Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Model Based Predictive Networked Control Systems

Ahmet Onat, A Teoman Naskali, [*] **Emrah Parlakay** [**]

[*] *Sabanci University, Faculty of Engineering and Natural Sciences, Istanbul 34956, Turkey; e-mail: onat@sabanciuniv.edu, naskali@su.sabanciuniv.edu*
[**] *Alcatel-Lucent, 2280 GG Rijswijk-Zh, The Netherlands, e-mail:emrah.parlakay@alcatel-lucent.com.tr*

Abstract:
Networked control systems where the sensors, controller and actuators of a digital control system reside on different computer nodes linked by a network, aim to overcome the disadvantages of conventional digital control systems at the application level, such as difficulty of modification, vulnerability to electrical noise, difficulty in maintenance and upgrades. However random communication delay and loss on the network may jeopardize stability since the communication delay decreases the phase margin of the control system and data loss can be considered as noise. In this project, we propose a novel networked control method where satisfactory control is possible even under random delay and data loss. We keep a model of the plant inside the controller node and use it to predict the plant states into the future to generate corresponding control outputs. At every sampling period the states of the model are reset to the actual or predicted states of the plant. The ambiguity of plant state during periods of total communication loss are also addressed.
The proposed model based predictive networked control system architecture is first verified by simulation on the model of a DC motor under conditions of data loss and noise. Then experiments are repeated on a dedicated test platform using a physical DC motor. Results show that significant amounts of data loss and delay can be tolerated in the system before performance starts to degrade.

1. INTRODUCTION

A Networked Control System (NCS) is a feedback control system where the control loop is closed over a communication network consisting of actuators, sensors and controllers, each of which are computer nodes on the network. Actuators and sensors generally also have some computational capability. This distributed structure is advantageous because of its inherent flexibility, reconfigurability and reduced vulnerability to noise and calibration errors.

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

The complexity of 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. 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:

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

The computational delay τ_c is variable, not only because of the time the control algorithm takes, but because of the scheduling algorithm used [Toerngren, 1998].

In case data is lost over the network, rather than resending to compensate for the lost data, it is more beneficial to send freshly acquired or calculated data, be it plant information, or control output. In this work, we utilize

[⋆] This work was supported in part by the Scientific and Technological Research Council of Turkey, project code 106E155.

this idea, and propose a networked control system where reception acknowledgment of network data is not required as will be explained in Section 3.

2. BACKGROUND

The separation of concerns between the control and computer communities is one of the reasons for the difficulty in implementing control methods for complex plants. As a solution, NCS's have been studied and several methods have previously been proposed to improve stability of NCS [Branicky et al., 2002, 2003, Ling and Lemmon, 2002, Toerngren, 1998]. Dead bands proposed by Otanez et al. [2000] aim to reduce the amount of communication [Yook et al., 2001] by eliminating repetitive transmissions of similar data, thus improving network conditions. However the network is assumed to be lossless. Gain adaptation by Mo-Yuen and Tipsuwan [2003] and network observers by Natori and Onishi [2006] 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.

General predictive control methodology is relevant to NCS's [Clarke et al., 1987a,b, Clarke and Mohtadi, 1989] and Model predictive controllers are used in similar scenarios as in Rawlings [2000], Liu et al. [2004] 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 do not communicate at all over a network. The reason is that if the sensor node to controller node transmission fails then the basis for predictions is lost. 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 the current and predicted control output to the plant for several time steps into the future at every sampling instant. All of these outputs are then sent to the actuator node at once. If there was no data loss in the controller to actuator link, the actuator applies the first control output to the plant. In case of data loss, a previously sent prediction is applied to the plant at each successive sampling instant, hence the name Model Based Predictive Networked Control System (MBPNCS).

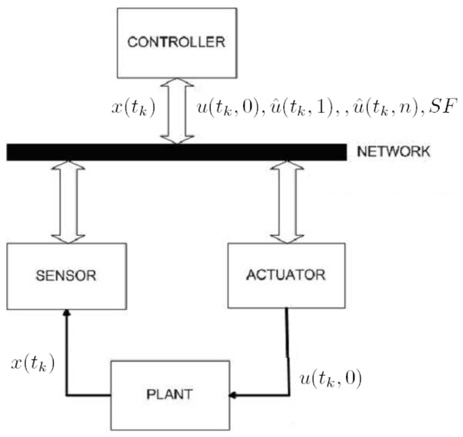

Fig. 1. Model Based Predictive Networked Control System Setup

The proposed control system is composed of five parts: a communication network, where we assume that packet loss and delay occurs completely randomly (despite the fact that noise in networks is generally more correlated, a complete random behavior was preferred for simulations for simplicity), one sensor node, one controller node and one actuator node, and finally a model of the plant $\hat{P}$ residing inside the controller node.

The sensor node samples the plant states $x(t_k)$ with period h and sends them to the controller at every sampling instant t_k. The rate terms of the control algorithm pertaining to the plant states are calculated at the sensor node since continuity of plant states cannot be assured at the controller node because of interruption of communication. The plant states and the rate terms are encapsulated in a network packet and sent to the controller node.

The controller node not only calculates the control output for the current plant state $u(t_k, 0)$ (notation will be clarified shortly) using a control algorithm, but also uses the plant states just received $x(t_k)$, to initialize the internal model and through an iterative process, generates a series of future control outputs, $\hat{u}(t_k, i)$ and state estimates $\hat{x}(t_{k+i}|t_k)$ where $i; i = 1, 2, ..n$ is the index of control output signal estimate that is expected to be applied at time $t_k + ih$ in the future if communications were to fail. If $x(t_k)$ was not available at the time of computation because of data loss or communication delay, its estimate $\hat{x}(t_k|t_{k-1})$ obtained using the model $\hat{P}$ from $x(t_{k-1})$ or if that is not available, $\hat{x}(t_{k-1}|.)$ from previous estimates. For this estimation to be valid, it must be assumed that the previous control output was transmitted properly and applied to the plant. How this restriction can be relaxed will be addressed shortly.

The fact that a state estimate has been used is important along the control decision line. Therefore this information is stored in a *sensor flag* (SF), which is set to high if current control output is based on actual plant sates from the sensor node, and low if state estimates were used. The control outputs are collected in a control packet $Pt(t_k)$

consisting of $n+1$ control outputs and a sensor flag: $[u(t_k,0), \hat{u}(t_k,1), \cdots, \hat{u}(t_k,n), SF]$. Figure 1 depicts the contents of data packets and their relations in MBPNCS.

The number of predictions n is chosen based on factors such as the accuracy of the plant model $\hat{P}$, the packet size compared to the network bandwidth, and available processing power.

Finally the actuator node receives packet $Pt(t_k)$ and applies the control output $u(t_k)$ to the plant at every sampling instant. If a new packet does not arrive on time because of data loss or communication delay, a predicted control output $\hat{u}(t_{k-i}, i)$ received previously is applied. This could cause the above mentioned problem of predicted plant states deviating from the actual plant states since if a sensor to controller communication loss also occurred simultaneously, the controller assumes that $u(t_k)$ was last applied to the plant when calculating its state predictions. In our method, this is called the loss of synchronization (slightly abusing the term), and our method is designed so that the actuator node is responsible for coping with the situation.

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.
Synchronized mode: The synchronized mode indicates that the states of the plant model are synchronized with the plant states. If 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 would be $u(t_k,0)$. If a transition of SF from high to low occurs in consecutive control packets indicating 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 $u(t_{k+j},0)$, since this does not violate the assumption made by the controller that these control outputs 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.
Interrupted mode: In this mode, the actuator applies $\hat{u}(t_k,i)$ of the last control packet received in synchronized mode at every sampling instant t_{k+i}, $i=1,2,\cdots,n$ until the last sample is reached or communication is restored. However, if the first control packet received in this mode has a low SF indicating that the controller is using state estimates based on the wrong assumption of applied control signal, then the packet is rejected. If the last prediction $\hat{u}(t_k,n)$ is reached without the communication being restored, the output is kept constant at that value thereafter. The actuator node enters synchronized mode when a control packet with a high SF is received.

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 prediction and packet loss between the controller node and the actuator node is compensated at the actuator node by usage of a selection algorithm and predicted control outputs. Late arriving packets are discarded in this work. A time synchronizing method is assumed to be used among the computer nodes. This is not a strong assumption because the network is generally physically small and the amount of synchronization accuracy is comparable to the sampling time.

4. RESULTS

The proposed method was tested using computer simulations using TrueTime by Henriksson et al. [2003] and experiments with actual computers and communication network.

4.1 Simulations

TrueTime is a Matlab toolbox developed by Henriksson et al. [2002] is designed to simulate real-time computer networks at 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. A DC servo motor described by the following transfer function is used as the system plant [Astrom and Wittenmark, 1997].

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

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 $H * x(t_k)$ where H is the output matrix of the plant and $x(t_k)$ are the plant states at time t_k. 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.

The sampling time of the system is $0.001s$, and there is a phase delay of $0.0001s$ between sensor, controller and actuator node periods to ensure that the network has time to deliver the data packets between the nodes. Such a phasing was not used for the experiments. Simulations are made with a perfect model of the plant to prove that the concept is functional. Further simulations examining imperfect models will be performed in the future.

Under ideal network condition of no packet loss both the MBPNCS and bNCS display identical results, Figures 2 and 3.

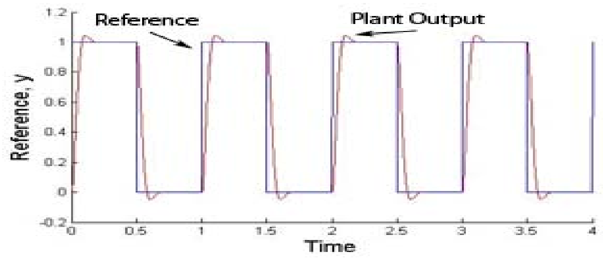

Fig. 2. Basic NCS, ideal conditions RMS Error: 0.2324

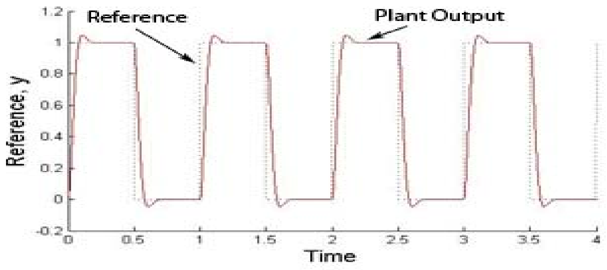

Fig. 3. MBPNCS, ideal conditions RMS Error: 0.23252

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 error in the bNCS is loss of stability because of increasing loop delay. The bNCS system becomes unstable after around 20% of packets lost or delayed(Figure 4). 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 6 where the reference and plant output are shown together with the control output signal estimate i(offset by -4 for clarity) used from the control packet. 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 in Liu et al. [2004]. MBPNCS remains stable, even for extreme rates of packet loss such as 90% (Fig. 6).

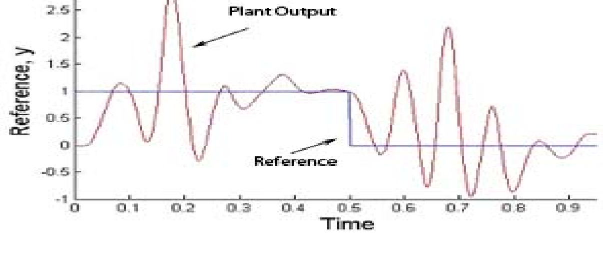

Fig. 4. Basic NCS %20 Packet Loss RMS Error: 0.66509

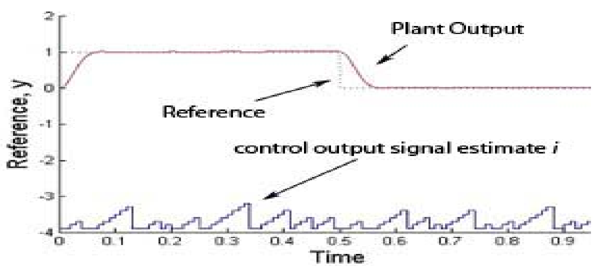

Fig. 5. MBPNCS %50 Packet Loss RMS Error: 0.22644

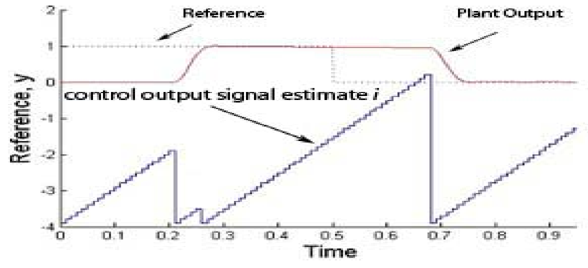

Fig. 6. MBPNCS %90 Packet Loss RMS Error: 0.65153

To observe the performance of the method under noise, band limited additive white noise is added to the control output that the actuator node applies to the plant. This has another destabilizing effect on the system: Since the actual control signal applied to the plant is different from the computed one, plant and model state synchronization will be jeopardized. The power of the noise to be added to the signal is related to the dynamic range of the signal. To have a relation between the noise power and the signal, noise power will be expressed as a fraction of the RMS value of the control output u generated under ideal network conditions given by:

$$u_n(t_k) = u(t_k) + n(t_k) \tag{8}$$

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

The performance of the bNCS is also almost identical under lossless network conditions. However, error due to packet loss almost immediately dominates error due to noise and the system becomes unstable.

The effect of low noise is not destabilizing up to %60 packet loss. The performance in Figure 7 where the MBPNCS is running in a network where there is no packet loss and Figure 8 where there is %70 packet loss may be compared. After %70 packet loss the effect of noise becomes significant in control quality. Effects of noise are reduced through the feedback loop. High rate of packet losses mean that the feedback loop cannot be closed until controller node to actuator node synchronization is regained as explained. Therefore losing the advantage of the closed loop system makes the system vulnerable to noise and control quality

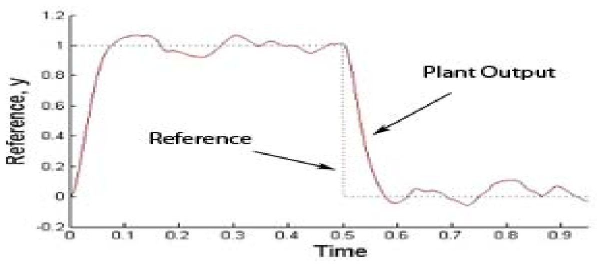

Fig. 7. MBPNCS subject to 0.001 Noise,%0 Packet Loss RMS Error: 0.22726

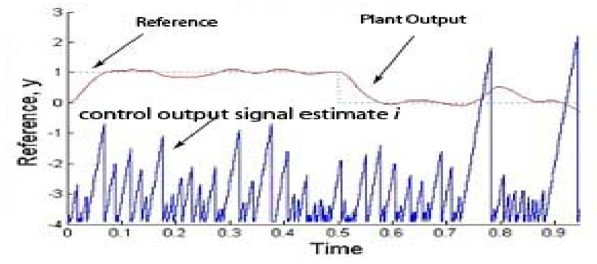

Fig. 8. MBPNCS subject to 0.001 Noise,%70 Packet Loss RMS Error: 0.27542

diminishes, as packet loss increases. This is illustrated in the figure between 0.7 and $0.8s$ where there is a long network blackout of $0.06s$. This means that the actuator node has not received consecutive 60 acceptable packets from the controller, yet the system remains stable despite noise. Therefore we can say that the proposed method increases resilience to data loss under noise, compared to bNCS which is not shown because it cannot tolerate such high packet losses.

The effect of jitter is analyzed on Naskali and Onat [2006].

4.2 Experiments

A physical setup comprising of three industrial computers running Real-Time Linux operating system for the computer nodes, a dedicated Ethernet network with a non-switching hub to allow packet collisions, and a DC motor with encoder and drive electronics was prepared for experiments. The computers were using NS Geode microprocessor running at 300MHz, and a maximum jitter for periodic task timing of around $100\mu s$ was measured. The sampling time was set at $1ms$. Again speed control of DC motor using PID method was targeted. Motor parameters were measured and used as plant model.

Since Ethernet is stochastic, controlled amounts of delay and data loss on the network were implemented by randomly accepting or rejecting incoming data packets on the computer nodes at a specified rate.

Tests performed on the experimental platform are similar to those for the simulations. The first test was for a practically no data loss or delay scenario of 0.33% for MBPNCS, to see performance under ideal conditions. The small amount of data loss is caused by the Ethernet hub. The results are given in Figure 9. It can be seen that the system can control the speed with little error. The noise at the $50rad/s$ reference level is also present if a completely centralized conventional controller is used and is believed to happen because of sensor noise which is not filtered.

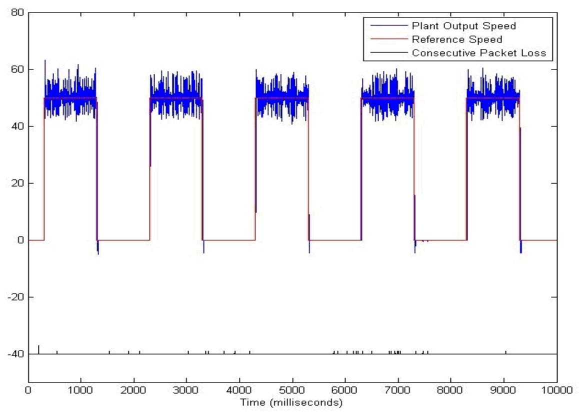

Fig. 9. MBPNCS experiment with 0.33% Packet Loss

As data delay and loss rate over the network is increased, the proposed method holds up well. As an intermediate value, Figure 10 shows the case where 70% of the packets are discarded for being late or dropped. Here, we assume a more realistic scenario where the network is completely down for the given rate, and functioning for the rest.

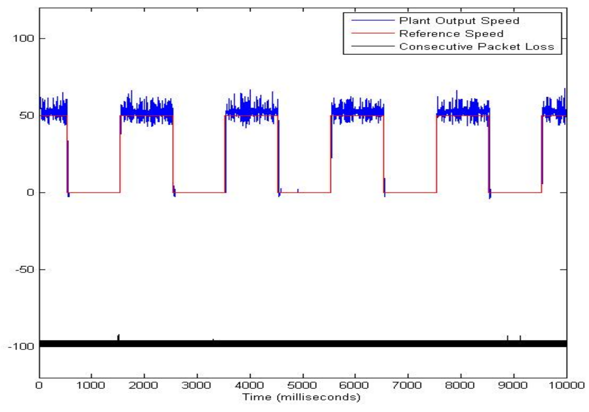

Fig. 10. MBPNCS experiment with 70% Packet Loss

Finally, if the data delay and loss rate is increased further, system becomes unstable, as shown in Figure 11, at around 98% of the packets being delayed or lost. Performance begins to degrade around 90%, and deteriorates steadily.

As a comparison, we also tested the case where the network is not completely down, but sensor to controller node packets and controller to actuator node packets are delayed or lost without any correlation. The result of that case can be seen in Figure 12 with performance similar to the case in Fig. 10

These results show that the performance in simulations and experiments are similar.

5. CONCLUSION

In this work, a novel networked control system, *model based predictive network control system* method (MBP-

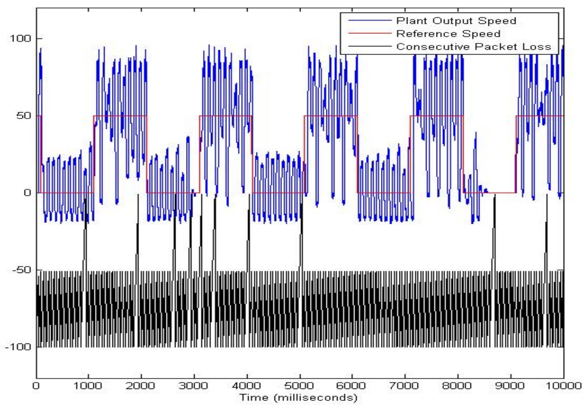

Fig. 11. MBPNCS experiment with 98% Packet Loss

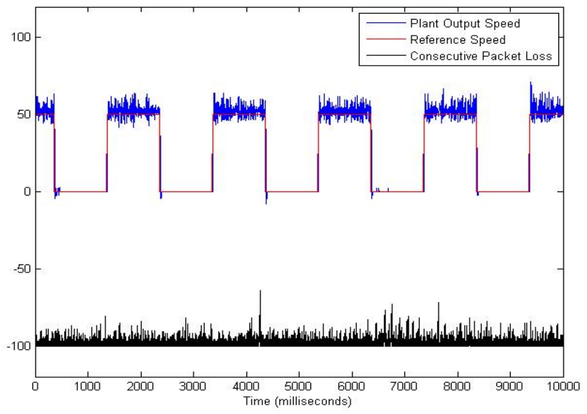

Fig. 12. MBPNCS experiment with 70% Packet Loss (not correlated)

NCS) that is robust as a networked control system (NCS) using non real-time networks is presented. A plant model within the controller node is used to make future control output predictions, and a state machine within the actuator is used to select the proper output to reduce the discrepancy between the actual plant states and those predicted by the model.

This new architecture is applied to a DC servo motor simulation and experimental setup with real-time computers, and various aspects of 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. Although less noticeable with a step reference, this effect may become significant with a constantly changing reference signal. The effect of noise on the control output is small when packet losses are also low, however since the feedback is interrupted, the systems performance is effected by noise when packet loss increases.

REFERENCES

K. J. Astrom and B. Wittenmark. *Computer Controlled Systems Theory and Design.* Prentice Hall, 3. edition, 1997.

M. S. Branicky, S. M. Phillips, and Wei Zhang. Scheduling and feedback co-design for networked control systems. In *IEEE Conf. On Decision and Control*, volume 2, pages 1211–1217, Dec. 2002.

M. S. Branicky, V. Liberatore, and S. M. Phillips. Networked control system co-simulation for co-design. In *American Control Conf.*, volume 4, pages 3341–3346, Jun. 2003.

D. W. Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 1989.

D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control - part i the basic algorithm. *Automatica*, 23(2):137–148, 1987a.

D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control - part ii etentions and interpretations. *Automatica*, 23(2):149–160, 1987b.

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.

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

Q. Ling and M. Lemmon. Robust performance of soft real-time networked control systems with data dropouts. In *IEEE Conf. On Decision and Control*, Dec. 2002.

G. P. Liu, J. X. Mu, and D. Rees D. Networked predictive control of systems with random communication delay. In *UKACC Int. Conf. on Control*, September 2004.

C. Mo-Yuen and Y. Tipsuwan. Gain adaptation of networked dc motor controllers based on qos variations. *IEEE Trans. on Indust. Electronics*, 50(5):936–943, Oct. 2003.

A. T. Naskali and A. Onat. Jitter analysis of model based predictive networked control system. In *6th WSEAS Intl. Conf. on Applied Computer Science*, volume 1, pages 199–206, Dec. 2006.

K. Natori and K. Onishi. Time delay compensation by communication disturbance observer in bilateral teleoperation systems. In *Adv. Motion Control*, pages 218–223, Mar. 2006.

P. Otanez, J. Moyne, and D. Tilbury. Using deadbands to reduce communication in networked control systems. In *American Control Conf.*, 2000.

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

M Toerngren. Fundamentals of implementing real-time control applications in distributed computer systems. *Real-Time Systems*, 14:219–250, 1998.

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

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

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. In *Proc. JUSFA Japan-USA Symposium on Flexible Automation*, July 2000.