezde öncelikle MODOAKOS tanımlanmıştır ve teorik bir kararlılık kriteri sunulmuştur. Bu sunuştan sonra MODOAKOS farklı tesisler kullanılarak uygulanmıştır. İlk olarak bir gözleyici kullanılarak DC motor üzerinden hız kontrolü yapılmıştır ve gözleyici kullanıldığı zaman sistemin verimliliği test edilmiştir. Ardından, MODOAKOS bir ters sarkaç tesisi üzerinde uygulanmıştır ve sistemin verimliliği açık döngüde kararsız bir tesis ile test edilmiştir. Son olarak iki ayrı MODOAKOS sistemi aynı ağ bağlantısı üzerinde uygulanmıştır ve sistemin endüstriyel uygulamalardaki elverişliliği test edilmiştir. Yapılan deneyler sonucu sistemimizin diğer basit öngörüsüz ağ bağlantılı kontrol sistemlerine oranla daha yüksek bir performans ve kararlılıkla çalıştığı görülmüştür.

To my family and friends

ACKNOWLEDGEMENTS

I would like to state my gratitude to my MS. supervisor, Asst. Prof. Dr. Ahmet ONAT. I would like to thank him for his enthusiasm, encouragement, guidance and help that helped me complete this thesis. He has provided me with good advice, good teaching, good company, and lots of good knowledge in the process of my thesis writing period. It has been a great honor and privilege to work with him.

I also would like to thank my fellow colleagues who have shown me a great support during my times at Sabancı University. I am especially grateful to Teoman Naskali and Emrah Parlakay for their thesis related help and support. I would also like to thank İlker Sevgen, Utku Seven, Evrim Taşkıran, Kaan Öner, Ender Kazan, Berk Çallı and many other friends at Sabancı University Mechatronics Laboratory.

Finally, I wish to thank my parents, Nadir Mutluer and Şükran Mutluer, my brother Selim Can Mutluer. They have been the fuel and moral support that has helped me finish this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

NCS	Networked Control System
MBPNCS	Model Based Predictive Networked Control System
bNCS	Basic Networked Control System
AD	Analog to Digital converter
DA	Digital to Analog converter
RMS	Root mean square
MPC	Model Predictive Control
ACK	Acknowledgement
EMF	Electromotive Force

Chapter 1

INTRODUCTION

A Networked Control System (NCS) is a control system that uses a real time communication structure which exchanges control and feedback data through a network. The control and feedback data are shared via communication packets. A basic NCS has four generic components; three computer nodes and one communication network. The three computer nodes are; sensor node which is responsible for gathering sensor data, controller node which calculates the control signal and the actuator node which implements the appropriate control signal output to the plant. The communication network is responsible for the communication between the computer nodes. The most prominent feature of the NCS is that it connects computer peripherals to a physical plant thus, enabling execution of control from long distance. A basic NCS structure is shown in Figure 1.1.

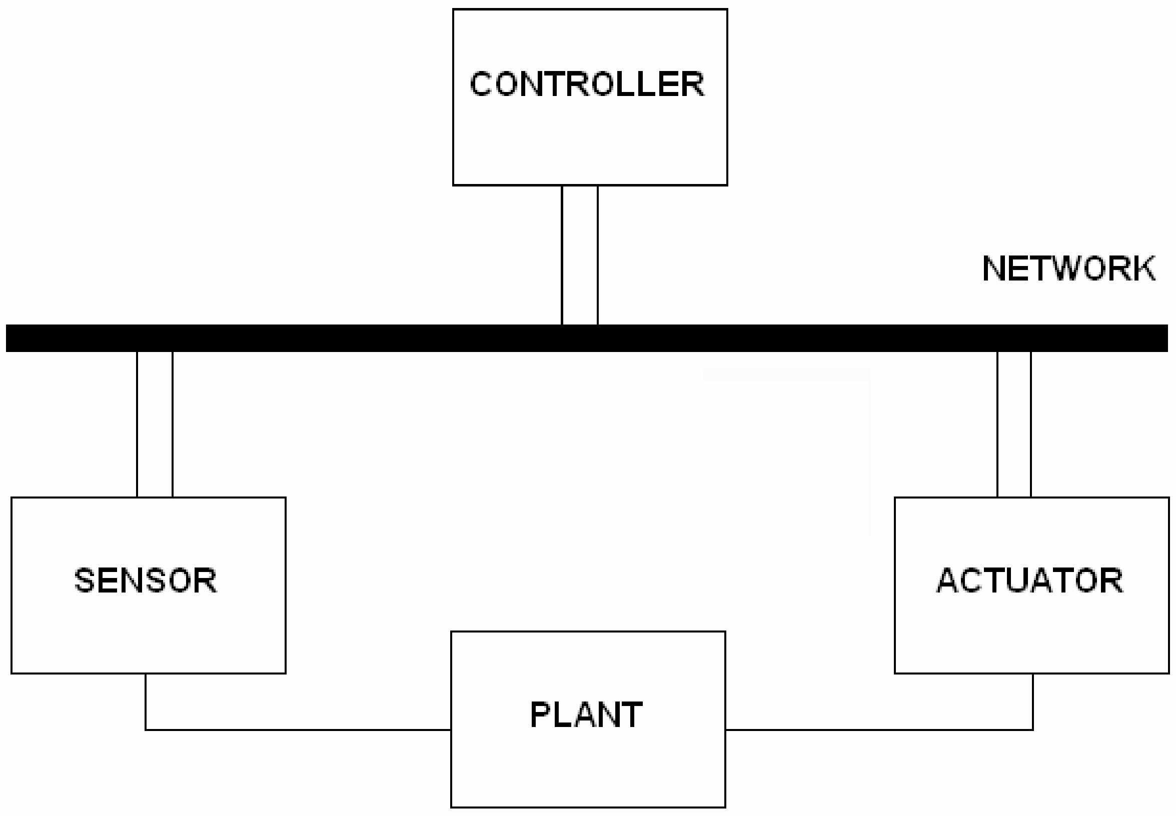

Figure 1.1 Basic Structure of Networked Control Systems

In the networked control system, the sensor node periodically samples the sensor data output of the plant, encodes the data into a packet and sends it to the controller node. The controller node takes in the sensor data, applies the control algorithm and sends out the control signal to be applied to plant to the actuator node within a data packet. The actuator node takes in the controller data and applies the control signal to the plant. Since the sensor, controller and actuator data travel through a network connection, there are communication delays and data loss, thus not all packets make it to their destination on time, and some are lost on the way due to problems associated with the network connection such as interference, collision and retransmission. Communication delay between the sensor node and the controller node that has occurred following sampling instant t_k is $\tau_{SC}(t_k)$, computation delay in the controller node that has occurred following sampling instant t_k is $\tau_C(t_k)$, and communication delay between the controller node and the actuator node that has occurred following sampling instant t_k is $\tau_{CA}(t_k)$ [1]. The total delay in the system is given by (1.1) :

$$\tau\ (t_k)=\tau_{SC}(t_k)+\tau_C(t_k)+\tau_{CA}(t_k) \tag{1.1}$$

Most communication systems use a confirmation of reception which, if not received within a specific amount of time, triggers a retransmission. However, in a networked control system, this means extra time is lost within a sampling interval, and rather than resending the old data, it is better to transmit a more recent plant output sample or a newer control signal instead. Similarly, delayed packets may be considered as lost since control signal applied late to the plant does not guarantee stability.

Networked control systems (NCS) improve the performance of conventional digital control systems and have increased system agility, reliability and ease of system diagnosis and maintenance. Although the system has such advantages, a simple NCS as explained above is still vulnerable to random communication delay and loss on the network which jeopardizes the stability unless special measures are taken since the communication delays decrease the phase margin of the control system and data loss can be considered as noise.

Model Based Predictive Networked Control System (MBPCNS) is a networked control system method where stable control is possible even under random delay and data loss. The loss and delay of packets in communication are compensated in MBPNCS by the help of an intelligent predictive scheme put in the controller and the actuator node. MBPNCS holds a model of the plant inside the controller and computes the next *n* predicted controller output states each period based on the plant model. The predicted controller outputs are appended to the controller output signal in a given period and sent to the actuator node. The actuator node employs a state machine to determine whether the generated control predictions are based on a valid plant state. MBPNCS rejects delayed packets and dropped packets are not retransmitted.

In order to achieve a trustful distributed control system that would be used in an imperfect environment, one has to prove the stability of the proposed method through theory and demonstrate practical implementation. This thesis aims in answering the performance and stability related questions of the MBPNCS.

In this thesis, we aimed at showing that the performance of the MBPNCS, a system that has been simulated [1] and implemented with a DC motor [2], is better than that of conventional NCS by implementing the system with an open loop unstable plant and with an observer. A theoretical stability discussion is also shown.

Chapter two of this thesis addresses previous studies that have been carried out in the area of NCS and solutions that have been proposed to the problems associated with it. In chapter three, MBPNCS method used in this thesis is explained in detail. Chapter four explains the setup and the implementation procedure of MBPNCS that have been carried out. Chapter five includes the theoretical stability discussion. Chapter six presents the experimental results and chapter seven concludes the study and presents ideas for future work.

Chapter 2

2. LITERATURE REVIEW

Studies on distributed control of large scale systems were active as early as 1970s when nationwide phase synchronization of power plants in large countries was an important issue since it was not known how to transmit power over thousands of kilometers lacking a common time base. Important work has been done in the last decade on how to synchronize country wide electricity grids that use multiple power plants [3][4][5].

NCS has been an outcome of development in the network theory and control theory and lack of interaction thereof. The network theory aims in increasing the average throughput of the network and has little concern on the latency of transmitted data packets which results in the aim of increasing efficiency of the network by sending a batch of data packets at once. In the case of a packet loss, retransmission of the packet is usually expected and the loss of a packet is sensed by the use of confirmation of receipt flag called acknowledgement (ACK). Retransmission however steals from the transmission time of the packet and causes latency which is one of the reasons why it is not possible to put an upper bound on the packet transmission time and the stochastic nature of the network being another. In control theory however, the control loop is assumed to work in a centralized manner and have no information loss or delay due to the transmission of information meaning the implementations are free of jitter. The stability of the system is proved with this assumption in the control theory.

However, in a NCS stability of the controlled system depends on the timely and correct delivery of transmitted packets. Delivery of signals from sensor node to controller node and from controller node to actuator node must be guaranteed for each sampling time. For this reason, general application based computer networks used today are not suitable without presenting a robust solution to the NCS [6]. On the other hand, MBPNCS does not use retransmission or other network compensation methods. MBPNCS does not assume any direct link between the nodes. MBPNCS is designed to work in a network with packet loss and delay. In this chapter, several methods that attack the problem areas in NCS with various assumptions and shortcomings are summarized.

2.1. Co-design of NCS

An unnecessary rate of data transfer would increase lost packets and packet latency and these two problems would risk stability, whereas too low rate will not be enough for the requirements of the control algorithm. The network and control components of a NCS should be designed together in order to find an optimum amount of data transfer that would not corrupt the stability of the controlled system. In order to keep stability and not lose network performance, the network and the control system should be designed together with a suitable compromise on each side. Branicky, Phillips and Wei have used rate monotonic scheduling algorithm and have studied the effects of packet loss and the associated cost functions [7]. NCSs are overloaded which results in some loss of packets. Overloading results in un-schedulable systems to be scheduled and the effects of dropped packets are concluded to be insignificant according to this research.

2.2. Reduction of Communication

Packet loss and latency reduce the quality of service (QoS) of the controlled system and affects the stability. Much of the work on NCS has focused on decreasing the amount of network communication which aims in finding a better linear time invariant approximation of the network [8]. The research has used several methods in reducing the amount of communication in the network. Some of the methods used are described below.

2.2.1 Deadbands

Networks have unnecessary communication with packets containing identical data transmitted between the nodes. This research focuses on reducing the amount of communication by reducing the amount of unnecessary communication. Only the first packet is sent in the case where consecutive packets contain identical or similar data. In the case of no packet transmission for a given time, receiver node uses the most recent packet received. Otane, Moyne and Tilbury studied the effect of deadband control in a network with no packet loss and reliable communication [9]. This research is assumed to work with a perfect network that has neither packet loss nor delay.

2.2.2 Estimators

Estimators use the model of the plant on the receiver side, and aims in reducing the amount of communication in the network. The models of components produce the data to be used by the nodes, thus data that arrive from the network is not needed which reduces the amount of communication in the network. The system holds a threshold value that is the upper limit to the error between the model estimated values and the true values. If this threshold is exceeded, the actual value is broadcast to the system and the estimators are updated to the actual value. Yook, Tilbury, Wong and Soparkar concluded that this system saves great amount of bandwidth due to communication reduction, however great risk to the stability occurs in the case of a communication breakdown when the threshold is exceeded and the parameters are updated [10].

2.3. Network Observers

The delay in the network can be considered as a disturbance and a disturbance observer is utilized as a solution. Disturbance observer architecture is used to calculate this disturbance which is then added to the control signal. This would have similar effects on the closed loop control system as the Smith predictor [11] and eliminates the effect of the delay introduced by the network. This research assumes that delay in the network is slow varying or correlated.

2.4. Gain Adaptation

The Quality of Service (QoS) of the system may change due to changes in the traffic load of the network. An intelligent gain adaptation scheme is used to measure the QoS in the network and calculates the controller gains. The performance of the network and delay directly affects the system, thus recalculated controller gains are used in the new delay affected network [12].

2.5. Model Predictive Control

Model Predictive Control (MPC) is a predictive control scheme that has been used for a long time in control systems. MPC assumes that the sensor and the controller are connected directly without any communication network and a-priori knowledge of the reference is assumed. The model of the plant resides in the controller node and the control outputs will be calculated using the model several sampling times in the future. First the system calculates a cost function that will be used a choice factor for the optimization of the future control variables. The control output to be applied to the plant is chosen by looking at the cost function and determining which output minimizes the cost function the most. This predictor resides in the controller and calculates the control signals up to the control horizon. This scheme is called the model predictive control.

Another predictor, called the network control predictor resides in the actuator and selects which signal to be applied to the plant. When the network predictor and model predictor are put together the networked model predictive control is found. The networked control predictor compensates the network communication delay and the predictive controller controls the system [13].

The short coming of this system is the direct connection between the sensor and the controller which is not feasible for every application. Breakdowns are frequent between the sensor and the controller and electrical noise may be a problem which should be taken into account for. Also prior knowledge of the reference is not applicable to every control system.

2.6. Predictive Approaches

Some studies have been conducted in recent years on the stability of predictive controllers similar to MBPNCS. Montestruque and Antsaklis [14] [15] focus on finding a state response for the system and finding a limiting factor for the norm of the response to prove Lyapunov based stability. Their research focus on a NCS model that has a prior knowledge of the update interval, thus the system update interval is not random. This research also assumes a lossless network. Liu, Xia et al. [16] also studied on a scenario of delay and packet losses and predictive controller similar to MBPNCS. However, their research does not utilize a mechanism of accounting for drift of state estimates caused by delay and loss in the controller to actuator link.

Chapter 3

3. MODEL BASED PREDICTIVE NETWORKED CONTROL SYSTEMS

3.1. MBPNCS

The main problems associated with networked control systems arise from the existence of packet delay and loss associated with the common network protocols and topologies connecting the nodes. The purpose of MBPNCS is to bring a solution that is stable and tolerant to problems that exist in the networked control systems. Minimizing delay and eliminating packet loss is one way to solve the problems that jeopardize stability of the NCS, however one can not guarantee that this solution is generic and would work on every network system available. Thus, MBPNCS does not deal with reducing packet delay and eliminating packet loss. In other words, it does not guarantee the timely and correct delivery of packets between the nodes. MBPNCS is a system that augments stability in networks with packet loss and delay.

A basic Networked Control System (bNCS) is a simple and commonly used networked control system. It will be used as the benchmark for the tests in this research. A bNCS works in the following way: The sensor node samples the output of the plant periodically, and sends the output to the controller. The controller node works in an event based manner, meaning that it is notified when there is a data packet arriving from the sensor node. The controller node applies the control algorithm to the incoming sensor data and sends out the control signal to the actuator via a data packet. The actuator node is also event based and notified on the event of a new message arriving from the controller. When there is a data packet arriving from the controller node, the actuator applies the control signal to the plant. It is important to note that the actuator node and the controller run their tasks only when there is a data packet arriving from the previous node.

At every sampling instant the controller node computes the output to be applied to the actuator at that time *t* and for the next *n* time instants using a model of the plant. The actual control signal and *n* predicted control signals are placed into a packet and sent to the actuator. In the case of a communication break between the sensor and the controller, the controller uses the estimated plant state values to predict the sensor data and implements the control algorithm with these values. The communication between the sensor, controller and the actuator node is done with data packets.

At every sampling time the actuator implements the control output to the plant using the actual output that is located at the beginning of the newly received controller-actuator packet. The *n* predicted control signals that follow the initial control signal are stored in a buffer in case of a communication breakdown between the controller and the actuator. In case of a communication breakdown the actuator starts applying the predicted control signals to the plant. This procedure is repeated until the communication between the controller and the actuator is restored. The limit for the number of predictions that can be applied to the plant is limited with the number of predictions that is *n*.

Model based predictive networked control systems are composed of five parts: A sensor node, a controller node and an actuator node, a communication network which is assumed to cause data loss and protocol delay, and a model of the plant presiding inside the controller node.

3.2. Sensor Node

The sensor node in MBPNCS periodically gathers data from the plant in every sampling time t_k , $x(t_k)$, the sensor node of the MBPNCS works similar to the sensor node of a NCS. The acquired sensor data is put into a packet and sent to the controller. No other communication is done by the sensor node; it uses a one way communication; gathering data and sending it out. In the case of a communication breakdown with the controller node, the sensor node is not responsible for compensation. The required compensation is done by the controller node.

3.3. Controller Node

The controller node in MBPNCS is an intelligent component of the system and is also time based. At the beginning of every period it receives the plant states from the sensor node. A control signal that will be consecutively applied to the plant is created using the sensor data and the control algorithm. The control algorithm is used to obtain the actual control signal based on $x(t_k)$ and the resulting control signal $u(t_k)$ is sent to the actuator via a packet using the communication network. The plant of MBPNCS is governed by (3.1) and (3.2):

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) \tag{3.1}$$

$$y(t_k) = Cx(t_k) \tag{3.2}$$

Since MBPNCS discards late packets and does not allow retransmission, also a model of the plant resides in the controller node. The estimated plant states of MBPNCS are governed by (3.3) and (3.4):

$$\hat{x}(t_{k+1}) = \hat{A}\hat{x}(t_k) + \hat{B}u(t_k) \tag{3.3}$$

$$\hat{y}(t_k) = \hat{C}\hat{x}(t_k) \tag{3.4}$$

where $\hat{A}$, $\hat{B}$, $\hat{C}$ are the plant model state transition and input matrices respectively.

If complete plant state cannot be measured, an observer can be used when the plant is observable. The resulting control algorithm would be [13]:

$$\hat{x}(t_k | t_{k-1}) = \hat{A}\hat{x}(t_{k-1} | t_{k-2}) + \hat{B}u(t_{k-1}, 0) + K_0(y(t_k) - \hat{y}(t_k | t_{k-1})) \qquad (3.5)$$

where $\hat{x}(t_k | t_{k-1})$ is the state estimate for t_k based on the informationfrom t_{k-1}, K_0 is the observer gain, $y(t_k)$, $\hat{y}(t_k)$ are actual and estimated plant outputs respectively. For example in the absence of a current sensor, an observer can be used to calculate the control output of a speed or position of a DC motor.

3.4. Control Algorithm

The control algorithm that resides in the controller node is a state feedback control, calculates the real control output using the control gain K_c. Thus the control output looks like:

$$u(t_k) = K_c x(t_k) \qquad (3.6)$$

This actual control signal is placed at the top of the control packet that is sent to the actuator to be applied to the plant consecutively.

The model in the controller is used to calculate n future estimates of the state of the plant where n is the estimate number used in our research but can be changed by changing the size of the transmitted packet. So a series of predicted control signals $\hat{u}(t_k, i)$ are calculated in an iterative fashion[17]:

$$\hat{x}(t_{k+i}) = \hat{A}\hat{x}(t_{k+i-1}) + \hat{B}\hat{u}(t_k, i-1) \qquad (3.7)$$

$$\hat{u}(t_k, i) = K_c \hat{x}(t_{k+i}) \tag{3.8}$$

where $i = 1, 2, ... n$.

At time t_k, control signal $u(t_k)$ applied to the plant is applied to the model. The output of the model $\hat{x}(t_{k+1})$ is the state estimate of the plant at time t_{k+1}. To compute the controller output at time t_{k+1}, control algorithm is applied to the estimated states $\hat{x}(t_{k+1})$. The control output $\hat{u}(t_{k+1})$ is then applied to the model of the plant to compute the next predicted output of the plant. This process is recursively applied *n* times and computed control outputs from $\hat{u}(t_{k+1})$ to $\hat{u}(t_{k+1})$ are placed in a data packet together with $u(t_k)$ to be sent to the actuator node. The error between the model estimates and the real plant output can be defined as:

$$\tilde{x}(t_k) = x(t_k) - \hat{x}(t_k) \tag{3.8}$$

$$\tilde{x}(t_{k+n}) = [(A + BK_c)^n - (\hat{A} - \hat{B}K_c)^n]x(t_k) \tag{3.9}$$

The plant model state transition matrices $\hat{A}, \hat{B}$ and control value K_c must guarantee that $\tilde{x}(t_{k+n})$ has an upper bound [2].

This scheme is the key to MBPNCS as it is useful in two cases; transmission problems between the actuator and the controller and the transmission problems between the controller and the sensor. The delay and loss packets between the sensor and the controller are compensated by this intelligent algorithm.

The controller node holds a variable called sensor flag (SF), which will actually be used by the actuator node. At time t_k, if no packet loss occurs between the sensor node and the controller node, the controller node sets the sensor flag variable to '1' and works as defined above by computing the control output signal $u(t_k)$ and predicted control output signals from $\hat{u}(t_{k+1})$ to $\hat{u}(t_{k+n})$ using the plant states $x(t_k)$ received from the sensor node. Else, sensor flag is set to '0'. Sensor flag is sent to the actuator within every packet to signal whether a control packet is based on a measured state or an estimated state of the plant.

At time t_k, in the case of a packet loss between the controller node and the sensor node, the controller node computes the $u(t_k)$ and $\hat{u}(t_{k+1})$ to $\hat{u}(t_{k+n})$ using the predicted plant state $\hat{x}(t_k)$ computed at time t_{k-1} and sets the sensor flag to 0. Since predicted plant states are used to compute *u(k),* control signal output is less reliable in comparison to the control signal output computed with the real plant states. If the packet loss events consecutively follow each other, reliability of the computed control signal output and predicted control signal outputs decrease each period with a rate of $\widetilde{x}(k)$. To overcome this reliability problem, sensor flag parameter is sent by the controller node and a state machine runs on the actuator node to asses the validity of the arriving control signal packets.

The controller and the actuator nodes are time based and run at the same sampling period with the sensor node. A data packet is disregarded by the controller and the actuator nodes depending on its arrival time. The nodes check if the data packet arrives before or after a pre determined decision time within the sampling interval. This pre determined decision time for the controller t_{DC} and the actuator t_{DA} can be calculated as in (3.10) and (3.11).

$$t_{DA}(t_k) = t_{k+1} - \tau_a(t_k) \tag{3.10}$$

$$t_{DC}(t_k) = t_{DA}(t_k) - \tau_c(t_k) - \tau_{pd} \tag{3.11}$$

where t_{k+1} is the beginning of the next sampling interval, τ_{pd} is the average time delay of data transmission in the network, $\tau_a(t_k)$ and $\tau_c(t_k)$ are the delays associated with the actuator and the controller node respectively.

3.5. Actuator Node

The actuator node is responsible for receiving the control signal packets from the controller node, assessing the validity of the received packets, selecting the appropriate ones and applying them to the plant. The actuator node is time based and runs a periodic task that checks for a received control signal packet at the beginning of every period. The actuator node is an intelligent unit that determines which control signal to apply to the plant by using a state machine to make this selection. The actuator node applies the actual control signal $u(t_k)$ received from the controller node if there is no packet loss and the SF =1 indicating that the packet is based on an actual plant state measurement. In the case of a packet loss, the actuator node uses the following state transition diagram to decide which control signal to apply to the plant, which is explained in Figure 3.1.

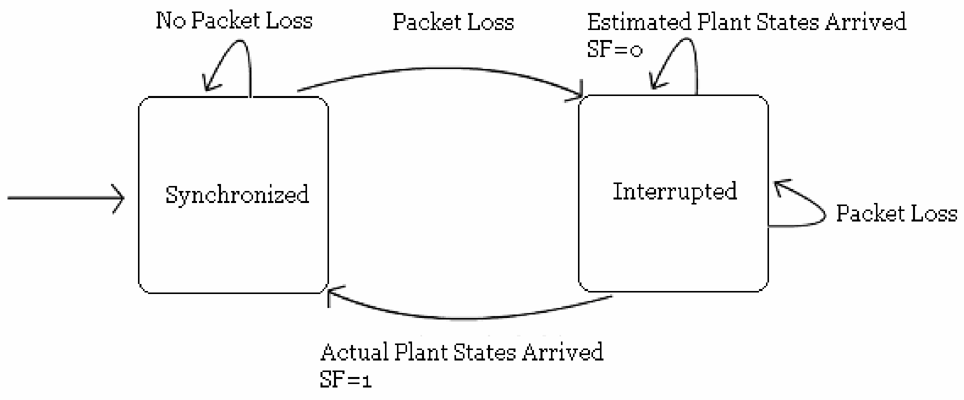

Figure 3.1 State Machine of the Actuator Node

The synchronization or loss thereof is sensed by the actuator using the SF flag in the control packet and the information of actual packet loss. The actuator node has two modes, the synchronized mode and the interrupted mode.

In the synchornized mode the states of the plant model are synchronized with the plant states. If SF =1 and the actuator node receives a control packet from the controller node when it is in the synchronized mode then it applies the first control output from that packet to the plant, which is $u(t_k,0)$. If the consecutive packets that the controller sends have SF switched from '1' to '0'; this indicates that the controller is not receiving actual plant states, but there is no controller to actuator data loss, then the actuator keeps applying the first control output from the received packets $u(t_{k+j},0)$. The actuator keeps applying the first output because in this situation the controller makes the assumption that the network is conveying the calculated control signal to the actuator node properly and are being applied to the plant and the actuator node stays in synchronized mode. If data is lost due to network delay or packet loss, the actuator node enters the interrupted mode.

When the actuator enters the interrupted mode the actuator node applies the control signal $\hat{u}(t_k,i), i=1,2,3\ldots$ to the plant until the last sample is reached or communication is restored. However, if one of the control packets received in this mode has SF =0 indicating that the controller is using state estimates based on the wrong assumption of applied control signal as stated above, then the packet is rejected. In the interrupted state, packets based on estimated states are rejected even if they are received without delay and the actuator stays in the interrupted state. If the actuator is still in the interupted state after the last prediction $\hat{u}(t_k,n)$ is reached without the communication being restored, the output is kept constant at that value thereafter. In order for the actuator to enter the synchronized mode it has to receive a control packet with SF =1.

All of the computer nodes in MBPNCS are time based and intelligent systems. All computer nodes run periodic tasks as a computational model. Packet loss between the sensor node and the actuator node is compensated at the controller node by predictions calculated by the model in the controller and packet loss between the controller node and the actuator node is compensated at the actuator node by usage of a selection algorithm based on the state machine and predicted control outputs. Late arriving packets are discarded in this work and no retransmission is done. A time synchronizing method is assumed to be used among the computer nodes. This is not a strong assumption because the network is generally pyhsically small and the amount of synchronization accuracy is comparable to the sampling time[17].

Chapter 4

METHODS AND APPARATUS

This research aims to verify that we can implement MBPNCS using an observer and verify that we can control an open loop unstable plant. In order to achieve this aim we have conducted experiments in our laboratory environment. First, MBPNCS is tested with an inverted pendulum plant to show that MBPNCS is efficient with an open loop unstable plant. Next, MBPNCS is tested on a DC motor with a Luenberger observer to verify that MBPNCS is efficient with an observer in the system. MBPNCS is designed to work in an industrial environment, thus multiple systems should be implementable on a common network. Thus, a final experiment is carried out by connecting two separate MBPNCSs with separate DC motor with a Luenberger observer plants to the same Ethernet hub to show that two separate MBPNCSs are implementable on a common network. The plants will be explained in detail in the subsequent chapters, and the NCS setup is common to both plants.

4.1. The Inverted Pendulum

The performance of the MBPNCS is verified through experiments with a real inverted pendulum. The inverted pendulum is a nonlinear system that enables us to see an open loop unstable controllable system to be tested on the MBPNCS. The inverted pendulum is more sensitive to the control method than the DC motor since it is open loop unstable.

An inverted pendulum is frequently used in the demonstration of controlling an unstable system. The inverted pendulum consists of a pole that has mass on its top and has a pivot attached to a laterally moving cart. It is controlled to keep it in the upright direction. In other words the inverted pendulum has two degrees of freedom; the angle of the rod and the position of the cart, but the input is the force acting sideways on the cart. The inverted pendulum is linearized around the upright position by assuming that the angle of the rod makes only small perturbations.

4.1.1 The Inverted Pendulum System Model

The inverted pendulum has two equilibrium points, one being stable and the other being unstable. The stable equilibrium corresponds to the rod pointing downwards toward gravity and making a -90 degree with the plane of the cart. This equilibrium point being stable means that the rod will return to this position in the absence of any control and force acting on the cart. The stable equilibrium requires no control input to be achieved thus, is uninteresting from a control perspective. The unstable equilibrium corresponds to a state in which the pendulum points strictly upwards and, thus, requires a control force to maintain this position. The basic control objective of the inverted pendulum problem is to maintain the unstable equilibrium position when the pendulum initially starts in an upright position at rest [18].

In order to design the control to be applied to inverted pendulum, first the system model should be derived. The system model of the inverted pendulum can be derived using the Lagrange equations or free body diagrams taking Figure 4.1 for reference [19].

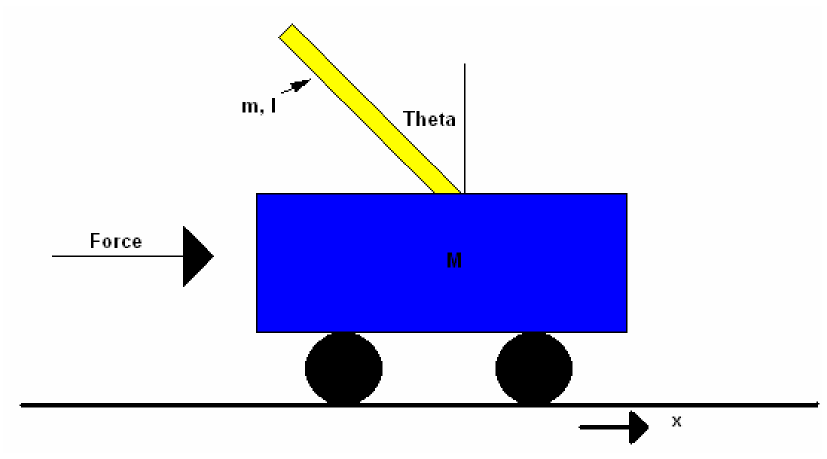

Figure 4.1 Basic Inverted Pendulum Diagram

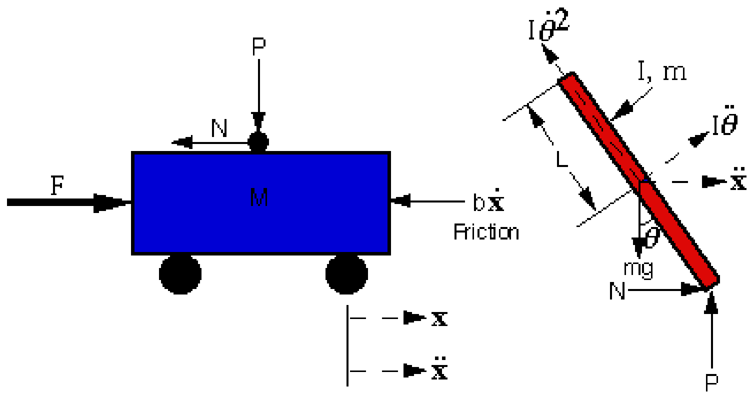

Figure 4.2 Inverted Pendulum Free Body Diagram

Summing the forces in the horizontal direction of the cart the following equation is obtained:

$$M\ddot{x} + b\dot{x} + N = F \tag{4.1}$$

Summing the forces in the horizontal direction of the pole the following equation is obtained:

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2 \sin\theta \tag{4.2}$$

Substituting the second equation into the first equation we get the following:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2 \sin\theta = F \tag{4.3}$$

Summing the forces perpendicular to the pendulum, we get the following equation:

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{4.4}$$

Summing the moments around the center of mass of the pendulum, we get the following equation:

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{4.5}$$

Combining these two equations, we get the following equation:

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{4.6}$$

In order to work with linear functions, this set of equations should be linearized about $\theta = \pi$. Assume that $\theta = \pi$ +ø where ø represents a small angle. Therefore, *cos(*θ*) = -1, sin(*θ*) =* -ϕ, and $\hat{\hat{\theta}}$ *= 0*. After linearization the two equations of motion become:

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{4.7}$$

$$(M+m)\ddot{x}+b\dot{x}-ml\ddot{\phi}=u \tag{4.8}$$

The state sapce representation of the dynamics of the inverted pendulum is:

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & \dfrac{-(I+ml^2)b}{I(M+m)+Mml^2} & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \dfrac{-mlb}{I(M+m)+Mml^2} & \dfrac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ \dfrac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \dfrac{ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}u \tag{4.9}$$

4.1.2 Chassis

An aluminum chassis was designed and built for the purpose of carrying out the inverted pendulum experiments. The plane of the cart is 1 meters long and the cart is positioned on the plane via a toothed belt. The trigger belt is positioned on the plane via two pulleys. The cart runs on a round rail with radial ball bearings. The chassis is given in Figure 4.3.

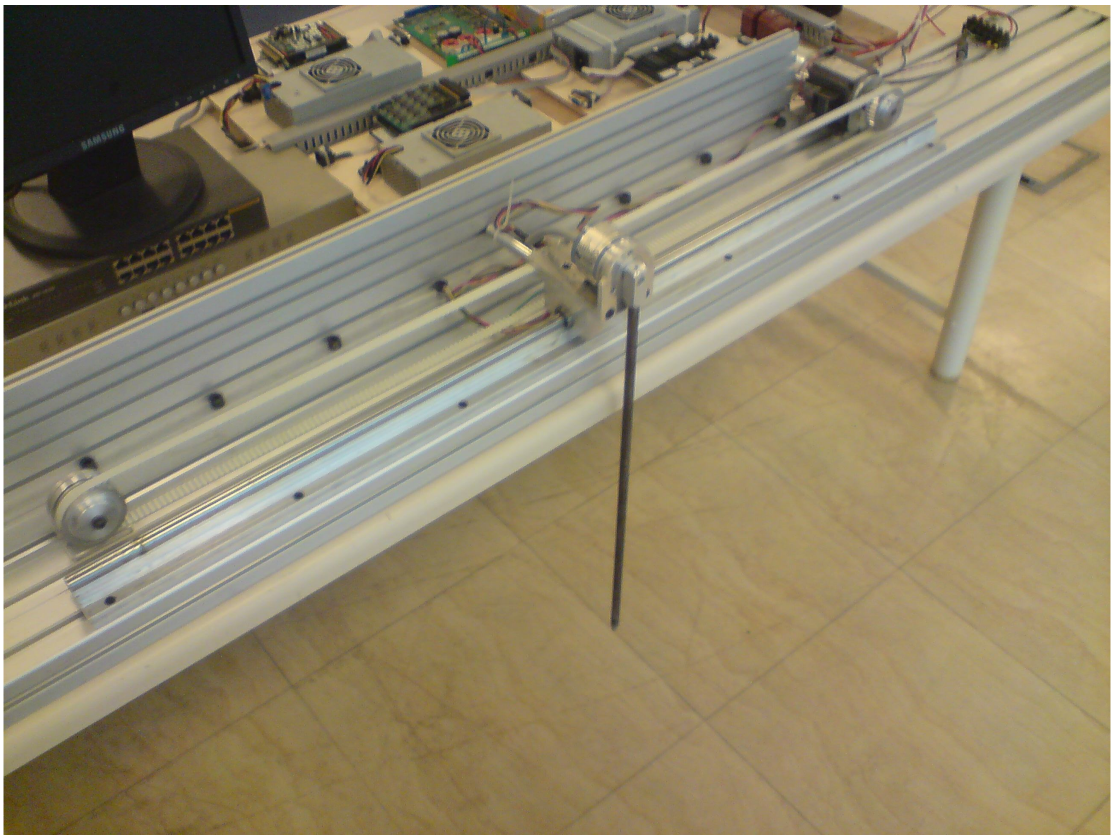

Figure 4.3 Inverted Pendulum Chassis

The below parameters are measured and used with the state space model and discretized:

M = .552 [kg]	Mass of the Cart
m = 0.0825 [kg]	Mass of the Rod
b = 2.5 [N/m/sec]	Friction
g = 9.8 [N/kg]	Gravity
l = 0.25 [m]	Length to Rod Center of Mass

$$A = \begin{bmatrix} 1 & 0.0010 & 0 & 0 \\ 0 & 0.9955 & 0.0013 & 0 \\ 0 & 0 & 1.0 & 0.0010 \\ 0 & -0.0165 & 0.0411 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0.0018 \\ 0 \\ 0.0066 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

4.2. Observer Based DC Motor Control with MBPNCS

The stability of the MBPNCS is experimented with a DC motor to verify theoretical results. The DC motor is a simple and easy to use control plant that can be controlled with speed and position values. This research focuses on the speed control of DC motor and applies state feedback algorithm with a Luenberger observer. Luenberger observer is used to compensate for a lack of current sensor and allows for a better verification of performance in MBPNCS.

In this research two DC motor control setups were built and experimented. First, one Luenberger observer based DC Motor control was implemented and tested. Second, two Luenberger observer based DC Motor controls were implemented and tested where each setup communicates on the same communication medium.

4.2.1 The DC Motor System Model

DC motor is a simple and common actuated plant in the control systems. Voltage provide to the DC motor provides rotary motion and the electrical modelling and the free body diagram of the motor is shown in (4.4) and (4.5) [19]:

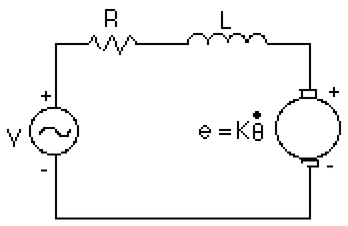

Figure 4.4 DC Motor Electrical Modeling

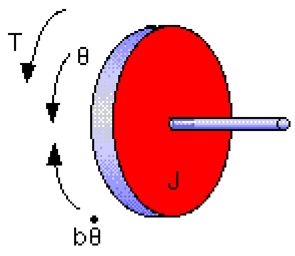

Figure 4.5 DC Motor Free Body Diagram

Since MBPNCS requires model of the plant to reside in the controller, the DC motor modelling is required.

The DC motor has a torque, τ, which corresponds to the armature current, *i*, explained in the following equation:

$$T = K_t i \tag{4.10}$$

where K_t is the torque constant.

The back EMF, *e*, is related to the rotational velocity by the following equation:

$$e = K_e \dot{\theta} \tag{4.11}$$

In SI units torque constant, K_t, is equal to voltage constant K_e. This constant is called electromotive force constant and the following equation holds true:

$$K = K_e = K_t \tag{4.12}$$

From the free body diagram of the motor and using the Newton's law the following equation is obtained:

$$J\ddot{\theta} + b\dot{\theta} = Ki \tag{4.13}$$

Using the electrical model of the motor and using the Kirchhoff's law the following equation is obtained:

$$L\frac{di}{dt} + Ri = V - K\dot{\theta} \tag{4.14}$$

Using these two equations and the state space representation of the DC motor can be obtained. In the state-space form, the equations above can be expressed by choosing the rotational speed and electric current as the state variables and the voltage as an input. The output is chosen to be the rotational speed and the following equations are obtained:

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix}-b/j & K/j \\ -K/L & -R/L\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ 1/L\end{bmatrix}V \tag{4.15}$$

$$\dot{\theta} = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} \tag{4.16}$$

4.3. Implementation of the Experimental Setup

4.3.1 The Computer Hardware of Sensor, Controller, Actuator:

Each sensor, controller and actuator node reside in a separate PC 104 type computer that is equipped with a 300 MHz AMD Geocode processor. The codes are written and compiled in a Linux server and the PC 104's use RT-Linux as the operating system. RTLinux is necessary to guarantee that the periodic tasks of the sensor, controller and actuator work in a real time environment. Calculations and previous simulations in Matlab show that the discrete control system with 10-3 seconds of sampling time works under stable conditions [1]. The setup of the MBPNCS is given in Figure 4.6.

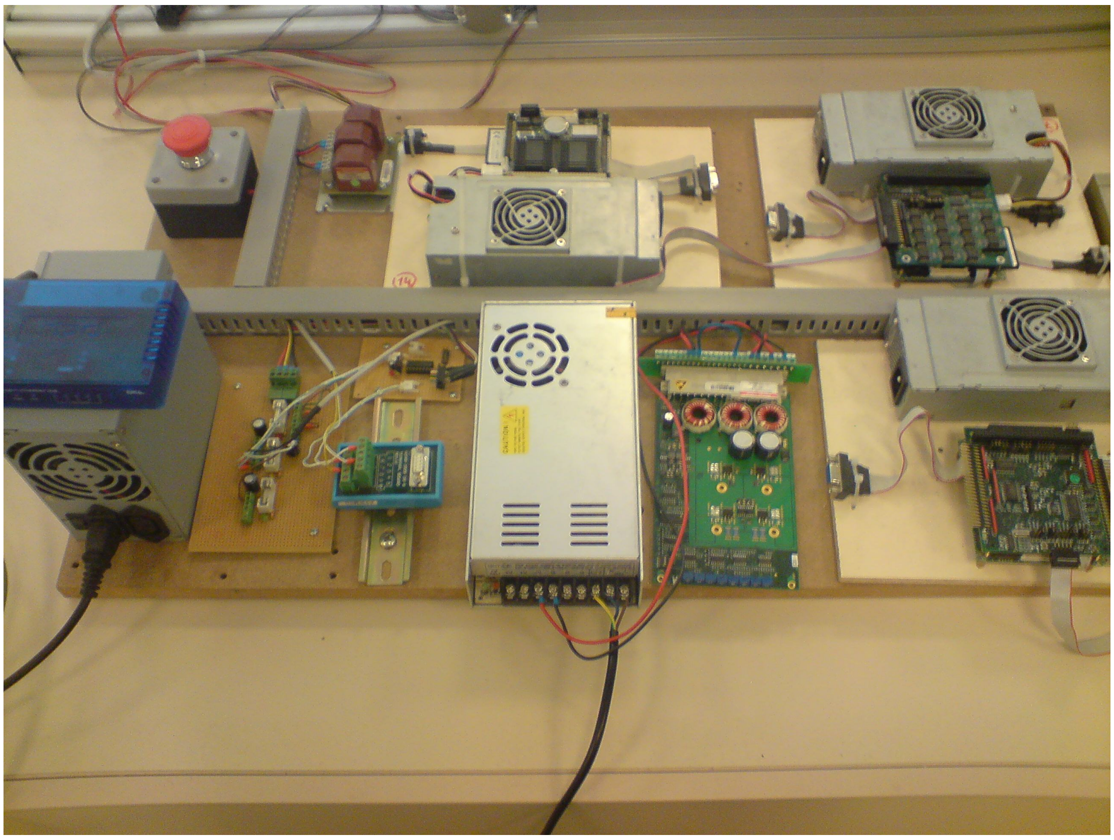

Figure 4.6 MBPNCS Setup

4.3.2 AD & DA Converter

Analog to digital and digital to analog conversions were performed using a Kontron ADIO 128. Kontron ADIO128 is a 12 bit module is used by the actuator to drive the plant with the control signal sent by the AD/DA. The digital to analog converter function of the Kontron ADIO128 is used for this purpose. Kontron ADIO128 is able to create an output voltage between -10 and 10 volts which is used as a reference to the motor drivers. The driver software of the Kontron ADIO128 is prepared and run as a kernel driver inside the actuator. This software was written by the project team.

4.3.3 Quadrature Decoder and Encoder

Shaft angles of the motors and the pole are measured using quadrature encoders. Sensor node is equipped with a quadrature decoder to acquire the position information from the plant. MSI P400 with fifteen input channels is used for this purpose. The kernel driver is prepared and placed in the sensor node since this device also does not have any Linux driver published. An encoder is mounted on the cart of the inverted pendulum and the cart moves on the aluminum platform via a toothed belt. The encoder takes in the angle of the rod in units of radians, 0 radian means the rod is standing up and makes a 90 degree angle with the platform. Another encoder is used for calculating the position of the cart. This encoder calculates the position in terms of meters, a 0 m means the cart is on the initial position. The two sensor data is referenced with 0 to calculate the error value in the system.

4.4. TrueTime

MBPNCS performance in the above mentioned applications was simulated in the computer environment using TrueTime, which is a Matlab toolbox developed by Henriksson, Cervin and Arzen [20][21]. TrueTime is a MATLAB/SIMULINK based tool used to simulate networked embedded systems. The tool can be used to create low level of instruction and simulations can be done on the instruction execution level and network communication can be done on the wanted transport level. This allows for user to choose the execution time of every instruction and also assign execution times to individual code blocks. 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. In this research, the chosen communication network was a model 100BaseT Ethernet with suitable packet loss and delay rates and realistic transmission speeds.

The code and algorithms developed under TrueTime can be directly exported to the actual implementation of digital control systems. The user is able to choose the type of scheduling algorithm applied on the simulated computer by TrueTime such as rate monotonic scheduling algorithm. Different standard network protocols can also be tested using TrueTime making it easy to see and measure their influence on networked control system. In the simulation the application level code of the sensor, controller and actuator nodes were written in the ‘C’ code, ‘m’ code of Matlab and Simulink blocks to implement the desired algorithms to be performed by network nodes at the kernel were also used.

4.5. Motors

A Minertia J Series motor is used to drive the cart of the pendulum. Some parameters of the motor such as winding resistance and the torque constant K_t of the motor should be known in order to drive the cart with the correct voltage value. Since there is no reference published by the manufacturer, the torque constant K_t is measured. A pulley setup is prepared to calculate the torque of the motor. The torque of the motor is given as:

$$\tau = mgr = iK_t \tag{4.17}$$

where τ is the torque of the motor, m is is the mass applied to the motor and r is radius of the pulley. A known weight, such as 1 kg is applied to the pulley. An ampermeter is put in series between the motor and the power supply. Voltage is started to be fed slowly to the motor. At the exact instant when the motor stops the current of the motor is noted. The torque constant K_t is measured to be 0.1767 Nm/Amp. This motor was driven using a Maxon motor driver, in torque control mode.

A Maxon motor type of 144501 is used for in the DC motor experiments and motor parameters published by the manufacturer were used.

L= 3.16131e-3 [Henri]	Terminal inductance
Kt= 118.54e-3 [Nm/A]	Torque constant
R= 11.80 [Ohm]	Terminal resistance
b= 2.1008e-006 [Nms/rad]	Friction
j = 6.2800e-006 [kgm^2]	Rotor inertia

The above parameters are measured and used with the state space model and the following model is found after discretization:

$$A = \begin{bmatrix} 0.8663 & 4.4722 \\ -0.0088 & -0.0180 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.1315 \\ 0.0750 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$D = 0$$

4.6. Computer Network

In order to experiment with network problems such as packet loss and delay, a non-reliable network environment is chosen. Ethernet is chosen as the communication network since it is widely used and is supported by PC 104. A typical non switching hub is used as the connection point of the nodes.

4.6.1 Random Number Generator

A random number generator is used to simulate dropped packets. By using a random number generator we are able to determine the rate at which packet loss occurs. The used random number generator is of type Linear Congruential Generator (LCG) and uses the following equation:

$$X_{n+1} = (aX_n + c) \bmod m \tag{4.18}$$

where X_n is the array of random values, m is the modulus, a is the multiplier, c is the increment and X_0 is the starting value.

In this research the below parameters are used for the LCG as proposed by Press, Teukolsky, Vetterling and Flannery[22]:

m= 2^{32}

a= 1664525

c= 1013904223

$X_0 = 1$

Chapter 5

THEORETICAL STABILITY

The stability for MBPNCS can be proven by showing that the control based on state estimates during intervals of disturbance in the network is stable and state estimates do not deviate from actual states. MBPNCS updates the state variables at the beginning of the sampling time, thus estimated data is reset to actual data in random integer multiples of the sampling period. The stability of the MBPNCS can be proven in the Lyapunov sense if a suitable Lyapunov equation can be discovered as explained below.

To derive a stability criterion, it will be shown that during the intervals when transmitted data is not delayed or lost, the system behaves as a normal digital control system, and during the intervals when the communication is interrupted, the state estimates do not deviate significantly from the actual states, thus control based on the state estimates does not jeopardize stability. The existing results of Montestruque and Antsaklis can be applied to MBPNCS with some modifications[14][15][17].

Montestruque and Antsaklis proposed that the stability of their NCS can be proven by using the following procedure. In an ideal simple NCS with no delay and packet loss, the state estimate $\hat{x}$ is updated when the actual state variable x of the plant is received by the controller node in the sampling period; $t_k = t_o + kh$, k=0, 1, ... where h is the sampling period. The dynamics of the plant is $\dot{x} = Ax + Bu$ and the dynamics of the model is $\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u$, but containing some modeling error. This research uses state feedback control in order to control the plant; $u = K\hat{x}$. We assume that there exists some modeling error, $\widetilde{A} = A - \hat{A}$ and $\widetilde{B} = B - \hat{B}$. Therefore the dynamics of the overall system can be written using the augmented state vector $z = [x^T e^T]^T$ where state error is $e = x - \hat{x}$ and the augmented system dynamics can be represented as $\dot{z} = \Lambda z$ where Λ is given by (5.1):

$$\begin{bmatrix} A + BK & -BK \\ \widetilde{A} + \widetilde{B}K & \hat{A} - \widetilde{B}K \end{bmatrix} \tag{5.1}$$

If one uses an ideal NCS with no delay and packet loss, and the state esimate $\hat{x}$ is reset to the actual state value x after each state update which occurs at every sampling interval. After each update the error component $e = x - \hat{x}$ becomes zero. Since the error term becomes zero in every status update the state response in between the state updates, for $t_k \le t \le t_{k+1}$, *z(t)* can be represented as:

$$z(t) = e^{\Lambda(t-t_k)} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)^k z_0 \tag{5.2}$$

where t_k is the last update time and $z_0 = z(t_0)$ is the initial value.

Taking the norm of each term on both sides of the equation above, a limiting factor for $z(t)$ can be found and this factor is shown below:

$$\|z(t)\| = \| \; e^{\Lambda(t-t_k)} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)^k z_0 \; \| \le \left\| e^{\Lambda(t-t_k)} \right\| \left\| M^k \right\| \left\| z_0 \right\| \tag{5.3}$$

Since the terms $\left\| e^{\Lambda(t-t_k)} \right\|$ and $\|z_0\|$ are limited the state response of the system $z(t)$ is proved to be globally exponentially stable around the solution $z = [0 \quad 0]^T$ if the eigenvalues of the matrix *M* which is shown next are strictly inside the unit circle.

$$M = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \tag{5.4}$$

However, this proof is shown for non-random update times and should be expanded for random update times for MBPNCS. If the update time is random $(h(j) \in [h_{\min}, h_{\max}])$, the system has the following response $z(t)$:

$$z(t) = e^{\Lambda(t-t_k)} \prod_{j=1}^{k} M(j) z_0 \tag{5.5}$$

where M is shown below:

$$M = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \tag{5.6}$$

The stability of such system can be shown by using the Lyapunov theory where the system with state response $z(t)$ shown in (5.5) is asymptotically stable for $h \in [h_{\min,} h_{\max}]$ if a positive definitive matrix X exists such that $X = MXM^T = Q$ is also positive define for all $h \in [h_{\min,} h_{\max}]$.

Using the footsteps of this logic, the stability of the MBPNCS can also be proved using a similar approach. MBPNCS is system where the state update is not done periodically, but only after the connection is restored between the sensor and the controller or controller and the actuator, which happens in the arbitrary integer multiples of the update interval *h*. Modifying (5.5) we get the following state response equation for the MBPNCS:

$$z(t) = e^{\Lambda(t-t_k)} \prod_{i=1}^{l} M(a_i) z_0 \tag{5.7}$$

where:

$$M(a_i) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda a_i h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \tag{5.8}$$

where: $a_i \in (1,2,3,...,a_{\max})$

The initial state response is z_0 and during the interval (t_0, t_k), there may be a communication breakdown that span from time h to $a_{max}h$. However, these communication breakdowns can happen only in finite number of intervals due to the nature of the MBPNCS.

For a given MBPNCS system $h_{\min}$ can be defined by the minimum packet transmission latency and $h_{\max}$ can either be the maximum network delay, or left open as a condition of stability. In both cases, there are a finite number of update intervals for which the stability condition must be checked. The number of predictions necessary in open loop stable plants can be related to the settling time of the plant when disturbances are small[17].

Thus, if a Lyapunov equation can be found in the sense that $X = MXM^T = Q$

where:

$$M(a_i) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda a_i h} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$$

where: $a_i \in (1,2,3,...,a_{\max})$

$a_{\max}$ will give us the maximum number of consecutive packet loss the system can tolerate.

The effect of number of predictions n in the system should also be emphasized briefly. The maximum number of packet loss the system can tolerate $a_{\max}$ will be compensated with the number of predictions in the system which equals n. Thus, if $a_{\max} < n$ the MBPNCS can be stated to be Lyapunov stable. In this thesis, n is chosen to be a sufficiently large value that can't be depleted even with % 99.8 packet losses. However, the amount of time required to calculate n number of predictions should not exceed sampling time of the system. Thus, the stability of the MBPNCS is related to the number of predictions in the system by the following equation:

$$a_{\max} < \mathrm{n} < n_{\max} \tag{5.9}$$

where $n_{\max}$ equals to the number of maximum predictions that can be calculated in each sampling time T_S.

Chapter 6

6. RESULTS

The purpose of this research is to verify that MBPNCS holds performance and stability with

a) A control plant with an observer

b) An open loop unstable control plant and that MBPNCS is suitable for industrial applications.

In order to reach this aim this thesis focused on implementing the MBPNCS with an inverted pendulum and a Luenberger observer based DC motor. Simulations are carried out in MATLAB Simulink and TrueTime toolbox for simulating network communication and the real time computers. The performance of MBPNCS is measured with respect to the loss over the network. Experiments are carried out in two MBPNCS setups one for the inverted pendulum and the other for the DC Motor control. The experiments run stochastic test programs that increment packet loss with sampling periods. Test programs increment the packet loss percentage in pre-determined intervals and after each increment the MBPNCS is initialized. This procedure is valuable in observing the effects of increasing packet loss and delay in the network. The packet loss is simulated by a random number generator that drops packets by the help of a threshold value as explained in chapter 4.7.1. As stated in chapter 4.3.1 MBPNCS is stable with a sampling time of 10^{-3} seconds, thus experiments use this value as sampling time.

The experiment results of the inverted pendulum are benchmarked with the experiment results of a basic Network Controlled System (bNCS) to identify the improvements MBPNCS offer over conventional NCSs. bNCS is an event based NCS, where the sensor node periodically samples and sends plant states to the controller node and the controller and actuator nodes produce output only when they receive data. The bNCS model has no intelligent units in the computer nodes and is prone to problems associated with packet loss and delay in the network. The bNCS is also run with stochastic test programs that increment packet loss with sampling periods. Test programs increment the packet loss percentage in pre-determined intervals and after each increment the bNCS is initialized.

A performance metric for the setup is necessary to be able to objectively compare the MBPNCS with bNCS. One suitable metric for comparing the performance of MBPNCS and bNCS is Root Mean Square (RMS) error. The formula for calculating the RMS error is shown in (6.1).

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - ref)^2}{n}} \tag{6.1}$$

where y_i is the plant output and *ref* is the reference given to the plant, at every sampling time; iT_s*;* where i = 1, 2, ….n.

6.1 Observer Based DC Motor Control Experiment

A Luenberger observer based DC motor control experiment is conducted to verify that MBPNCS holds performance, in the case where the plant state vector can not be measured but the output of the plant is measurable and the plant is observable. This test is valuable since MBPNCS should hold performance and stability in the case where all state variables are not measurable which is common in industrial applications. This test is prepared to verify that MBPNCS would be successful in industrial applications.

This experiment runs with a sampling period of 10^{-3} seconds. The model used in this experiment was stated in chapter 4.2.1. The control algorithm applied is state feed back control. The DC motor speed reference toggles between 50 and 0 rpm in every 1000 milliseconds. The picture of the setup is given in Figure 6.1.

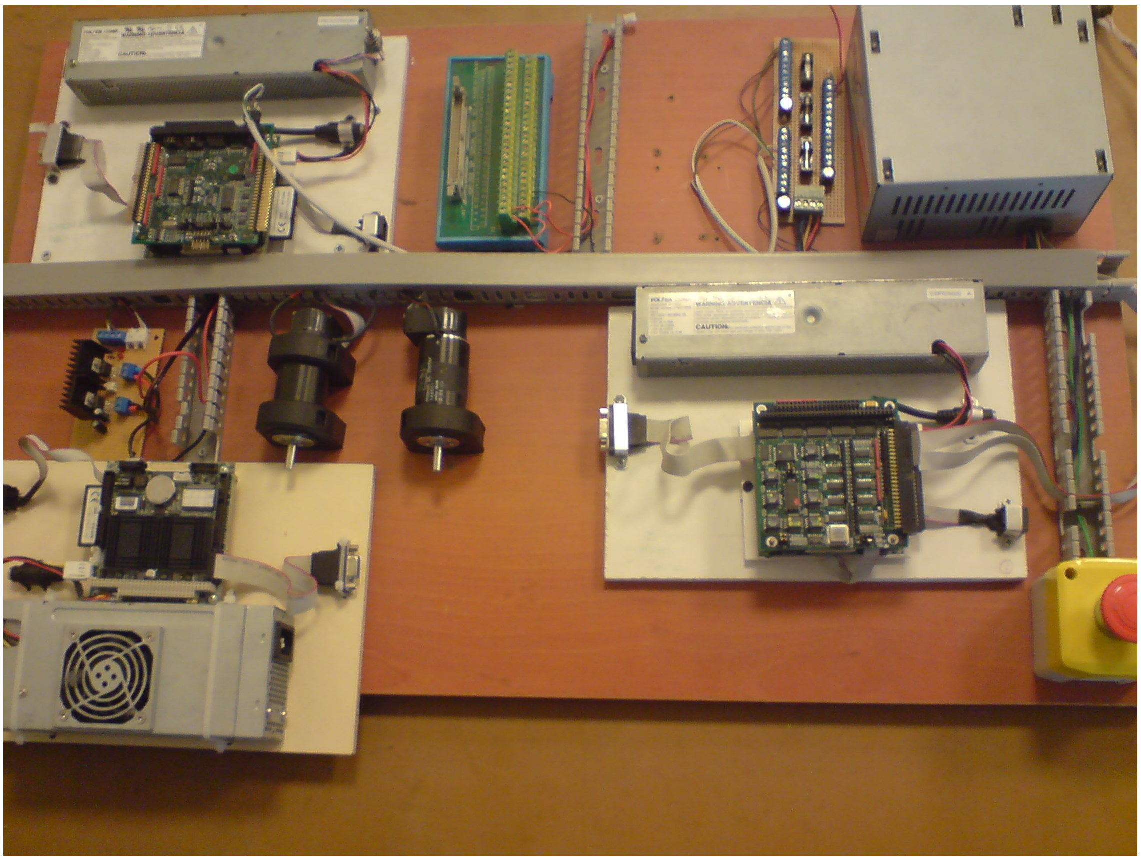

Figure 6.1 Setup of the DC motor MBPNCS

The RMS error performance of MBPNCS with an observer is shown in Figure 6.2. In this figure y axis is the RMS Error in speed of the motor calculated with (6.1) and x axis is the packet loss percentages in the MBPNCS.

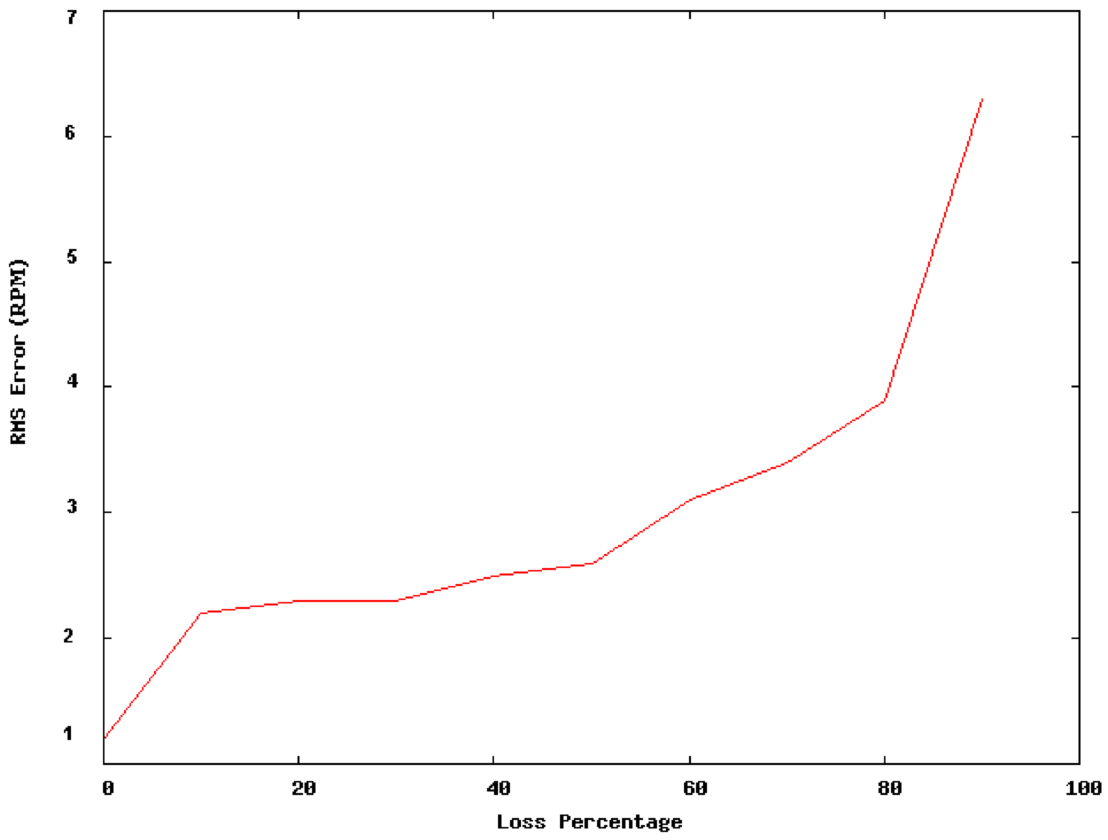

Figure 6.2 RMS Error of MBPNCS in DC motor control with Observer

Figure 6.2 shows that the MBPNCS can support stability with packet losses up to %90. The MBPNCS can sustain its performance up to %80 packet loss and degrades in performance with higher rates of packet loss. Detailed view of MBPNCS performance with packet loss rates of %0, %30, %50, %70 and %90 are depicted in Figures 6.3, 6.4, 6.5, 6.6 and 6.7.

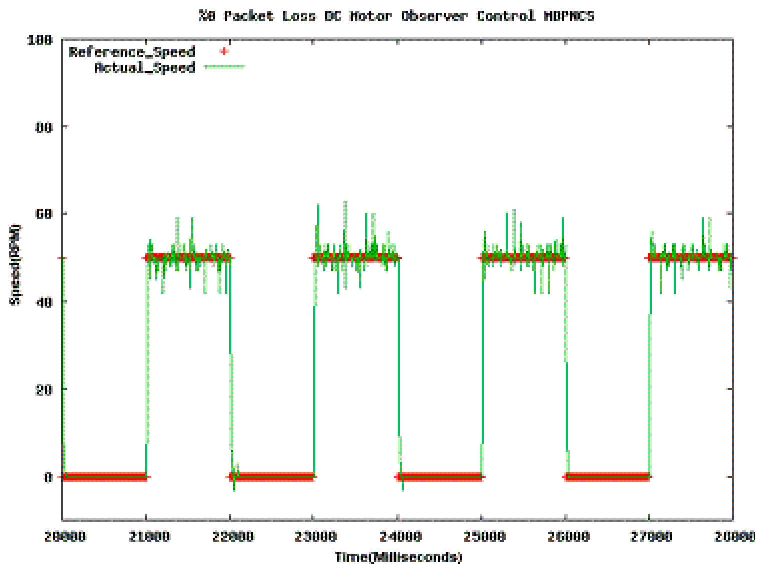

Figure 6.3 Time Graph %0 Loss –Observer DC Motor

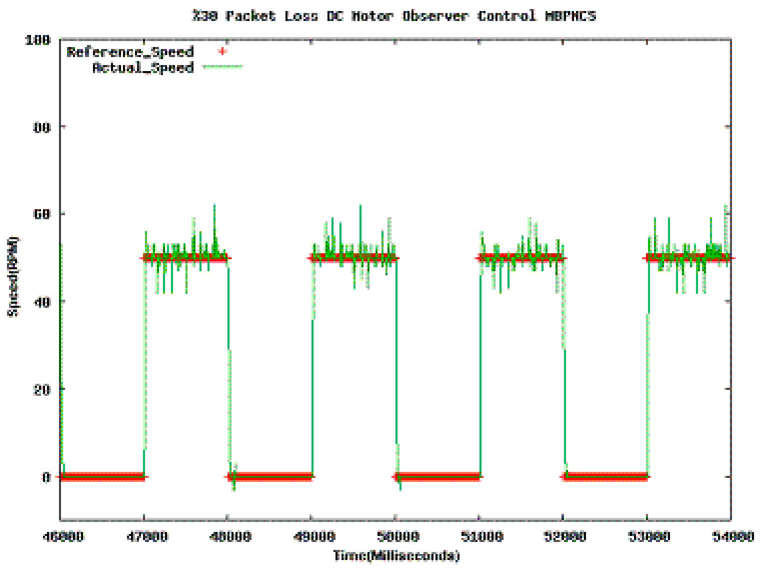

Figure 6.4 Time Graph %30 Loss –Observer DC Motor

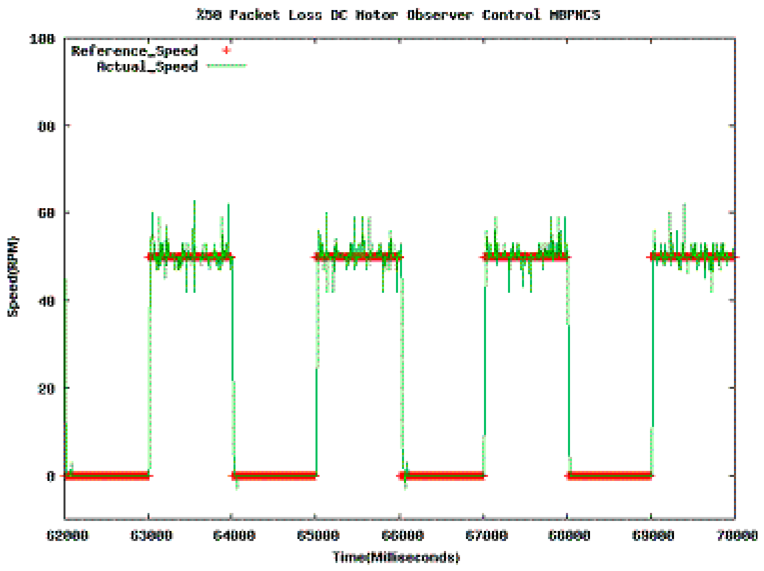

Figure 6.5 Time Graph %50 Loss –Observer DC Motor

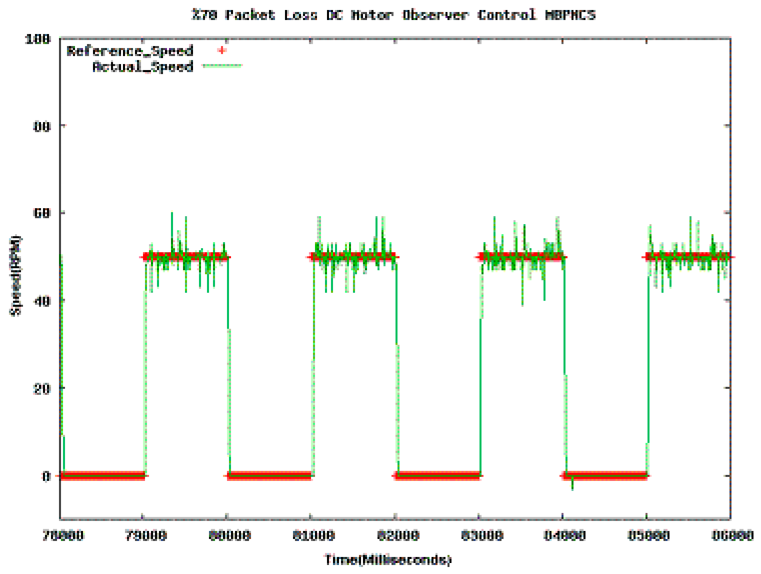

Figure 6.6 Time Graph %70 Loss –Observer DC Motor

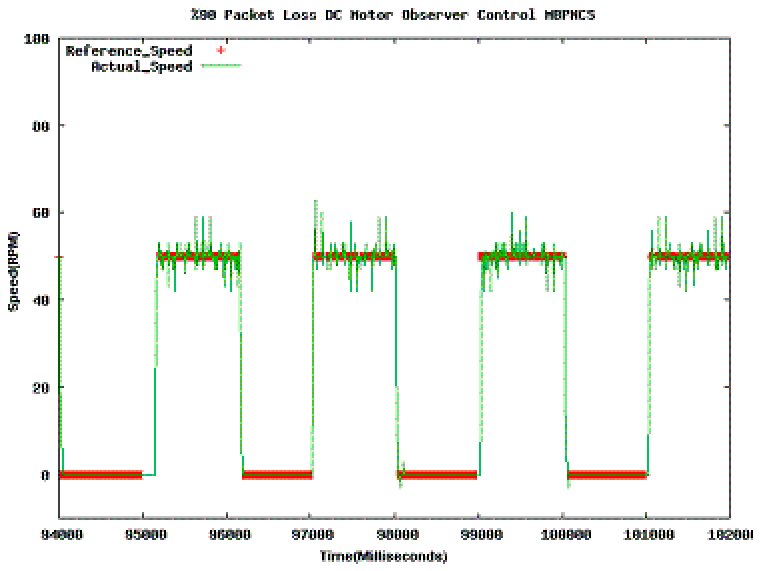

Figure 6.7 Time Graph %90 Loss –Observer DC Motor

In Figures 6.3 through 6.7, y axis is the speed of the motor in RPM units and x axis is the time in milliseconds. Figure 6.3, 6.4, 6.5 and 6.6 support our previous conclusion that the MBPNCS holds performance up to % 90 packet loss. The negative effect of increasing loss percentage is not visible until Figure 6.7 which shows packet loss of % 90. The MBPNCS works with good performance and holds stability with an observer up to %90 packet loss.

6.2 Dual Observer Based DC Motor Control Experiment

As stated before, MBPNCS is designed work in industrial applications. In order to work in the industrial applications, several MBPNCS s should be able to work together in order to cover large physical spaces. Thus, we have implemented two MBPNCS s over o common Ethernet network to verify this usage. Two DC motors are implemented with a Luenberger observer as explained in Chapter 6.1 with both of them using the same network hub. In Figure 6.8, y axis is the RMS Error in speed of the motor calculated with 6.1 and x axis is the packet loss percentages in the MBPNCS.

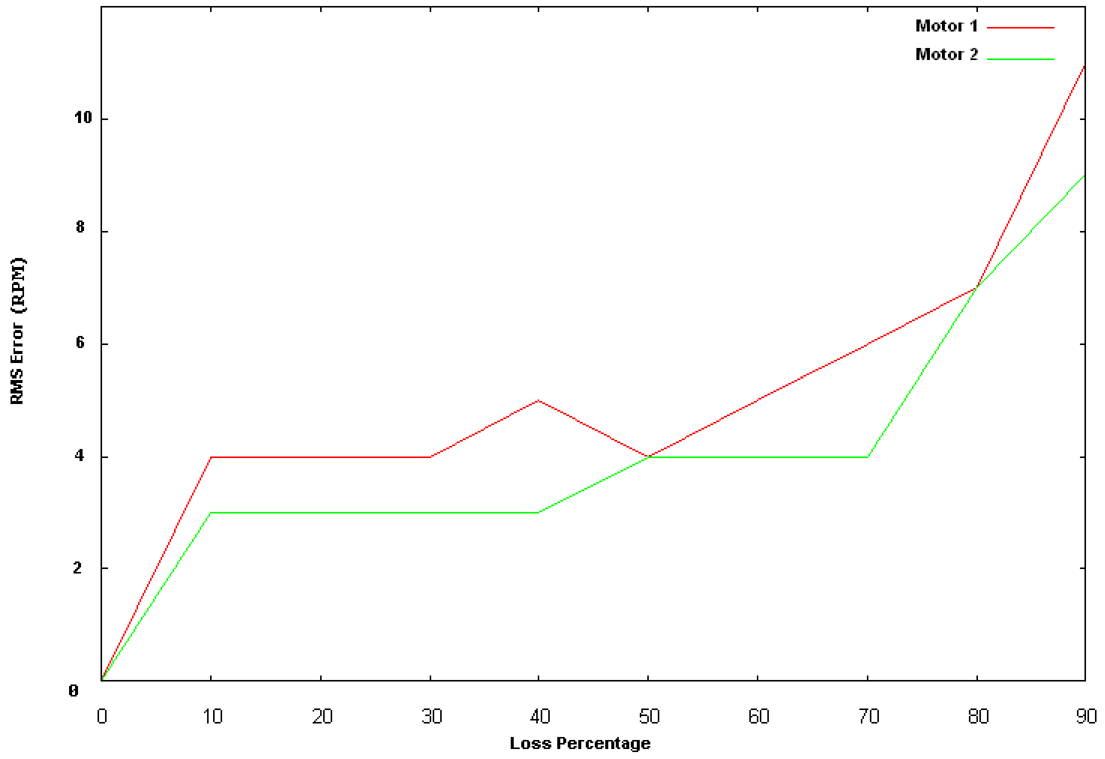

Figure 6.8 RMS Error of MBPNCS in Dual DC motor control with Observer

Figure 6.8 depicts that the two motors hold performance and stability up to %90 packet loss. This conclusion is concurrent with our previous conclusion in chapter 6.1. This conclusion is verification that several MBPNCS s can be used with a common network and thus is feasible for industrial applications. Time graphs of the two motors are separately given in Figure 6.9 to 6.16.

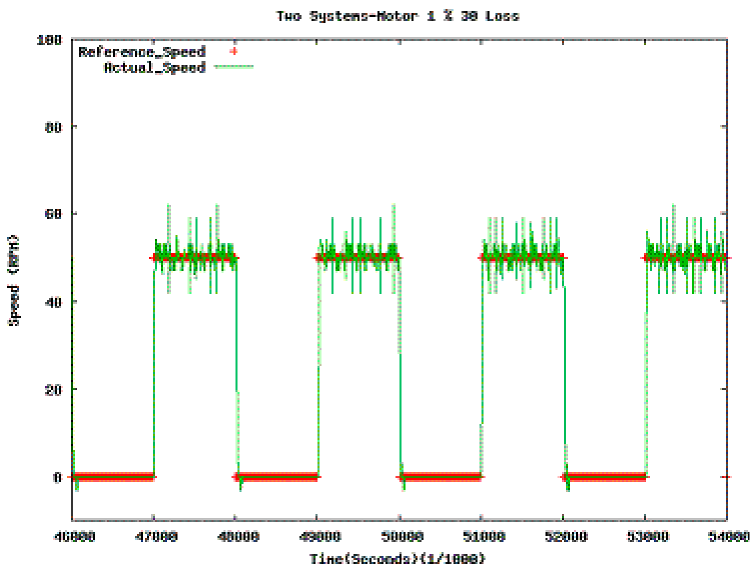

Figure 6.9 Time Graph %30 Loss –DC Motor 1

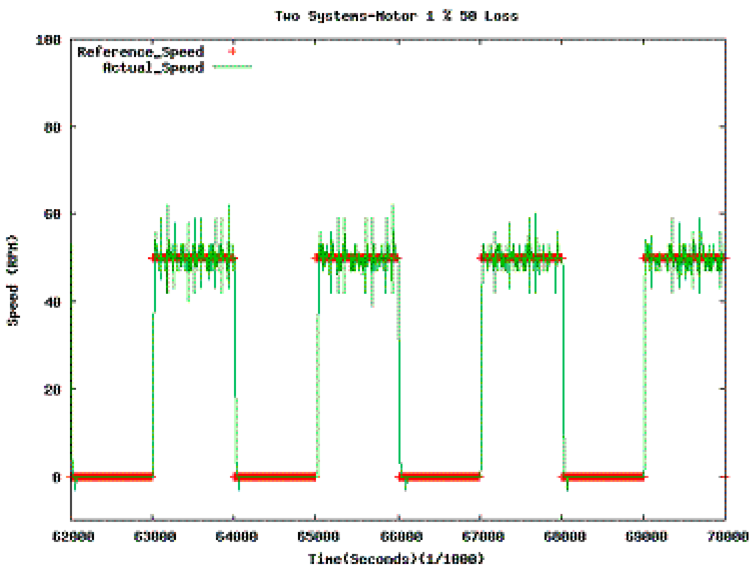

Figure 6.10 Time Graph %50 Loss –DC Motor 1

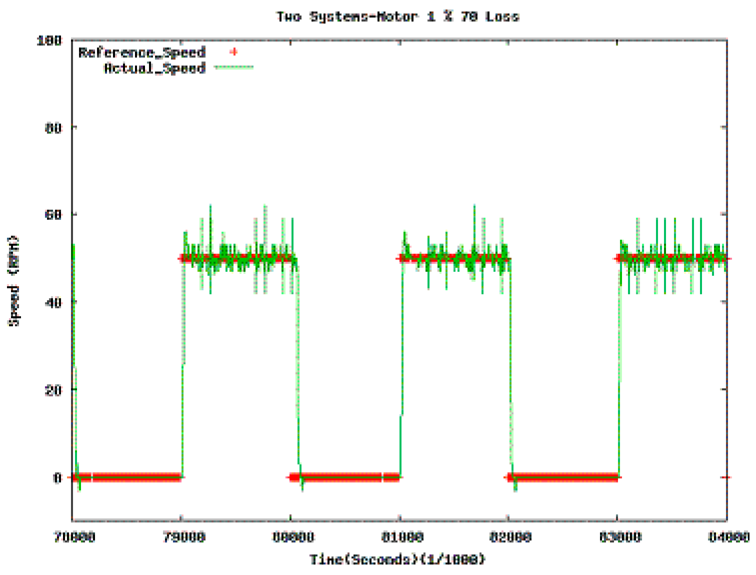

Figure 6.11 Time Graph %70 Loss –DC Motor 1

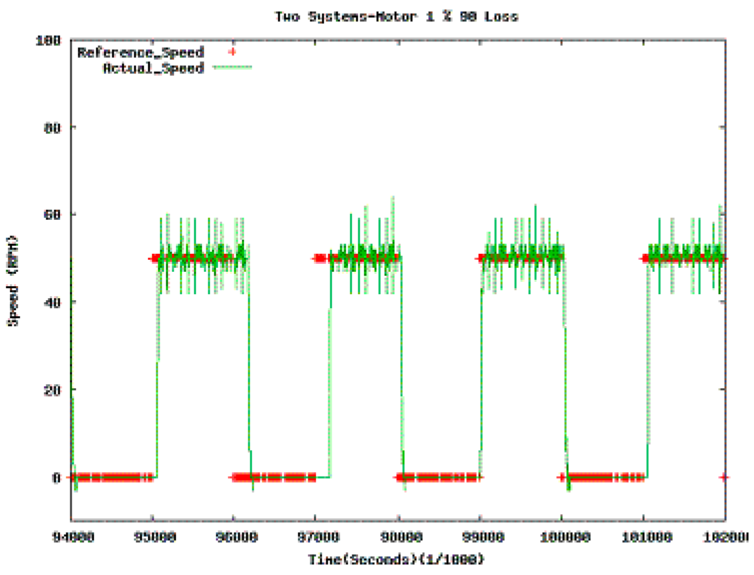

Figure 6.12 Time Graph %90 Loss –DC Motor 1

Figures 6.9 to 6.12 show that Motor 1 holds performance and stability with increasing packet loss in the system. MBPNCS has no degrading in performance with packet loss percentages of %30, %50, %70 and %90.

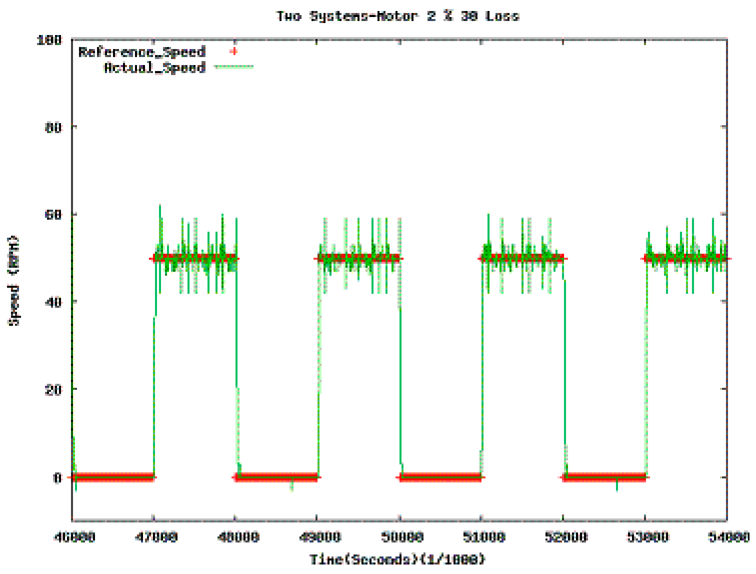

Figure 6.13 Time Graph %30 Loss –DC Motor 1

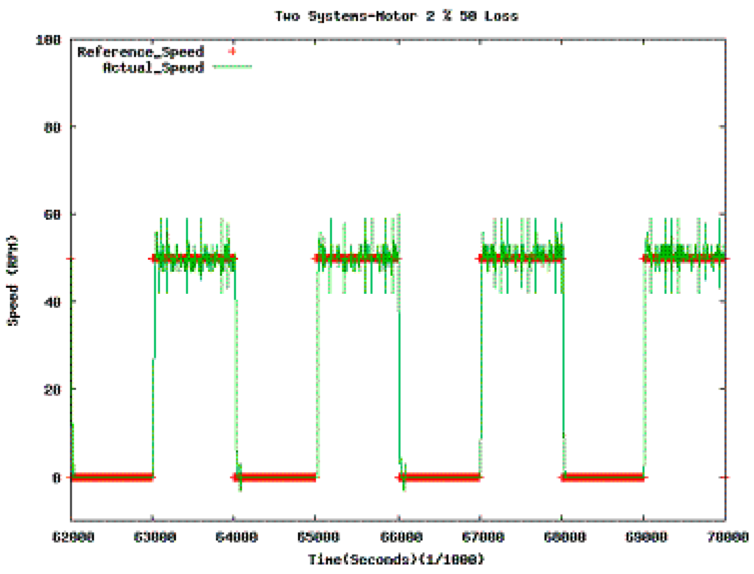

Figure 6.14 Time Graph %50 Loss –DC Motor 1

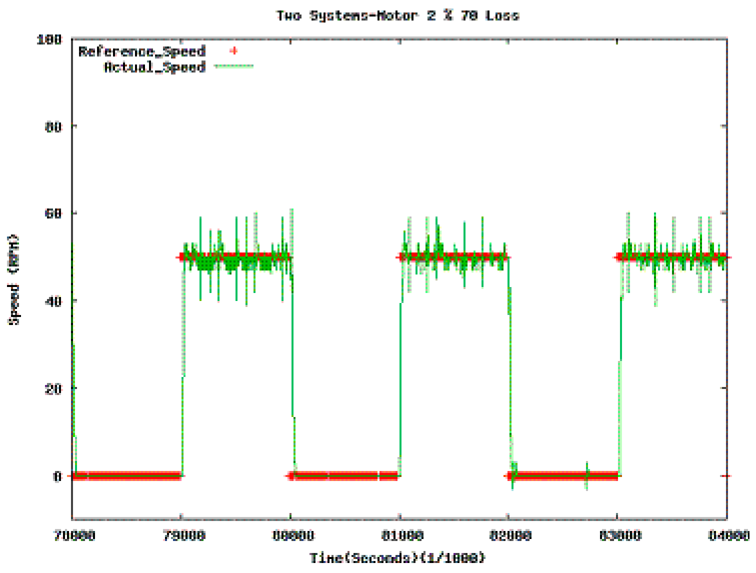

Figure 6.15 Time Graph %70 Loss –DC Motor 1

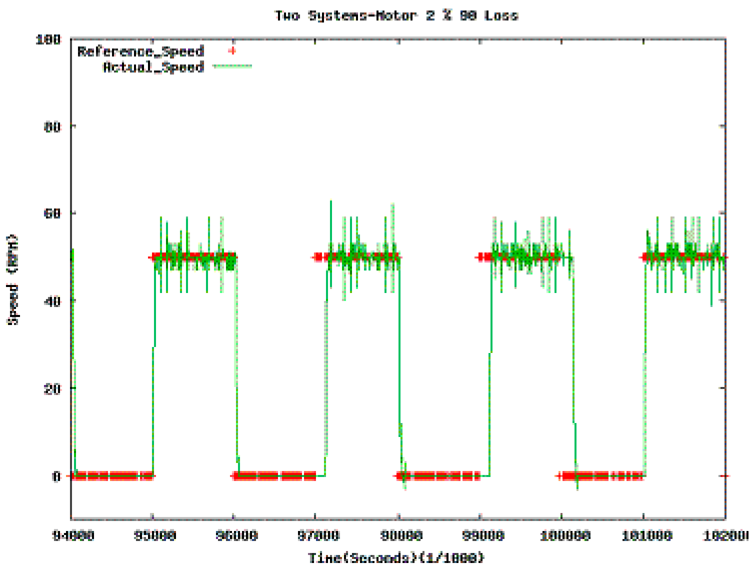

Figure 6.16 Time Graph %90 Loss –DC Motor 2

Figures 6.13 to 6.16 show that Motor 2 holds performance and stability with increasing packet loss in the system. MBPNCS has no degrading in performance with packet loss percentages of %30, %50, %70 and %90.

As it can be observed there is little degrading on the stability or the performance of neither motor with packet losses up to %90. This conclusion is verified with RMS error in the systems and Figure 6.8. This experiment is valuable in observing that using a common network communication has no degrading effect on the performance and the stability of multiple MBPNCS s. Not only MBPNCS can support control when all state variables are not measurable, but also it can support this performance when multiple systems work together. This experiment backs up our conclusion from chapter 6.1 that MBPNCS would be successful in industrial applications.

6.3 Inverted Pendulum Control Experiment

An inverted pendulum is built and controlled to verify that MBPNCS outperforms bNCS with an open loop unstable plant. Previous studies in MBPNCS [1][2] failed to show MBPNCS holds performance and stability with an open loop unstable plant. This research aimed in showing that MBPNCS is efficient with difficult plants, thus an inverted pendulum setup was built as explained in Chapter 4. Inverted pendulum has two state variables; position of the cart and the angle of the pole, as explained in Chapter 4.1.1. In order to verify the results, each state vector is figured and analyzed separately. The reference given to the cart and the pole of the inverted pendulum is always ‘0’.

This experiment runs with a sampling period of 10^{-3} seconds. The model used in this experiment was stated in chapter 4.1.1. The control algorithm applied is state feed back control.

6.3.1 Simulations

Model based predictive networked control system was simulated under TrueTime, with the inverted pendulum used as the plant. The sampling time of the systemis 0.01s, and a state feedback control is used. The communication network was a model of 100BaseT Ethernet with suitable packet loss and delay rates, and realistic transmission speeds. Figure 6.17 shows the MATLAB simulink block diagram of the setup. Our experimental setup was replicated in MATLAB simulink and the stochastic packet loss was simulated by TrueTime.

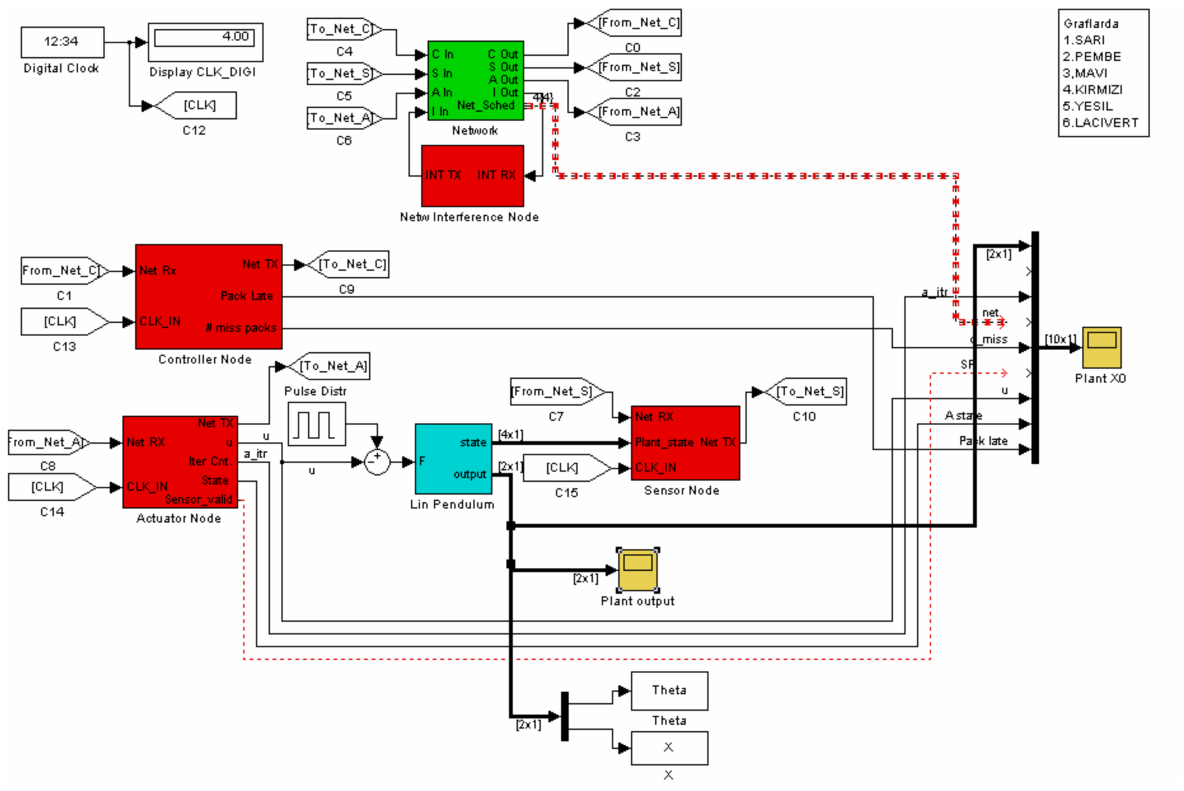

Figure 6.17 TrueTime simulation block diagram

The inverted pendulum was simulated with varying stochastic loss percentages. As it can been seen in the following RMS error graphs, Figure 6.18 and 6.19 simulations results verify that the MBPNCS can support an open loop unstable system up to %90 packet loss. The pole angle and cart position enter the stable region in less than 4 seconds with packet loss up to %90. The system is uncontrollable only when the system has % 90 packet losses. It should be noted that the RMS error in pole angle is graphed in terms of 10^{-3} radians and the RMS error in cart position is graphed in terms of millimeters.

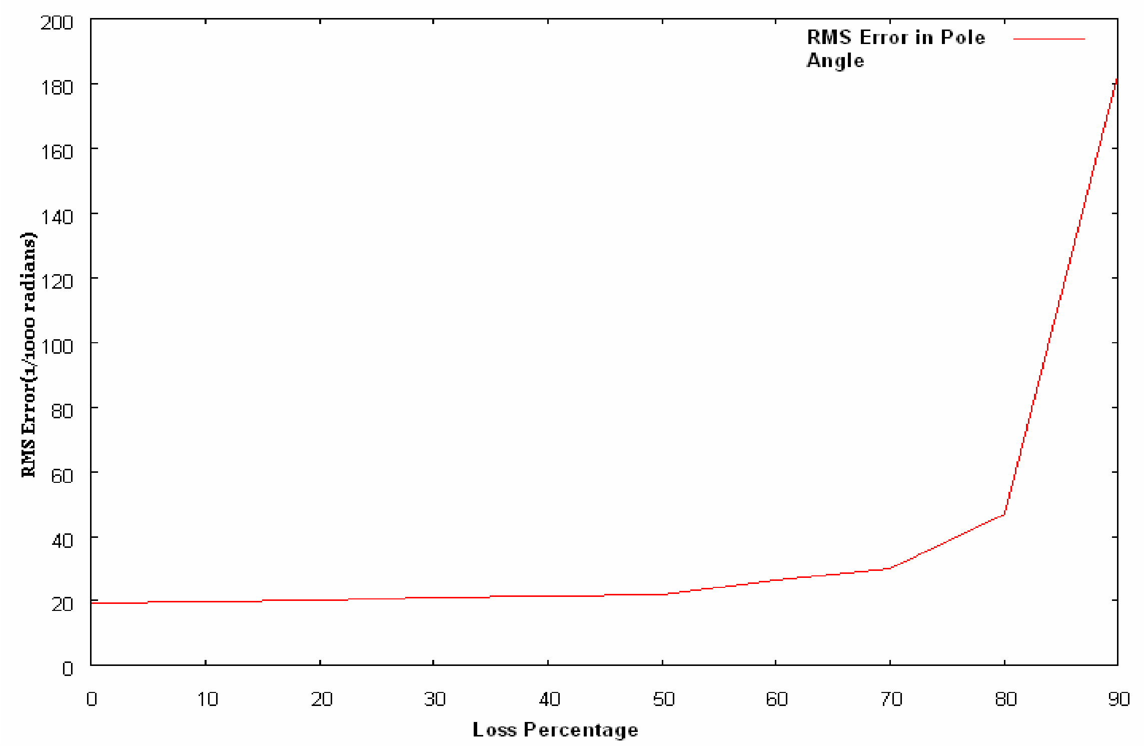

Figure 6.18 Simulated RMS Error in pole angle

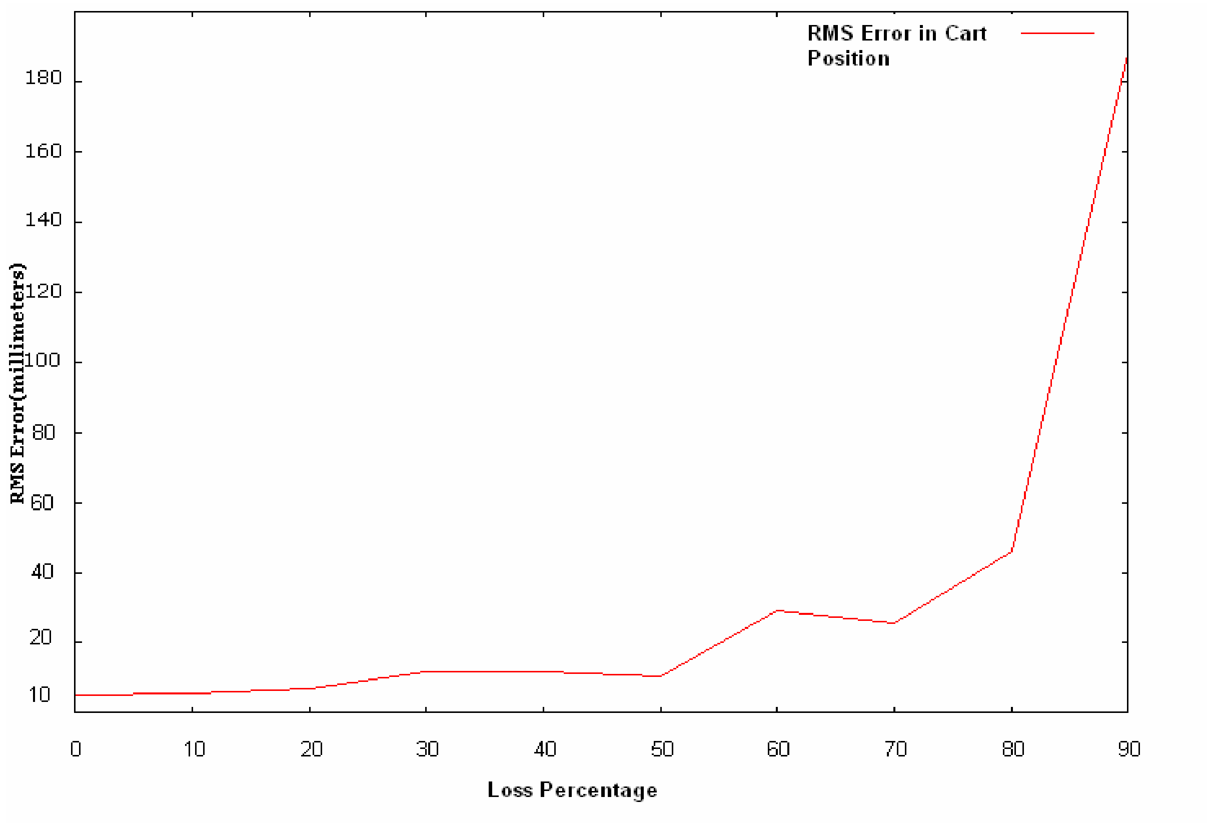

Figure 6.19 Simulated RMS Error in cart position

Simulating the inverted pendulum was necessary to confirm our experimental results are strong and reliable. As it will be seen in chapter 6.3.2 our simulated results concur with our experimental results.

6.3.2 Experimental Results

An experimental setup was built to verify our simulation results. A MBPNCS unit is connected to an inverted pendulum chassis. Test programs increment the stochastic packet loss in the system in pre determined intervals. Since the inverted pendulum state vector has two variables; pole angle and cart position, each variable should be analyzed separately. The experimental setup is given in Figure 6.20.

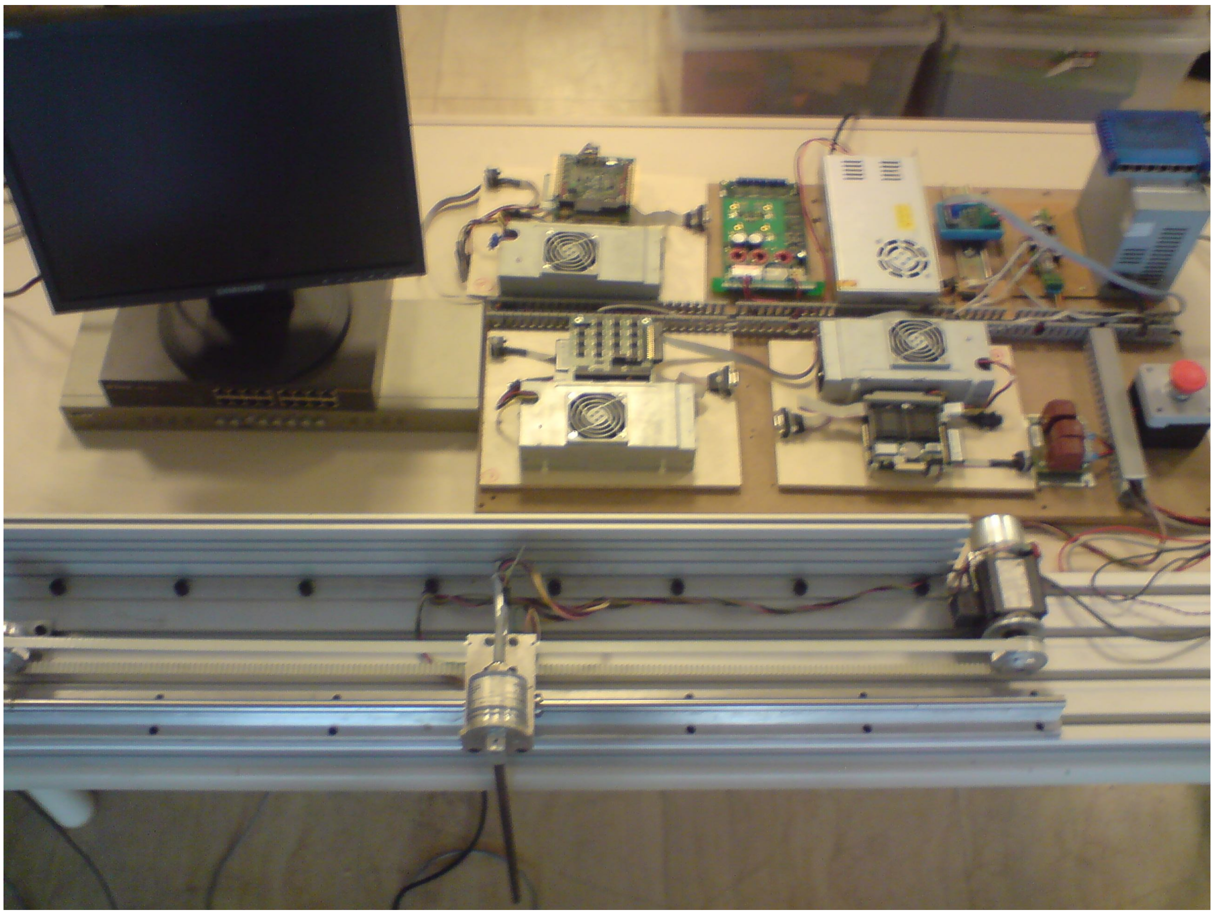

Figure 6.20 Setup of the inverted pendulum motor MBPNCS

The performance of MBPNCS is benchmarked with the performance of bNCS. The results show that MBPNCS outperforms bNCS in every packet loss percentage when used with an open loop unstable plant. This result can be seen in Figure 6.21 which shows RMS error in pole angle of MBPNCS vs. bNCS. It should be noted that the RMS error in pole angle is graphed in terms of 10^{-3} radians.

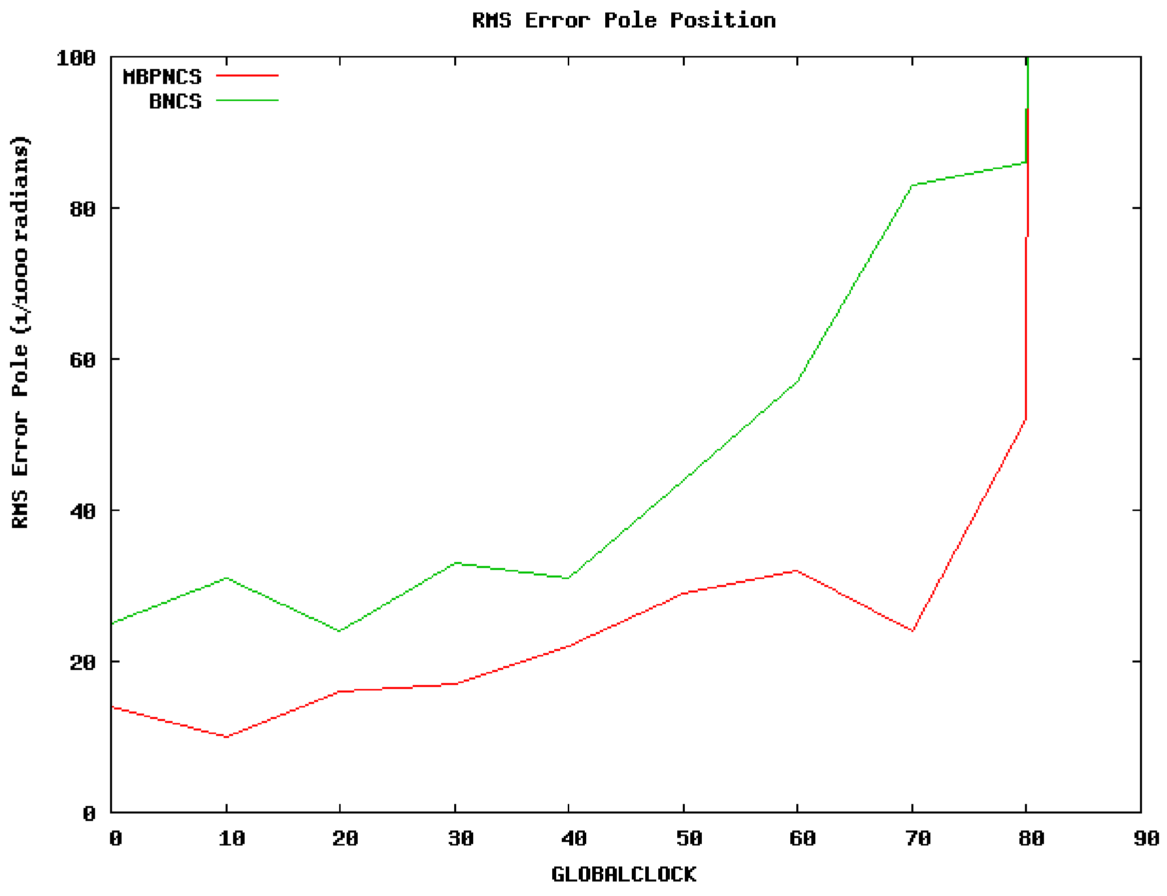

Figure 6.21 Experimented RMS error in Pole angle

It can be verified that MBPNCS outperforms bNCS in pole angle performance in every packet loss percentage. MBPNCS can support the inverted pendulum in an upright position with packet losses up to %80 but bNCS fails to achieve this.

Figure 6.22 shows RMS error in cart position of MBPNCS vs. bNCS. It should be noted that the RMS error in cart position is graphed in terms of millimeters.

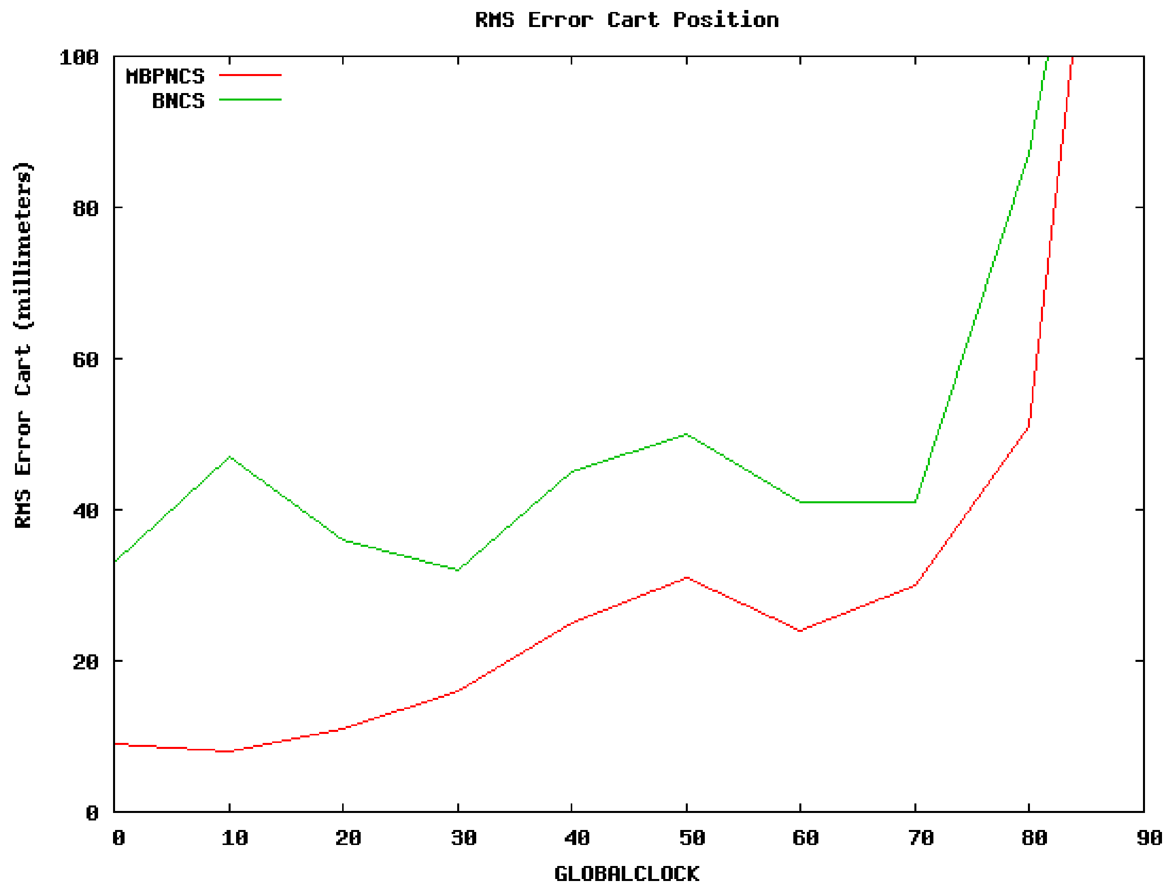

Figure 6.22 Experimented RMS error in cart position

The MBPNCS outperforms bNCS in RMS Error values in every loss percentage value. The MBPNCS can support stability up to %80 whereas bNCS can not. MBPNCS can support the inverted pendulum cart close to the starting point up to %80 but bNCS fails to achieve this.

In order to make a healthier observation of the performance of MBPNCS over bNCS, the detailed view of each individual stochastic packet loss interval is graphed. In each time graph the pole angle is graphed with units of 10^{-3} radians and the cart position is graphed in units of millimeters. It should be noted that the pole angle graphs has y axis from -0.314 to 0.314 radians. This is to show the relative position of the rod. Note that the rod in downright position makes -3.14 radians or 3.14 radians with the y axis. Allowed linear region for the rod to stay in upright position is -0.104 to 0.104 radians (-6 to 6 degrees).

Case 1 % 0 Loss Percentage

With % 0 loss percentage the performance of bNCS and MBPNCS are graphed in Figures 6.23 to 6.26.

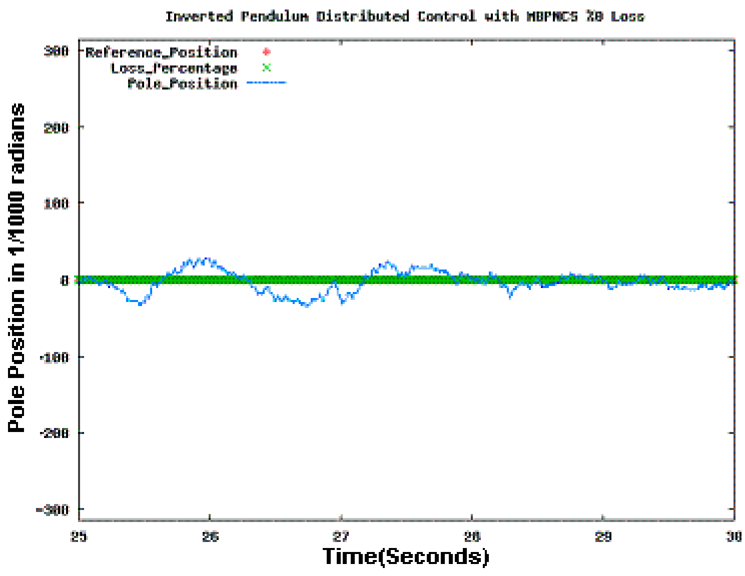

Figure 6.23 MBPNCS %0 Loss – Pole Angle

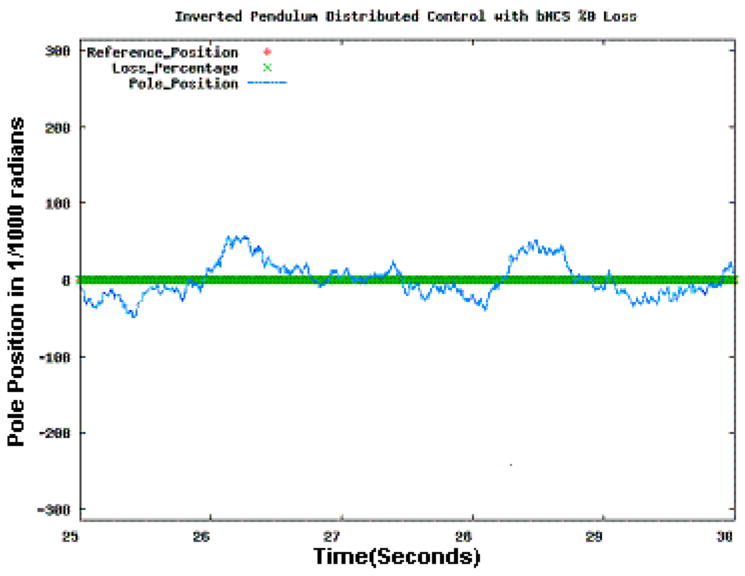

Figure 6.24 bNCS %0 Loss – Pole Angle

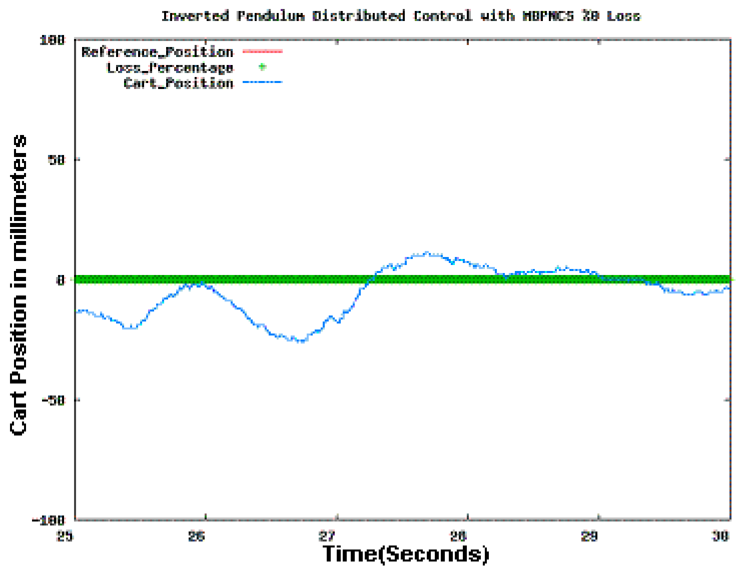

Figure 6.25 MBPNCS %0 Loss – Cart Position

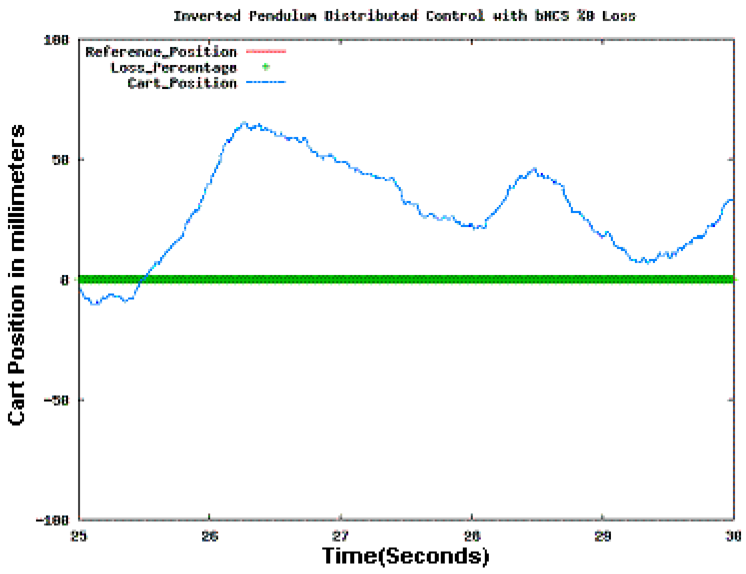

Figure 6.26 bNCS %0 Loss – Cart Position

MBPNCS shows similar performance over bNCS in %0 packet loss. Both systems achieve to hold the rod in an upright position and the cart on the initial starting position.

Case 2 % 30 Loss Percentage

With % 30 loss percentage the performance of bNCS and MBPNCS are graphed in Figures 6.27 to 6.30.

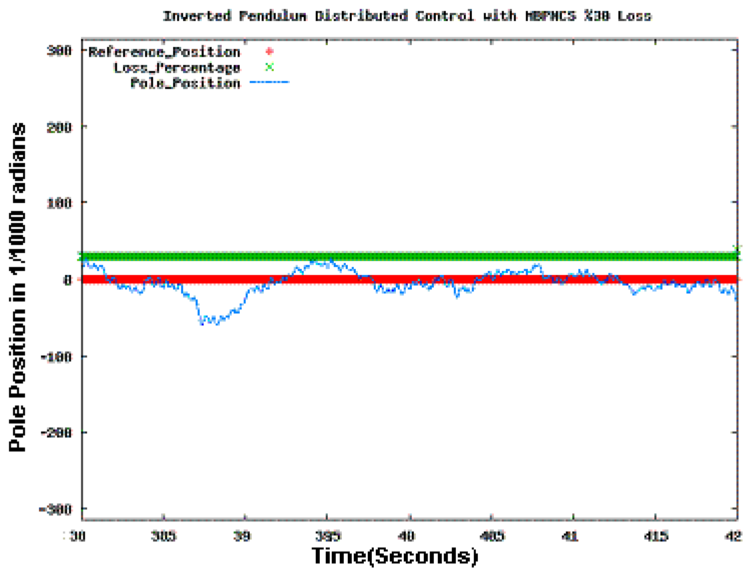

Figure 6.27 MBPNCS %30 Loss – Pole Angle

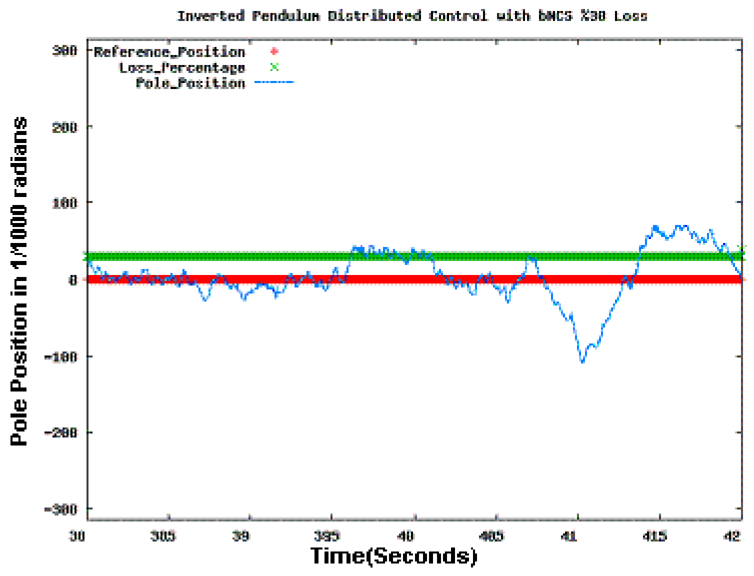

Figure 6.28 bNCS %30 Loss – Pole Angle

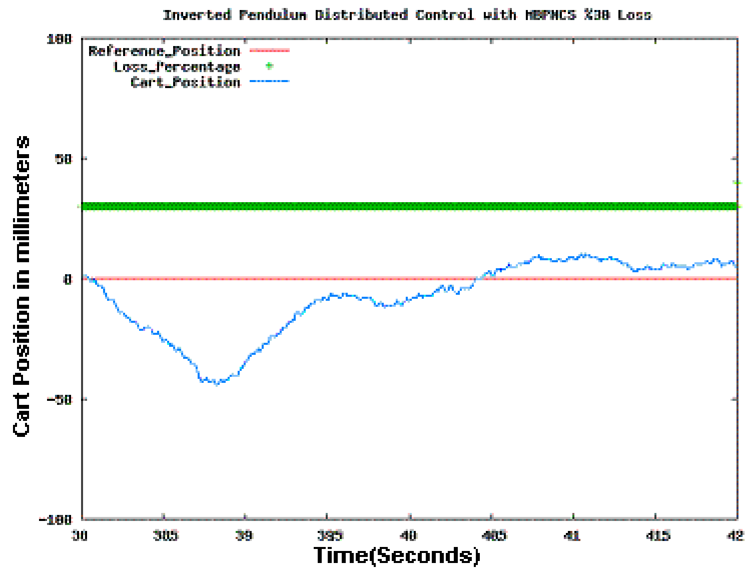

Figure 6.29 MBPNCS %30 Loss – Cart Position

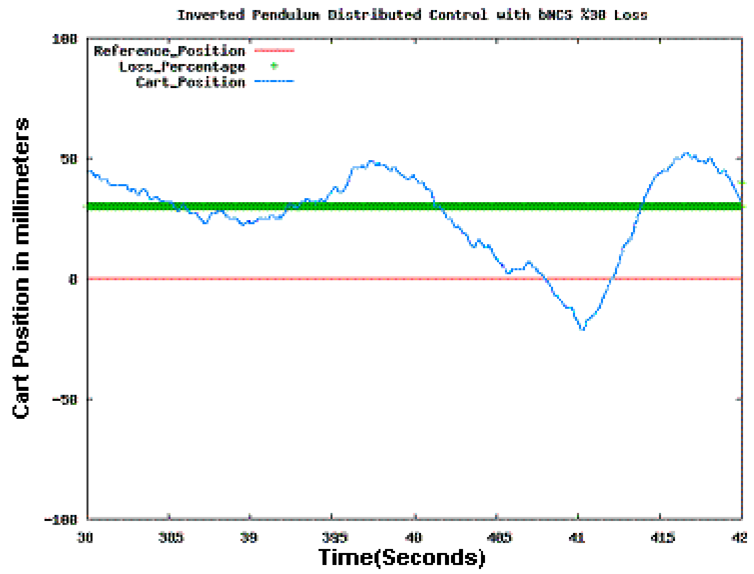

Figure 6.30 bNCS %30 Loss – Cart Position

MBPNCS starts to outperform bNCS in %30 packet loss. The rod is much more stable in MBPNCS and the cart does not diverge from the initial starting point as it does in bNCS.

Case 2 % 50 Loss Percentage

With % 50 loss percentage the performance of bNCS and MBPNCS are graphed in Figures 6.31 to 6.34.

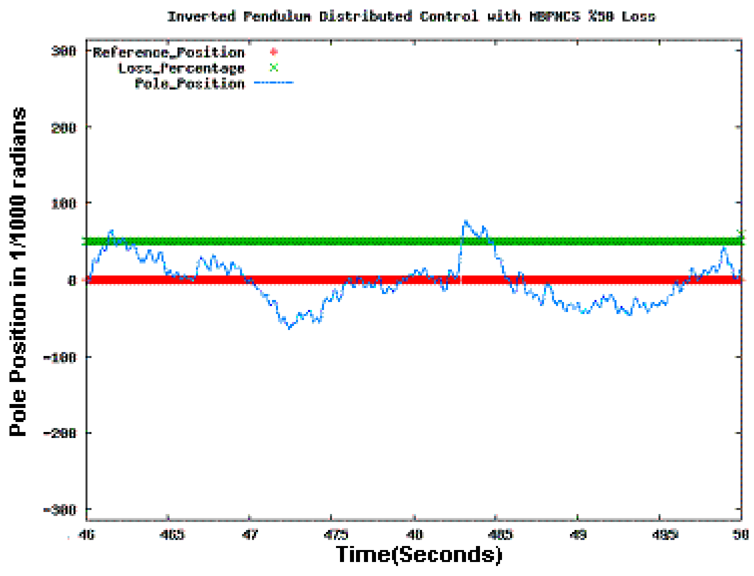

Figure 6.31 MBPNCS %50 Loss – Pole Angle

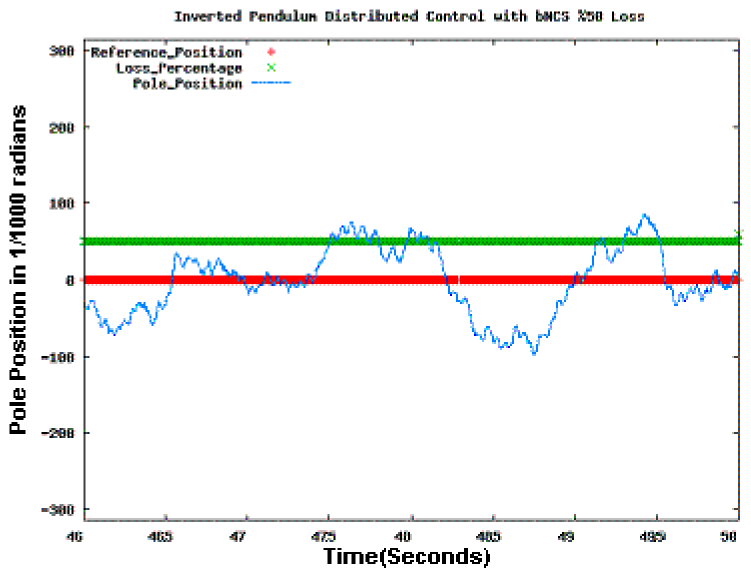

Figure 6.32 bNCS %50 Loss – Pole Angle

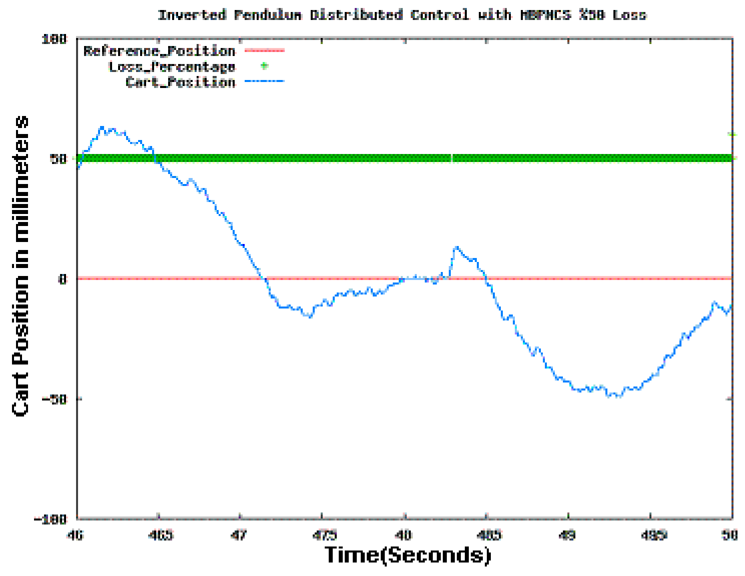

Figure 6.33 MBPNCS %50 Loss – Cart Position

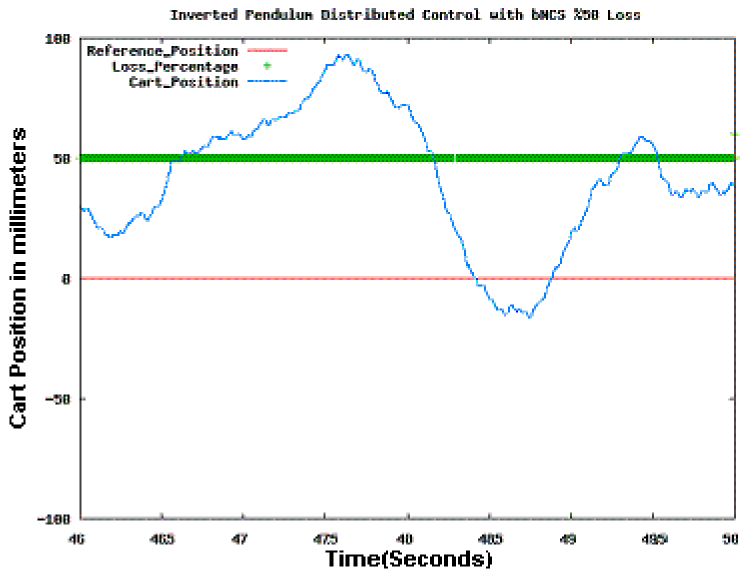

Figure 6.34 bNCS %50 Loss – Cart Position

MBPNCS outperforms bNCS in %50 packet loss. The rod is much more stable in MBPNCS and the cart does not diverge from the initial starting point as it does in bNCS.

Case 2 % 70 Loss Percentage

With % 70 loss percentage the performance of bNCS and MBPNCS are graphed in Figures 6.35 to 6.38.

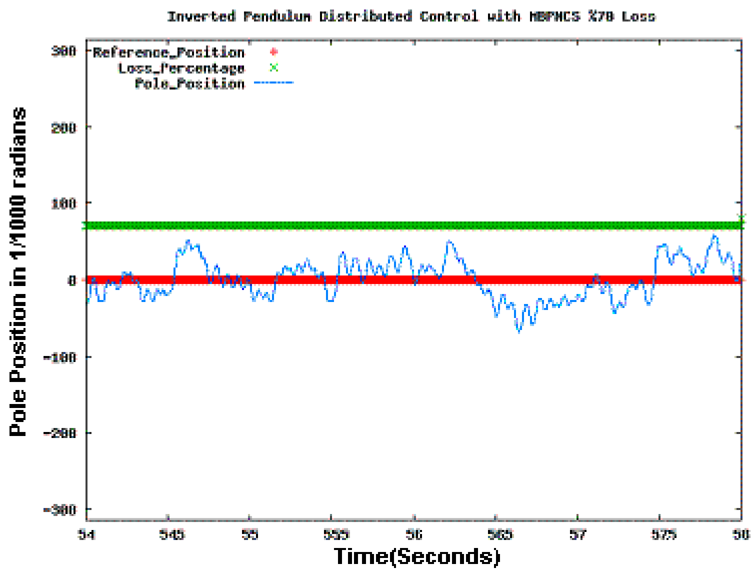

Figure 6.35 MBPNCS %70 Loss – Pole Angle

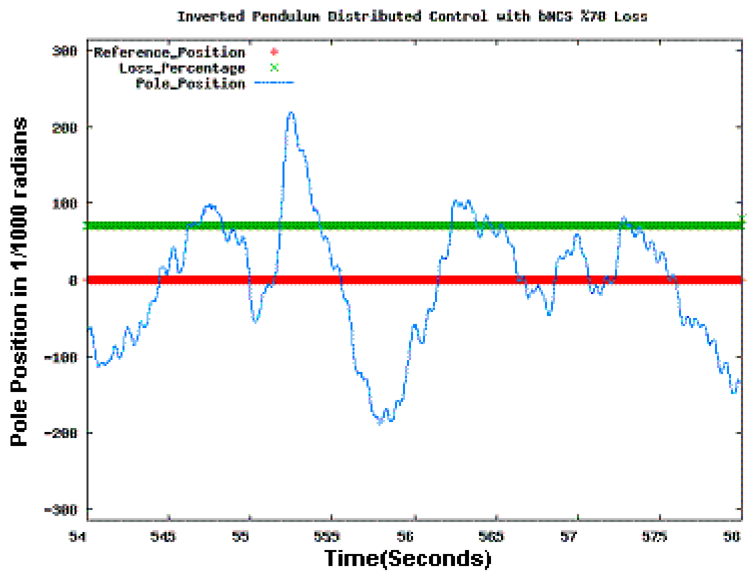

Figure 6.36 bNCS %70 Loss – Pole Angle

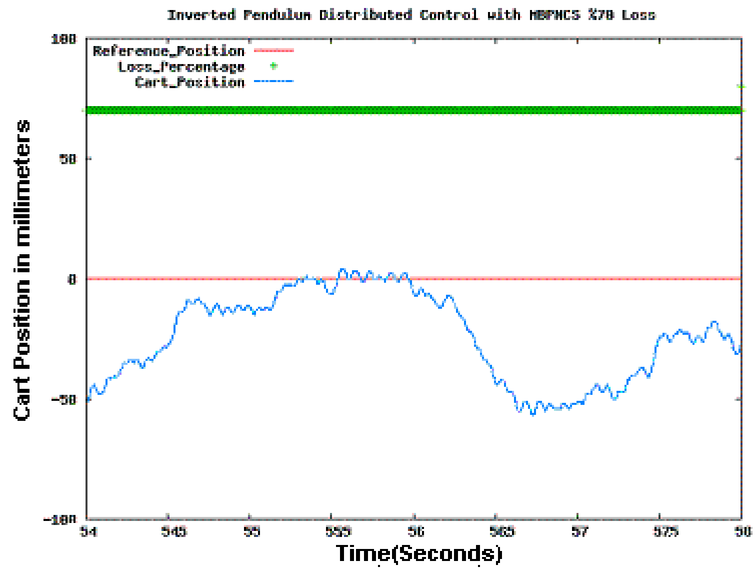

Figure 6.37 MBPNCS %70 Loss – Cart Position

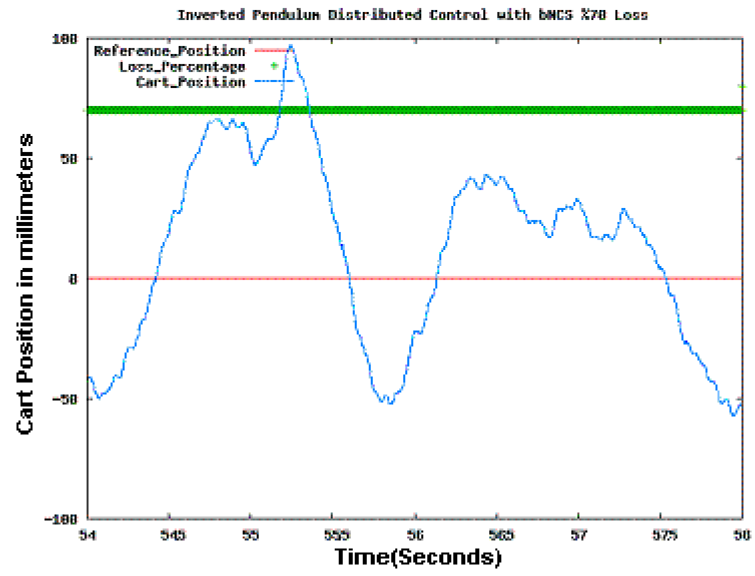

Figure 6.38 bNCS %70 Loss – Cart Position

The biggest performance difference between the MBPNCS and bNCS occurs in %70 packet loss. MBPNCS is able to hold the rod in an upright position where as the bNCS can not hold it in the allowed region.

Case 2 % 90 Loss Percentage

With % 90 loss percentage the performance of bNCS and MBPNCS are graphed in Figures 6.39 to 6.42.

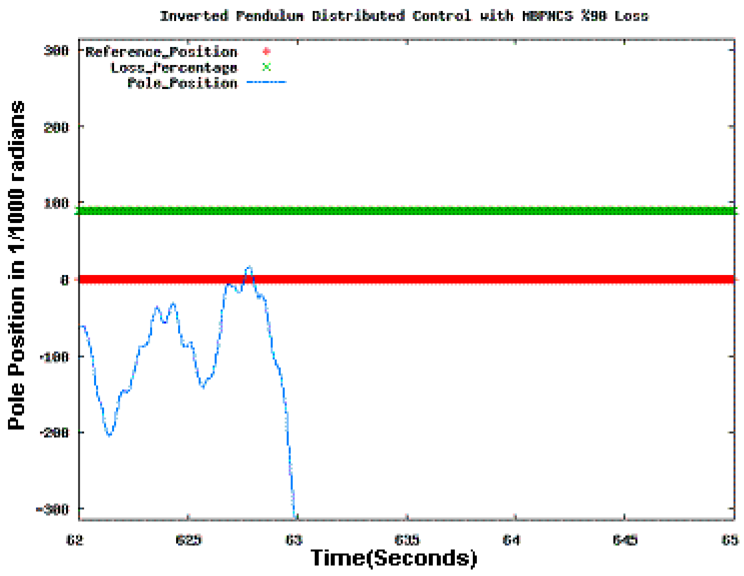

Figure 6.39 MBPNCS %90 Loss – Pole Angle

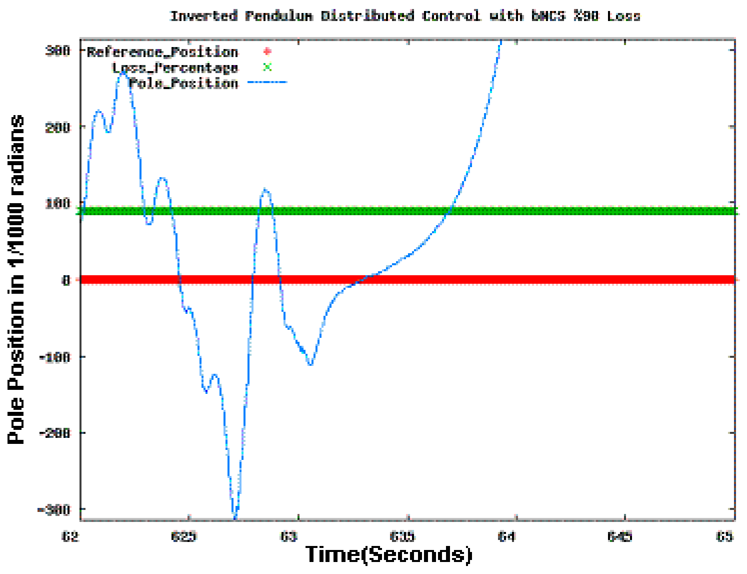

Figure 6.40 bNCS %90 Loss – Pole Angle

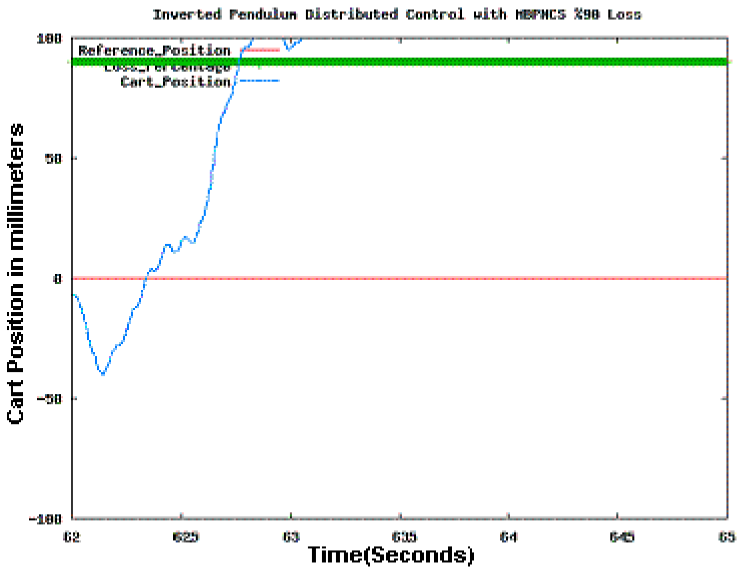

Figure 6.41 MBPNCS %90 Loss – Cart Position

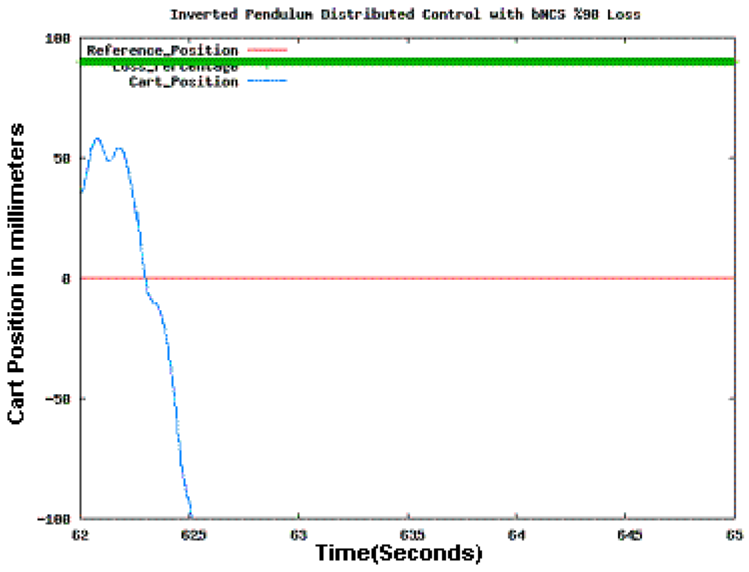

Figure 6.42 bNCS %90 Loss – Cart Position

With % 90 loss percentage both system fail to hold the pole angle and the cart position in the stable region.

Our experimental results verify our simulation results that concluded MBPNCS holds performance and stability with an open loop unstable plant up to % 90 packet losses.

Chapter 7

CONCLUSION and FUTURE WORK

In this thesis we experimented with a novel networked control system method; Model Based Control Networked Control Systems that aims in overcoming problems associated with data loss and random delay by implementing an intelligent predictive scheme. The intelligent predictive scheme predicts future plant states and control signals are calculated according to these states in the case of a data loss or delay.

Previous solutions to problems associated with Networked Control Systems had assumptions on the properties of the network or the control of the system and these assumptions would not be applicable in the real world industrial applications. However, MBPNCS is designed to work on wide range of network protocols and control algorithms with realistic assumptions, which makes it preferable over previous solutions.

In chapter 2 of this thesis, previous works on NCS solutions are summarized. In chapter 3, MBPNCS is introduced and explained in detail. Chapter 4 was on the implementation method and apparatus of the MBPNCS. Different plants and their models were also explained. Chapter 5 introduced a theoretical stability criterion for MBPNCS that builds a foundation for future work on MBPNCS. Finally, in chapter 6 results of experimental tests and simulations were produced. The aim of these tests were to conclude

a) that MBPNCS would be efficient in industrial applications by showing that MBPNCS holds performance when used with a control plant with an observer and when two separate MBPNCS s are working together. A control plant with an observer is implemented with a DC motor with a Luenberger Observer to achieve this aim.

b) that MBPNCS is more efficient and stable than a basic Networked Control System (bNCS) when used with an open loop unstable control plant. An open loop unstable plant is implemented with an inverted pendulum in these tests.

Through simulations and experimental results, this thesis has showed that MBPNCS has significantly better performance compared to an event based networked control system such as bNCS when used with an open loop unstable plant, tolerating communication losses up to 90%, whereas the latter may become unstable at 30%. Results have also showed that MBPNCS holds performance when working with an observer and that multiple MBPNCS s can be supported with a common communication medium. Based on these results, we believe that MBPNCS will be successful in industrial applications.

The theoretical stability criterion introduced in this thesis should be expanded and verified with experimental results for future work.

REFERENCES

[1] A.T. Naskali, A. Onat, "Model Based Predictive Networked Control Systems", *Ms. Thesis, Sabancı University*, 2006

[2] E. Parlakay, A. Onat, "Implementation of a Distributed Control System Using Real Time Operating System", *Ms. Thesis, Sabancı University*, 2007

[3] 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

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

[5] 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

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

[7] 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

[8] 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.

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

[10] 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

[11] K. Natori, K. Onishi “An approach to design of feedback systems with time delay”, *Industrial Electronics Society, 2005. IECON 2005. 32nd Annual Conference of IEEE 6-10 Page(s):6 pp*, Nov. 2005

[12] 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

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

[14] L. Montestruque and P. Antsaklis, “On the model-based control of networked systems,” *Automatica*, vol. 39, pp. 1837–1843, 2004.

[15] L. Montestruque and P. Antsaklis, “Stability of model-based networked control systems with time-varying transmission times,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1562–1572, 2003.

[16] G. Liu, Y. Xia, J. Chen, D. Rees, and W. Hu, "Networked predictive control of systems with random network delays in both forward and feedback channels," *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 1282–1297, 2007.

[17] A.Onat, "Control over Imperfect Networks: Model Based Predictive Networked Control Systems," *IEEE Transactions on Industrial Electronics*, to be published.

[18] Lam, J. "Control of an Inverted Pendulum". http://wwwccec.ece.ucsb.edu/people/smith/student projects/Johnny Lam report 238.pdf

[19] http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html

[20] D. Henriksson, A. Cervin, and K. Arzen, *"Truetime: Real-time control system simulation with matlab/simulink,"* in Proc. of the Nordic MATLAB Conference, 2003.

[21] D. Henriksson, A.Cervin, and K. Arzen, *"Simulation of control loops under shared computer resources,"* in Proc. 15th IFAC World Congress on Automatic Control, 2002.

[22] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, "*Numerical Recipes in C: The Art of Scientific Computing*," Cambridge University Press, 1993