Random Network Delay in Model Based Predictive Networked Control Systems

A. TEOMAN NASKALI
Sabanci University
Mechatronics Engineering Department
Orhanli Tuzla Istanbul
TURKEY

AHMET ONAT
Sabanci University
Mechatronics Engineering Department
Orhanli Tuzla Istanbul
TURKEY

Abstract: Networked control systems (NCS) provide many advantages for idustrial development. For larger scale systems and more complex systems usage of NCS's will become mandatory. However usage of networked control systems introduces 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, resulting in destabilization of the system and jeopardizing system stability. In this work a novel networked control system architecture that runs under non-ideal network conditions where packet loss and random time delays occur. previously introduced MBPNCS architecture is expanded by including random delay in the communication network. The delays and data losses caused by the communication network are compensated for using the computational power of the computer nodes of the networked control system. The architecture is independent of the control algorithm and uses a model to predict the plant states into the future to generate corresponding control outputs. This approach enables the system to be controlled in a pre-simulated manner and stability can be maintained even with high packet loss probabilities. In this approach, it has to be assured that the predicted control signals are applied while their prediction conditions are still valid. The proposed model based predictive networked control system architecture is simulated on a DC motor. The overall effect is that stability is maintained although the response of the plant to the reference can be delayed. The system remains stable even when there exists network jitter. The number of predictions that have to be made to keep the system running is also examined and a prediction horizon dependent on rise time of plant is proposed.

Key–Words: Model Based Control, Predictive Control, Networked Control Systems

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. Digital approaches increase the decision making capabilities of the controller and facilitate the design and reconfigurability of the system. They also enable the use of standardized components that reduce cost and increase the versatility in the implementation of digital control systems.

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 are also computer nodes on the network with some computational capability.

In a NCS, sensor nodes have the task of measuring one or multiple plant outputs and transmitting the measured values over the network. Controller nodes use the plant outputs that they receive from sensor nodes to calculate control outputs by a control algorithm which are then sent to the actuator nodes Actuator nodes have the task of applying commanded values received over the network to the plant by means of suitable actuators. The data that travels over the network is encapsulated in a packet.

The complexity of control design and the communication delays are drawbacks of NCS's. With the addition of a communication network in the feedback control loop, the complexity of analysis and design for a NCS increases because delay in the control loop has to be accounted for. There are essentially three kinds of delays in a NCS which are dependent on the network scheduling policy and are generally not constant or bounded in common network protocols: Communication delay between the sensor node and the controller node that has occurred during sampling instant $t_k : \tau_{sc}(t_k)$, computation delay in the controller node that has occurred during sampling instant $t_k : \tau_c(t_k)$, and communication delay between the controller node

and the actuator node that has occurred during sampling instant t_k : $\tau_{ca}(t_k)$. The length of the transfer delay depends on network load, priorities of the ongoing communications, electrical disturbances and various other factors. Note that sensor and actuator nodes also have some computational load and therefore 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 and the sensor node calculation delay can be included in $\tau_{sc}(t_k)$ and the computation delay at the actuator node can be included in $\tau_{ca}(t_k)$. The total delay from sensing to actuation is the sum of the above delays:

$$\tau(t_k) = \tau_{sc}(t_k) + \tau_c(t_k) + \tau_{ca}(t_k) \qquad (1)$$

2 Background

NCS's have been studied and several methods have previously been proposed to improve their stability [1, 2, 3]. Dead bands [4] aim to reduce the amount of communication [5] by eliminating repetitive transmissions of similar data, thus improving network conditions. However the network is assumed to be lossless. Gain adaptation [6] and network observers [7] proposed by Ohnishi observe the condition of the network and compensate for the effect of delay in the control algorithm by adjusting the gain or adding a negative feedback term however they consider the changes in network to be relatively slow or the network delay times 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 [8, 9] are used in similar scenarios but they 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 they are connected with a non-networked link. Also a-priori knowledge of the reference signal is assumed in the model predictive control. To address these problems, we propose a method that improves the performance of a basic NCS under variable time delays and packet loss. Standard NCS architecture is assumed and no direct links 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 possible with most systems.

3 Model Based Predictive Networked Control Systems

We propose a novel NCS architecture that will improve the robustness and stability of networked control systems to data loss. We 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 at once every sampling instant 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 sampling instant, hence the name Model Based Predictive Networked Control System (MBPNCS).

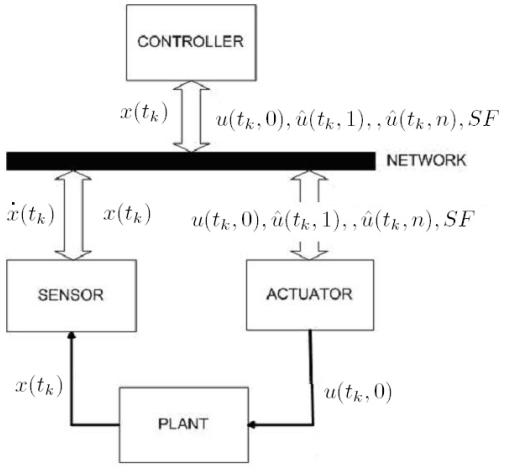

Figure 1: Model Based Predictive Networked Control System Setup

The proposed control system is composed of five parts: a *communication network*, a *sensor node* which has the primary function of sensing and sending plant states $x(t_k)$ to the controller, a *controller node* which generates and sends the control packet containing the control outputs $u(t_k, i)$ where t_k is the moment of sampling at the sensor and i is the predicted control signal that is expected to be applied 'i' sampling time steps into the future from t_k, a *model of the controlled plant* $\hat{P}$ which is reinitialized at the controller node whenever new plant states $x(t_k)$ arrive from the sensor node to limit deviation of the sates of the model from those of the actual plant and an *actuator* node which applies the control output $u(t_k)$ or a predicted control output $\hat{u}(t_k)$ generated by the controller node to the plant every sampling instant. All nodes run periodic tasks and late packets are considered to be lost. Packet loss between the sensor node and the

controller node is compensated at the controller node by prediction and packet loss between the controller node and the actuator node is compensated at the actuator node by usage of predicted control outputs and a selection algorithm which determines the appropriate control output or prediction.

The sensor node senses the plant states $x(t_k)$, then rate terms of the control algorithm are calculated at the sensor node since continuity of plant states cannot be assured at the controller node because of network packet loss. Then the plant states and the rate terms are encapsulated in a network packet and sent to the controller node. The controller node obtains the plant states $x(t_k)$ and the proportional and derivative terms from the sensor node and reinitializes the plant model with them. If packet loss has occurred then the controller node uses $\hat{P}$ and $x(t_{k-1})$ to generate $\hat{x}_{t_k-1}(t_k)$ instead, which is the predicted plant states for time t_k that were predicted with sensor data originating from time t_{k-1}, also this data loss information is stored in a boolean variable SF which is set to high if data form sensor is received and low otherwise. In these predictions it is assumed that the controller to actuator link has not been broken in the sampling period. Since the control output depends either on actual plant states or predicted ones, it is important to convey this information to the actuator; if the actuator has successfully applied the previous control outputs to the plant but the controller was not able to receive the current plant states from the sensor, its prediction $\hat{x}(t_k)$ will have small error, whereas if the controller to actuator link had broken during transmission of output $u(t_{k-1})$, then the actuator will have applied a different input to the plant, and if the current plant states $\hat{x}(t_k)$ could not be received and needed to be predicted, this prediction would not be correct. How this of synchronizationis handled will be explained in more detail at the end of this section. A control algorithm is applied to $x(t_k)$ or $\hat{x}_{t_{k-1}(t_k)}$ and a control output $u(t_k, 0)$ is generated. $u(t_k, 0)$ is applied to $\hat{P}$ with $x(t_k)$ resulting in the predicted states for time t_{k+1} : $\hat{x}_{t_k}(t_{k+1})$, then the control algorithm is used again to calculate a predicted control output $\hat{u}(t_k, 1)$. This procedure is repeated n times until $\hat{u}(t_k, n)$ is obtained. All of these control outputs and SF are inserted into a packet and sent over the network to the actuator node. The packet $Pt(t_k)$ generated at time tk therefore consists of n+1 control outputs and a sensor flag: $u(t_k, 0), \hat{u}(t_k, 1), \ldots, \hat{u}(t_k, n), SF$. Figure 1 depicts the contents of network packets and their relations in a NCS. The number of predictions is chosen based on the following factors:

1. The accuracy of the model should be considered.
2. The packet size should not decrease network quality of service.
3. Processing power available.

A method of identifying the prediction horizon for open-loop stable systems is given in section 4.3. The actuator node applies the signals obtained from the controller node to the plant. If no contol output has arrived from the controller node for sampling interval $[t_k, t_{k+1}]$, then a prediction from a previously received packet such as $\hat{u}(t_{k-1}, 1), \hat{u}(t_{k-2}, 2) \ldots \hat{u}(t_{k-n}, n)$ is applied. If no prediction is available for cases where transmissions have been broken for a long time, final value is held. If a control packet has actually been received successfully there are still two possibilities:

1. Either the contol singal just recived is based on successfully received state information or a correct state prediction

2. It is based on an erroneous state prediction. This depends on the synchronization between the plant and the model of plant. As explained above the *SF* in the packet is used to convey sensor node to controller node packet loss information. Therefore the actuator node has two states, the synchronized mode and the interrupted mode.

Synchronized mode: The synchornized mode indicates that the plant model states of the controller have small error compared to the states of the plant. If the actuator node receives a control packet from the controller node when it is in the synchronized mode then it applies the control output from that packet to the plant at time t_k, which would be $u(t_k, 0)$ if sensor flag is high, or $\hat{u}(t_k, 0)$ if sensor flag is low. A low sensor flag is ignored in the synchronized mode. However, if packet loss has occurred adn actuator node does not receive a control packet then $\hat{u}(t_{k-1}, 1)$ is applied to the plant and the state of the actuator node changes to interrupted mode. The actuator switches back to synchronized mode only when a packet with a high SF arrives from then on.

Interrupted mode:

When the actuator node fails to receive a packet the controller node is not aware of this event. The controller node assumes $u(t_k, 0)$ was applied to the plant and does its calculations accordingly, whereas actually $\hat{u}(t_{k-1}, 1)$ has been applied. This causes the synchronization between states of plant and its model to be lost. If a control packet with a low SF arrives after the actuator has entered the interrupted mode signifying this situation, then the packet is rejected. The control outputs in the packet received before the actuator entered the interrupted mode are used one after the other. The reason for rejecting packets is that the controller node generates control signals assuming that the controller to actuator link is not broken due to

packet loss. But the previous packet was definitely lost and therefore the control outputs in the current packet with low SF may not be correct.

4 Results

The proposed method was tested using computer simulations using TrueTime [10]. TrueTime is a Matlab toolbox which is designed to simulate real-time computer networks [11]. It has a low level of abstraction where it simulates computer systems at instruction execution level and communication network at data transport level. Therefore, we can say our results are close to actual implementation. The DC-servo described by the following transfer function is used as the system plant.

$$G(s) = \frac{1000}{s(s+1)} \tag{2}$$

A PD controller is implemented according to the following equations;

$$KP(t_k) = K(r(t_k) - y(t_k)) \tag{3}$$

$$KD(t_k) = \alpha_d KD(t_{k-1}) + \beta_d(y(t_{k-1}) - y(t_k)) \tag{4}$$

$$\alpha_d = \frac{T_d}{N_h + T_d} \tag{5}$$

$$\beta_d = \frac{NKT_d}{N_h + T_d} \tag{6}$$

$$u(t_k) = KP(t_k) + KD(t_k) \tag{7}$$

where $r(t_k)$, $y(t_k)$, $u(t_k)$ are reference, plant output, control output and $KP(t_k)$, $KD(t_k)$ are proportional and derivative components of control output, K is the proportional gain and t_k, is the sampling instant. N, N_h, α_d and β_d are constants. The value $y(t_k)$ is obtained by $Hx(t_k)$ where H is the output matrix of the plant and $x(t_k)$ are the plant states at time t_k. The sensor node also calculates the $KD(t_k)$ term using

$$KD(t_k) = \alpha_d KD(t_{k-1}) + \beta_d(y(t_{k-1}) - y(t_k)) \tag{8}$$

and also stores old values of $KD(t_{k-1})$ and $y(t_{k-1})$. Therefore $KD(t_k)$ used initially originates from the sensor node, estimates of $KD()$ are used for future predictions or if no data arrives from the sensor node. Therefore, if actual $KD(t_k)$ values from the sensor arrive in time for calculations, they are used in the controller node, but if they do not arrive on-time, their estimates are used instead. The controller node also calculates n other control outputs, only the first one is calculated with $KD(t_k)$ originating from the sensor. The other calculations are done with calculated $KD()$ values. A broadcast network topology using CSMA/CD scheme is employed.

The proposed control system is compared with a basic Networked Control System (bNCS) where only the sensor node runs a periodic task and the controller and actuator nodes run event driven tasks that function only when they receive a message from the network to calculate control output and apply it to the plant respectively. As performance metric, the root mean square of the error between the reference and plant output is used.

4.1 Model Based Predictive NCS (MBPNCS) Controller Setup

The state space representation of continuous plant model is discretized with a sampling time of 0.01 seconds. The controller node starts processing 0.001 seconds after the sensor node and the actuator node starts processing 0.001 seconds after the controller node. This is to ensure that the network has time to deliver the data packets between the nodes. Under ideal network condition of no packet loss both the MBPNCS and bNCS display identical results, Figures 2 and 3.

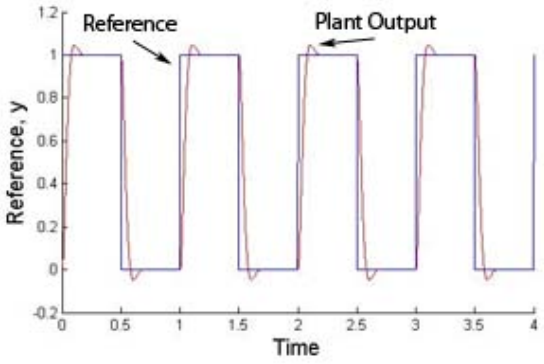

Figure 2: Basic NCS, ideal conditions
RMS Error: 0.2324

As packet loss increases degradation in performance is observable with the increasing RMS errors. However the reason for the degradation in control quality is different in both systems. The cause of increased RMS in the bNCS is loss of stability because packets are late causing loop delay and resulting in performance shown in Figure 5, on the other hand the increase in RMS of the MBPNCS is because even if packets are late, a calculated control output is applied to the plant. However after the calculation of this control output the reference may have changed. Therefore the plant is controlled towards an old reference. The retardation of the reference can be seen clearly on Figure 7 where the reference and plant output are shown together with the control output signal estimate

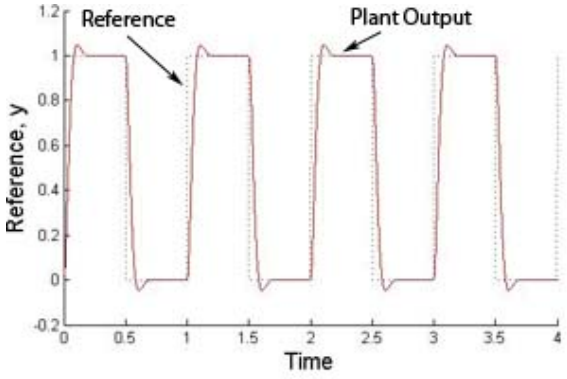

Figure 3: MBPNCS, ideal conditions
RMS Error: 0.23252

i used from the control packet. Signal estimate is offset by -4 for clarity. It can be seen that the plant is able to catch the reference once the actuator reenters a synchronized mode and the actuator node does not have to remain in synchronized mode to be able to go to the reference. Note that our system does not have a-priori knowledge of the reference signal in contrast to other research such as [9]. Performance of bNCS deteriorates much faster than a MBPNCS. Stability is also lost on a bNCS whereas it remains stable on a MBPNCS, even for extreme rates of packet loss. The degradation in performance is shown in Figures 4 and 5 [12].

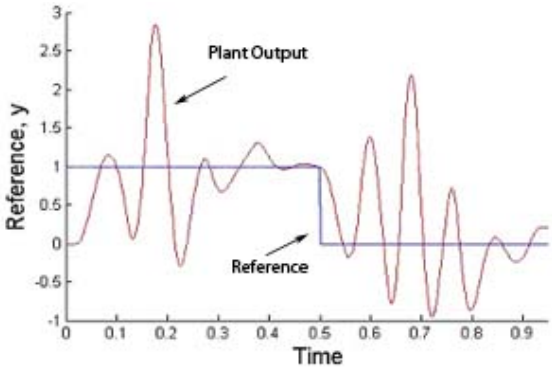

Figure 4: Basic NCS 20% packet Loss
RMS Error: 0.66509

The tolerance to noise is examined in [12].

4.2 Effect of Random Network Delay

Jitter is the uncertaity in the actual starting time of a scheduled event. It is always present in actual computer systems. In a networked control systems, there are two kinds of jitter: Due to uncertain execution times of tasks within the computer nodes and due to the uncertain time delays caused by the communication network. The former can be eliminated by taking the worst case execution times of tasks rather than the actual, and designing the computer nodes for those circumstances. The latter cannot be easily removed unless a real-time network of suitable conditions is used. However, in this work, we take a broadcast network topology using CSMA/CD contention management scheme. Other sources of jitter in the system are considered insignificant. It is possible to keep phasing of execution of each computer constant within the sampling period using clock synchronization algorithms and guarantee that evey node in the NCS wakes-up at the correct time with respect to others maximizing the possibility of correctly receiving data from other nodes in the network. By phasing the wake-up time of the controller node to the sensor node and of the actuator node to the controller node, we can offset the majority of problems caused by network protocol delay. However because of the higher granularity of clock synchronizaton algorithms and unbounded nature of network protocol delays, it is necessary to examine the system under certain such delay conditions. The aim is to examine if the system

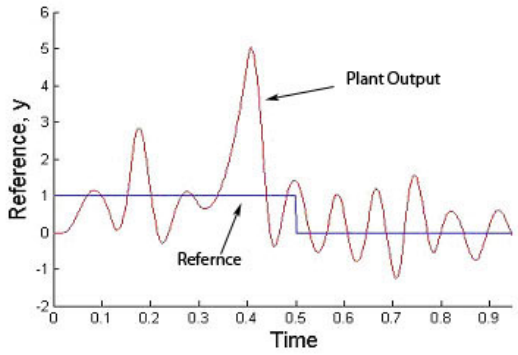

Figure 5: Basic NCS 30% packet Loss
RMS Error: 1.023

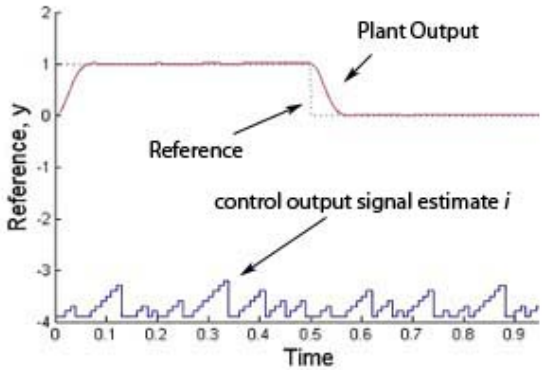

Figure 6: MBPNCS 50% packet Loss
RMS Error: 0.22644

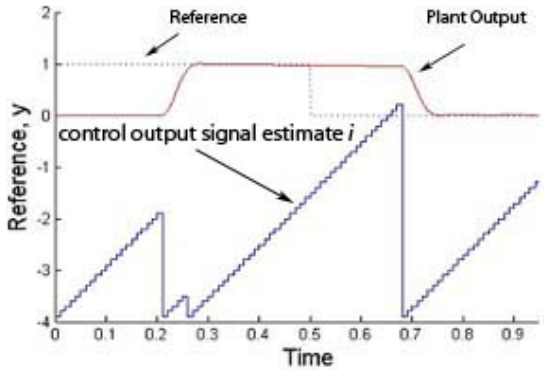

Figure 7: MBPNCS 90% packet Loss
RMS Error: 0.65153

can tolerate jitter while a clock synchchronization algorithm synchronizes the clocks.

In the simulations it is assumed that the sampling period at each computer node is fixed and no clock drift occurs. On startup the nodes synchronize and start at once and their periods stay the same. In actual implementations, computer nodes will have clock drift. However, clock synchronization algorithms will either adjust the drift to keep the nodes synchronized, or reset their clocks if drift causes them to lose data packets. This means, we can treat closck drift as part of the uncertain network protocol delay.

Random network delay is introduced in to the system as follows:

1. The sensor node delays the sensor data before sending it to the controller node τ_{sc}.

2. The actuator node has processing delay before the data is copied to the signal buffer τ_{ca}.

Since the controller does not apply a time stamp to packets, its calculation delay can be incorporated into either of the delays. These delays are expressed as a percentage of the sampling period. A delay of 10% in a system where the samling period of 0.01 seconds would be 0.001 seconds.

The wake-up time of the three nodes in the system are offset by 10% of the sampling time respectively. Therefore if there is a delay of more than 10% the packet will be late and it will be discarded. This is the most significant effect that delays have on MBPNCS. Beside this, excessive delays will alter the loop time and therefore cause instability.

The relation of the jitter to the packet loss is calculated with the following formula:

$$TsT_{off} = (1 - P_l)TsJ \tag{9}$$

where: Ts is sampling time, J is Jitter, T_{off} is offset P_l and is packet loss probability.

In figure 9 performance degradation due to jitter can be compared with that due to packet loss which is depicted in figure 8.

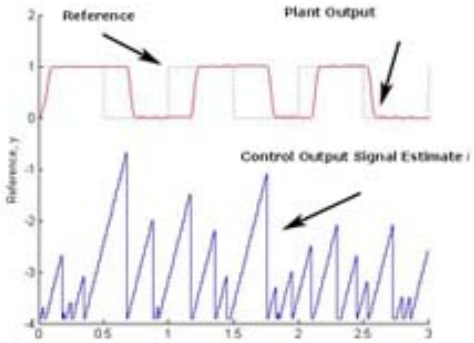

Figure 8: MBPNCS 70% packet loss only

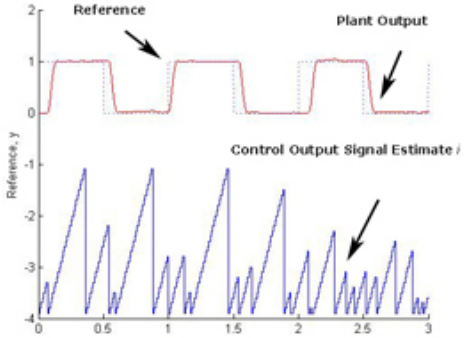

Figure 9: MBPNCS 70% Random network delay causing average packet loss equivalent to 70%

When compared to the basic NCS, the performance is comparable under normal rates of random network delay remaining within one sampling period. This is because the packets in a basic NCS are not dropped when they are late. This enables the control loop to be closed and the effect of the jittery delay can be considered as noise. For larger values the bNCS loses stability. The result of a bNCS applied subjec to network delay can be seen on Figure 13.

In figures 10 - 12 increasing amounts of random network delay are applied to the system. In all cases, the result is similar to that of pure network packet loss as expected. Long periods of communication may cause the reference singnal to be applied late to the plant. This is unavoidable since the reference is not known a-priori.

4.3 Prediction Horizon Calculation

To determine the minimum amount of predictions necessary to control the plant without any degradation

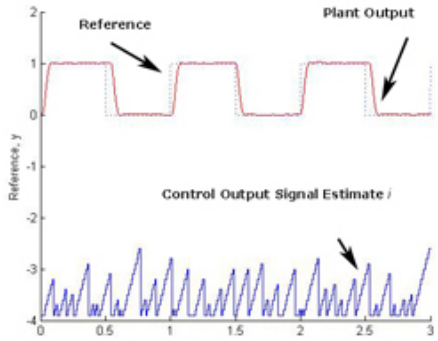

Figure 10: MBPNCS 30% Loss due to random network delay, 40% packet loss

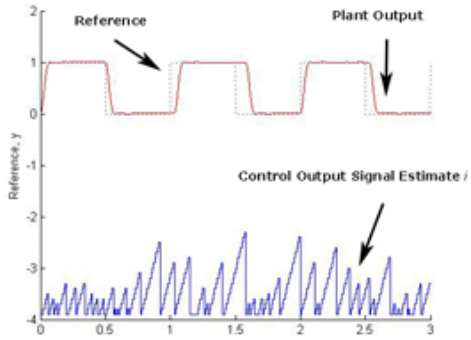

Figure 11: MBPNCS 35% Loss due to random network delay, 40% packet loss

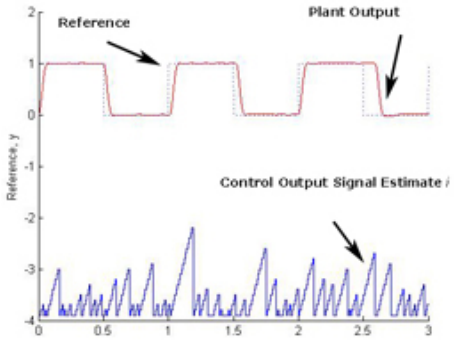

Figure 12: MBPNCS 50% Loss due to random network delay, 20% packet loss

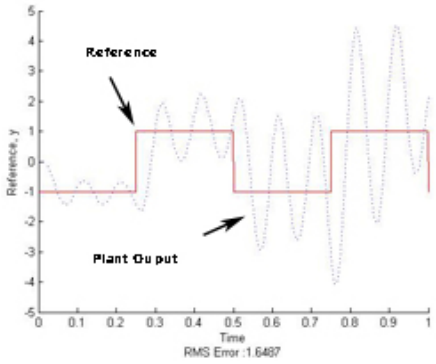

Figure 13: Basic NCS subject to network delay 80% of period

in performance, the control signal sent to the plant is examined. Figure 14 shows the reference, plant output and control signal. The amount of predictions necessary can be related to the settling time of the plant. The figure shows that the control signal rests at 0 after the 0.20th second. 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. Therefore, predictions need only be performed within a time span that corresponds to the rise time of the plant. However this will be applicable only for control applications involving an open loop stable plant.

5 Conclusion

In this work, a novel networked control system method; model based predictive network control system (MBPNCS) is presented. It takes advantage of the fast processing power of modern 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 plant state synchronization with the plant model inside the controller node. Controller states are partially processed at the sensor node to ensure synchronization is regained during recovery from packet loss. This new NCS method relies on a plant model to predict the effect of control outputs to be applied to the plant, and generates an array of control outputs that span over a specified prediction horizon into the future. 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. The control output is then applied to the plant by the actuator node. Should communication between the controller node and actuator node fail, the actuator node uses the predicted control outputs 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

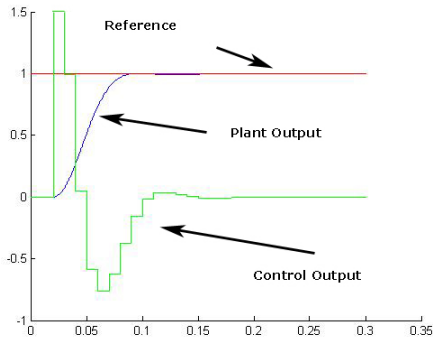

Figure 14: MBPNCS Control Signal

the MBPNCS have been examined. It has been observed that the architecture is robust against network 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 random network delay on the system is examined. Since late packets are considered lost, random network delays only introduce high rate of packet rejection which has an effect identical to packet loss. It is observed that the MBPNCS architecture remains resilient with random network delay. If the controlled plant is open loop stable, it is possible to limit the number of estimates to within the reise time of the plant. Further development of this method will include examination of performance under imperfect plant model and experimental implementation.

References:

[1] M.S. Branicky, S.M. Phillips, and Wei Zhang. Scheduling and feedback co-design for networked control systems. *IEEE Conference On Decision and Control*, 2(41):1211– 1217, December 2002.

[2] M. S. Branicky, V. Liberatore, and S. M. Phillips. Networked control system co-simulation for co-design. *American Control Conference*, 4:3341–3346, June 2003.

[3] Q Ling and M. Lemmon. Robust performance of soft real-time networked control systems with data dropouts. *IEEE Conference On Decision and Control*, December 2002.

[4] P Otanez, J. Moyne, and D. Tilbury. Using deadbands to reduce communication in networked control systems. *American Control Conference*, 2000.

[5] J. Yook, D. Tilbury, and N. Soparkar. Performance evaluation of distributed control systems with reduced communications. *IEEE Control Systems Magazine*, 21(1):84–99, 2001.

[6] C. Mo-Yuen and Y. Tipsuwan. Gain adaptation of networked dc motor controllers based on qos variations. *IEEE Transactions on Industrial Electronics*, 50(5):936– 943, October 2003.

[7] K. Natori and K. Onishi. Time delay compensation by communication disturbance observer in bilateral teleoperation systems. *Advanced Motion Control 2006*, pages 218 – 223, March 2006.

[8] J.B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3):38–52, June 2000.

[9] G. P. Liu, J. X. Mu, and D. Rees D. Networked predictive control of systems with random communication delay. *UKACC International Conference on Control*, September 2004.

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

[11] D. Henriksson, A.Cervin, and K. Arzen. Simulation of control loops under shared computer resources. *Proceedings of the 15th IFAC World Congress on Automatic Control*, July 2002.

[12] A. Teoman Naskali and Ahmet Onat. Model based predictive networked control systems. *Submitted to IPDPS*, 2007.

[13] J. K. Yook, D. M. Tilbury, 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 2000.

[14] J. Yook, D. Tilbury, K. Chervela, and N. Soparkar. Decentralized, modular real-time control for machining applications. *American Control Conference*, pages 844–849, 1998.